PROJECTE FINAL DE CARRERA

Diplomatura de Estadística

Titulo: Implementación de applets para la didáctica de la probabilidad

Autor: Daniel Alcaide Villar

Director: Pedro Delicado, Enrique Romero

Departamento: Estadística y Investigación Operativa, Lenguaje y Sistemas Informáticos

Convocatoria: Junio 2009

Facultat de Matemàtiques i Estadística

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Índice

1. Introducción

La revolución en las comunicaciones ha permitido ya a millones de personas estar mejor informadas, gran parte de ello se puede atribuir a internet ya que este medio crece día a día y con ello el número de páginas web.

Existen programas destinados a poder ejecutarse en una página web. Se trata de pequeños programas hechos en Java, que se transfieren con las páginas web y que el navegador ejecuta en el espacio de la página y que se denominan applets.

Muchas son los ámbitos donde se aplica este tipo de herramienta, y uno de ellos es el de la docencia. Existen plataformas destinadas a este tipo de prácticas donde explican conceptos científicos ilustrados por applets.

Este proyecto trata de explicar conceptos probabilísticos con la ayuda de estas aplicaciones haciendo más ameno el aprendizaje de esta materia. El resultado final son tres applets en Java que resuelven dos problemas de cálculo de probabilidades que se pueden encontrar en esta página web: http://www-eio.upc.es/~delicado/docencia/Daniel_Alcaide/inicio.html

La inquietud por desarrollar este trabajo vino infundada por otro proyecto final de carrera de la diplomatura de Estadística[1] que al igual que éste, se crearon varios applets intentando hacer más comprensible y accesible el cálculo de probabilidades.

1.1. Ejemplos de applets

A continuación se muestran varios ejemplos que con la ayuda de un applet explican conceptos estadísticos y matemáticos.

Statmedia

Statmedia es un conjunto de documentos que reúne material docente multimedia adaptado a una asignatura estándar de Estadística de la Universidad de Barcelona.

La aportación didáctica más importante de Statmedia es la amplia colección de programas redactados en Java, que realizan cálculos y gráficas relacionados con los conceptos importantes en Estadística.

Estos programas se han diseñado con una finalidad pedagógica, ante la necesidad de que el estudiante visualice los conceptos expuestos y experimente él mismo con la manipulación de los datos.

Un ejemplo donde utilizan los applets como medio ilustrativo es el apartado de "Función probabilidad condicionada"[2].

[1] Proyecto final de carrera: Elena Briones (Diplomatura de Estadística - UPC) (2008). *Herramientas informáticas interactivas para la didáctica de la probabilidad*.

[2] Puede encontrar más información sobre este apartado y sobre otros problemas de statmedia en este enlace: http://www.ub.edu/stat/GrupsInnovacio/Statmedia/demo/Statmedia.htm

Imaginemos que en la experiencia de tirar un dado regular supiéramos de antemano que se ha obtenido un número par. Es decir, que se ha verificado el suceso: B = número par. ¿Cuál es ahora la probabilidad de que se verifique el suceso mayor o igual a cuatro (denominado como suceso A)?

Lógicamente, el resultado sería: 2/3.
Por lo tanto, la probabilidad del suceso A = mayor o igual a cuatro se ha modificado.

Evidentemente, ha pasado de ser 1/2 (cuando no tenemos ninguna información previa) a ser 2/3 (cuando sabemos que se ha verificado el suceso B). ¿Cómo podemos anotar esta última probabilidad (2/3)?
Anotaremos P (A|B), que se lee como probabilidad de A condicionada a B.
Así, en este ejemplo,

$$P(A|B) = 2/3$$
$$P(A) = 1/2$$

En términos generales, estamos en condiciones de poder definir la probabilidad condicionada, y lo hacemos como:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

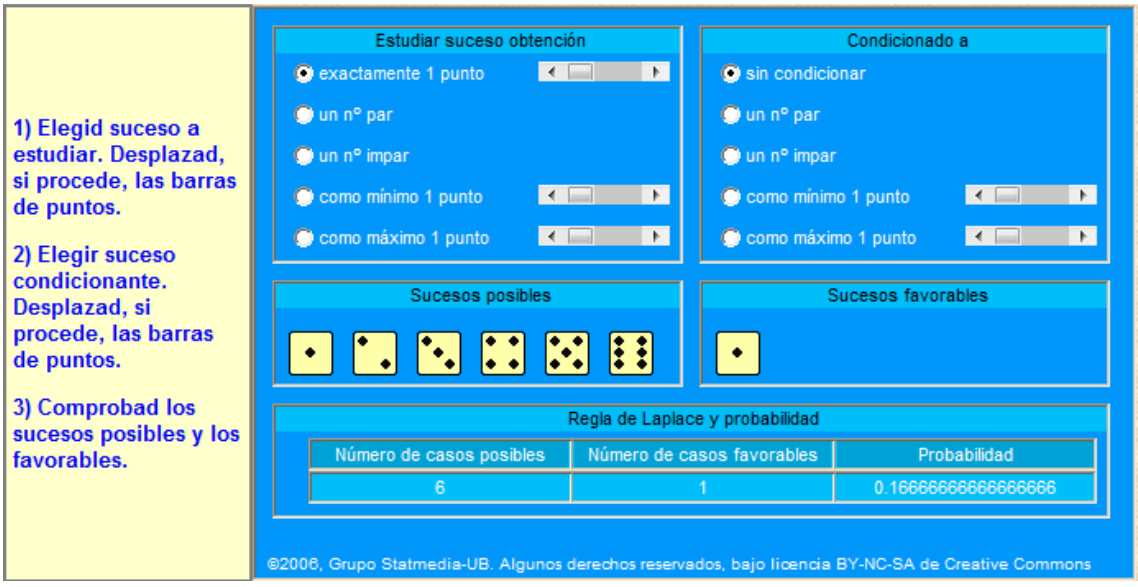


Figura 1. Ejemplo Statmedia

Simply Monty Hall

El problema de Monty Hall[3] es un problema de probabilidad que está inspirado en un concurso de televisión en que se puede ganar un coche como premio. El presentador enseña tres puertas, detrás de una de ellas hay un coche; detrás de las otras dos, cerdos. El presentador pide al concursante que elija solo una puerta y esta se marca, pero todavía no se abre. Una vez escogida, el presentador abre una de las otras puertas y muestra un cerdo. Puede actuar siempre de este modo porque sabe donde se encuentran los premios. Entonces da una última oportunidad de cambiar la puerta al concursante. Para ello el concursante debe plantearse. ¿Cuál sería la opción correcta?

- Quedarse con la puerta inicial.
- Cambiar a la otra puerta
- Es irrelevante cambiar o no cambiar.

Parece que la opción más lógica es la tercera (es irrelevante cambiar o no cambiar), pero la solución correcta no es tan trivial. En su página web explican la solución correcta del problema, aparte de poder comprobarlo con el applet del problema.

Figura 2. Ejemplo Simply Monty Hall

[3] Se puede encontrar más información sobre el problema: http://www.shodor.org/interactivate/activities/simpleMontyHall

Media y mediana[4]

Este applet de Java muestra la relación entre la media y la mediana e ilustra varios aspectos de estos indicadores de tendencia central.

El applet comienza mostrando un histograma de 9 números. Como se puede observar en la Figura 3, los números son los siguientes: 3, 4, 4, 5, 5, 5, 6, 6 y 7. La media y mediana de los números son 5.0.

La media se muestra en el histograma como una pequeña línea azul; la mediana se muestra como una pequeña línea morada. La desviación estándar es 1.15. Una línea roja se extiende una desviación en cada sentido a partir de la media. Puede cambiar los valores del conjunto de datos del histograma con el ratón.

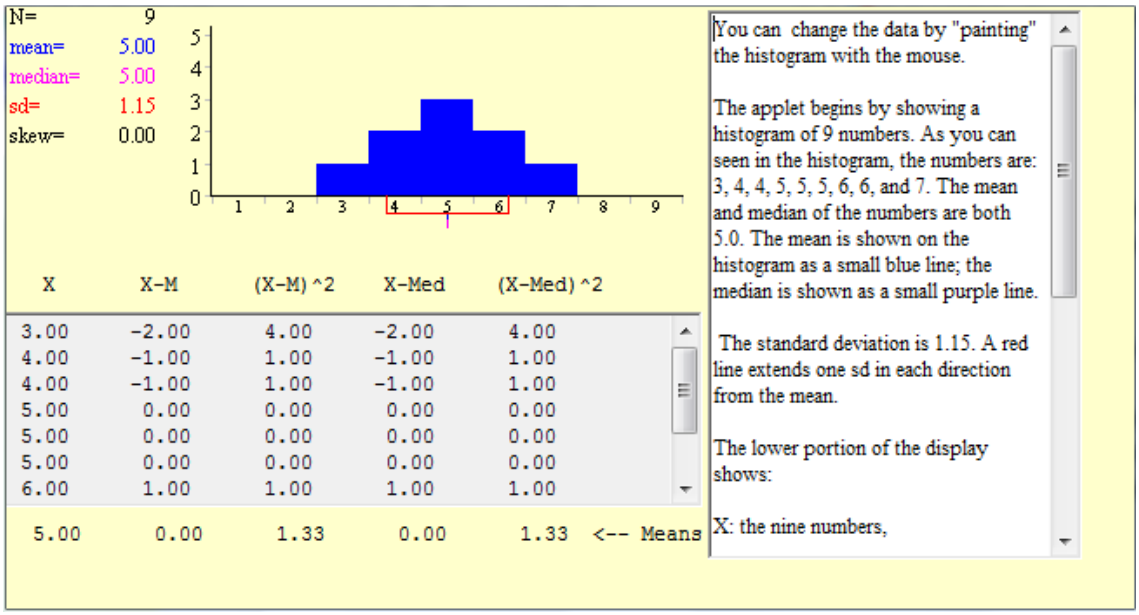


Figura 3. Ejemplo Media y mediana

[4] Toda la explicación y el applet en cuestión lo puede encontrar en este enlace: http://www.ruf.rice.edu/~lane/stat_sim/descriptive/index.html

1.2. Estructura del documento

Este apartado se describe de manera general el contenido de las próximas secciones con el fin de facilitar la comprensión del documento:

2. Descripción del proyecto: Metodología seguida en el proyecto.
3. Problemas: Descripción de los problemas y solución analítica de los problemas.
4. ¿Cómo realizar un applet?: Reseña general de los elementos y explicación de cómo implementar un applet.
5. Implementación de los applets: Resultado de los applets seleccionados y descritos en la sección 3.
6. Conclusiones: Aportaciones personales y posibles extensiones del proyecto
7. Bibliografía: Documentación consultada
8. Anexo: Código fuente de los applets realizados.

2. Descripción del proyecto

El objetivo principal de este proyecto es facilitar la comprensión del cálculo de probabilidades, mediante la elaboración de diferentes applets basados en una temática común. Para ello, fue necesario encontrar diversos problemas que se adecuaran a nuestras necesidades. Estos problemas debían tener, una cierta dificultad para que no resultara irrelevante la utilización de un programa informático y la flexibilidad necesaria para que el mismo problema tuviera diferentes puntos de vista.

Después de decidir los problemas que se iban a tratar ("*El jurado popular*" y *"Colección de cromos")* se implementaron estos problemas en varios applets, añadiendo algunas cuestiones que los enunciados originales no contemplaban.

Finalmente, estos applets se añadieron a una página web donde se pueden visualizar junto a la resolución de los problemas originales.

2.1. Elección de los problemas

Durante el proceso de elección de los problemas y ante la gran cantidad de posibles problemas a tratar, se decide que una buena idea es centrar los problemas sobre un tema en común.

En nuestro caso el tema elegido se centran en la probabilidad discreta ilustrado por dos problemas explicados a continuación:

2.1.1. El jurado popular

Este problema fue extraído del libro de "*Fifty Challenging Problems in Probability with Solutions"*[5]:

En un sistema judicial se plantea la posibilidad de instaurar un jurado popular. Se sabe que un juez con experiencia tiene probabilidad 0,9 de tomar la decisión correcta. También se sabe por los estudios preliminares que un miembro de un jurado sin formación jurídica toma la decisión correcta con probabilidad 0,7. En el caso del jurado popular el veredicto se decide por mayoría simple (los miembros del jurado toman las decisiones de forma independiente).

Si el jurado popular consta de 13 miembros y por otro lado hay un solo juez ¿cuál tendrá una mayor probabilidad de tener un veredicto correcto?

Para resolver este problema es importante conocer la distribución de probabilidad binomial.

[5] Mosteller, F (1965). *Fifty Challenging Problems in Probability with Solutions.* Dover Publication

2.1.2. Colección de cromos

Este problema fue extraído de la colección de problemas de la asignatura de Estadística Matemática 1[6]:

Hemos comenzado una colección de cromos que consta de n=10 cromos diferentes. Compramos los cromos de uno en uno y el vendedor nos asegura que la probabilidad de obtener un cromo de cualquier tipo es la misma. ¿Cuántos cromos esperamos comprar para completar la colección?

Para calcular la esperanza se debe reconocer que distribución sigue la variable aleatoria del problema (número de cromos que se han de comprar hasta encontrar uno faltante en la colección).

2.2. Entorno informático

Como se puede extraer de la introducción, los applets y las páginas web forman parte de este proyecto y ambos se describen mediante un lenguaje informático. Estos dos serán descritos a continuación de una manera conceptual para facilitar la comprensión de los siguientes capítulos.

2.2.1. Programación JAVA

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Un compilador es un programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (típicamente lenguaje máquina). De esta manera un programador puede diseñar un programa en un lenguaje mucho más cercano a como piensa un ser humano, para luego compilarlo a un programa manejable por un ordenador. Este proceso de traducción se conoce como compilación.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

[6] Gómez , G y Delicado, P (2008) Curs d'Estadística Matemàtica 1

El lenguaje JAVA, es el utilizado para programar applets. Los applets se han programado utilizando el entorno Eclipse. Estos conocimientos, tan específicos, no se tenían al inicio del proyecto y se han tenido que adquirir para poder realizarlos acudiendo a manuales como *"Aprenda Java como si estuviera en primero"* [7] y diferentes páginas web [8]que disponen de ejemplos

2.2.2. Lenguaje Extensible de Marcado de Hipertexto (XHTML)

Con el objetivo que las aplicaciones disponibles pudieran estar disponibles al público, se decidió utilizar Internet como medio de difusión y para ello, reflejarlo en una página web.

Una página web, es un documento adaptado para la Web y normalmente forma parte de un sitio web. Su principal característica son los hiperenlaces a otras páginas web, siendo esto el fundamento de la Web.

Una página está compuesta principalmente por información (sólo texto o multimedia) e hiperenlaces; además puede contener o asociar datos de estilo para especificar cómo debe visualizarse o aplicaciones incrustadas para hacerla interactiva.

La página se escribe en un lenguaje de marcado que tenga la capacidad de insertar hiperenlaces, generalmente en HTML (lenguaje de marcas de hipertexto) o XHTML.

El lenguaje utilizado en este proyecto, (XHTML) es un lenguaje de marcado pensado en sustituir a HTML cómo estándar de páginas web. La diferencia más importante entre ambos es que, tanto la información de la página como la forma de presentarla estén claramente separadas

Para realizar estas páginas en lenguaje XHTML se ha utilizado el programa Adobe Dreamweaver[9], conocimiento que se ha tenido que adquirir para este proyecto.

[7] García de Jalón, J (2000) *Aprenda Java como si estuviera en primero.* Escuela de Ingenieros Industriales de San Sebastián(Universidad de Navarra)

[8] En esta página podemos encontrar ejemplos de apples Java:
http://www.roseindia.net/java/example/java/applet/testURL.shtml

[9] La principal fuente consultada fue: Plasencia, L (2008) *Dreamweaver Cs3.* Anaya

3. Problemas

En este apartado se explicarán los problemas de probabilidad que se tratan en este proyecto (*El problema del Jurado Popular o Juez, El problema de la colección de cromos*) y se resolverán los problemas de manera analítica.

3.1. El problema del Jurado Popular o Juez

En un sistema judicial se plantea la posibilidad de instaurar un jurado popular. Se sabe que un juez con experiencia tiene probabilidad 0,9 de tomar la decisión correcta. También se sabe por los estudios preliminares que un miembro de un jurado sin formación jurídica toma la decisión correcta con probabilidad 0,7. En el caso del jurado popular el veredicto se decide por mayoría simple (los miembros del jurado toman las decisiones de forma independiente).

Si el jurado popular consta de 13 miembros y por otro lado hay un solo juez ¿cuál tendrá una mayor probabilidad de tener un veredicto correcto?

3.1.1. Resolución analítica

Reconocemos que, cada miembro del jurado, sólo puede tomar dos valores, inocente o culpable.

Por otra parte también sabemos que cada miembro toma su decisión de manera independiente.

Definimos la variable aleatoria (v.a.) 'X' como: El número de jueces que toman la decisión correcta.

Observamos que $X = \sum_{i=1}^{n} X_i, donde\ X_1, \dots, X_n$ son variables aleatorias independientes idénticamente distribuidas con distribución Bernoulli de parámetro p, B(p). Por lo tanto sabemos que X sigue una distribución Binomial de parámetros, (n=13; p=0,7).

La fórmula del binomio de Newton garantiza que $\sum_{j=0}^{n} P(X = j) = 1$:

$$\sum_{j=0}^{n} P(X = j) = \sum_{j=0}^{n} \binom{n}{j} p^j (1-p)^{n-j} = (p + (1-p))^n = 1$$

Según el enunciado extraemos que la mayoría simple es cuando 7 o más miembros del jurado toman la misma decisión:

$$P(X = 7) = \frac{13!}{7!(13-7)!}\ 0{,}7^7(1-0{,}7)^{(13-7)} = 0{,}103$$

$$P(X = 8) = \frac{13!}{8!(13-8)!}\ 0{,}7^8(1-0{,}7)^{(13-8)} = 0{,}18$$

$$P(X = 9) = \frac{13!}{9!(13-9)!}\ 0{,}7^9(1-0{,}7)^{(13-9)} = 0{,}234$$

$$P(X = 10) = \frac{13!}{10!(13-10)!}\ 0{,}7^{10}(1-0{,}7)^{(13-10)} = 0{,}218$$

$$P(X = 11) = \frac{13!}{11!(13-11)!}\ 0{,}7^{11}(1-0{,}7)^{(13-11)} = 0{,}139$$

$$P(X = 12) = \frac{13!}{12!(13-12)!}\ 0{,}7^{12}(1-0{,}7)^{(13-12)} = 0{,}054$$

$$P(X = 13) = \frac{13!}{13!(13-13)!}\ 0{,}7^{13}(1-0{,}7)^{(13-13)} = 0{,}010$$

En total:

$$P(X \geq 7) = \sum_{i=7}^{13} \frac{13!}{i!(13-i)!}\ 0{,}7^{i}\ 0{,}3^{13-i} = 0.9376248$$

3.2. El problema de la colección de cromos

Hemos comenzado una colección de cromos que consta de n=10 cromos diferentes. Compramos los cromos de uno en uno y el vendedor nos asegura que la probabilidad de obtener un cromo de cualquier tipo es la misma. ¿Cuántos cromos esperamos comprar para completar la colección?

3.2.1. Resolución analítica

Sea X_i = "El número de cromos que hemos de comprar para que salga uno que todavía no tenemos, suponiendo que tenemos i-1". Para i=1,…, n.

X_1 ~ Geom (p_1 = 1)

X_2 ~ Geom (p_2= 1- 1/n)

X_3 ~Geom (p_3 =1- 2/n)

…

X_i ~ Geom (p_i =1- (i-1)/n)

…

X_n ~Geom (p_n = 1- (n-1)/n = 1/n)

El nº de cromos que hemos de comprar es Y= $X_1 + X_2 + x_3 + \ldots + X_n$ y su esperanza es:

$$E\ (Y) = \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} \frac{1}{p_i} = \sum_{i=1}^{n} \frac{1}{1-\frac{i-1}{n}} = \sum_{i=1}^{n} \frac{n}{n-i+1} = n \sum_{j=1}^{n} \frac{1}{j}.$$

Si n=10, $E\ (Y) = 10\ \left(1 + \frac{1}{2} + \cdots + \frac{1}{10}\right) = 10 \cdot 2{,}929 = 29{,}29$

4. ¿Cómo realizar un applet?

El proceso para crear un applet es el que se sigue en cualquier programa informático, salvo particularidades propias del programa como es la interfaz gráfica. Durante la diplomatura de Estadística los programas que se realizaron no contenían este medio de interacción con el usuario.

La Figura 4 muestra un ejemplo del aspecto de un applet al ser ejecutado, marcando los componentes que intervienen en el programa.

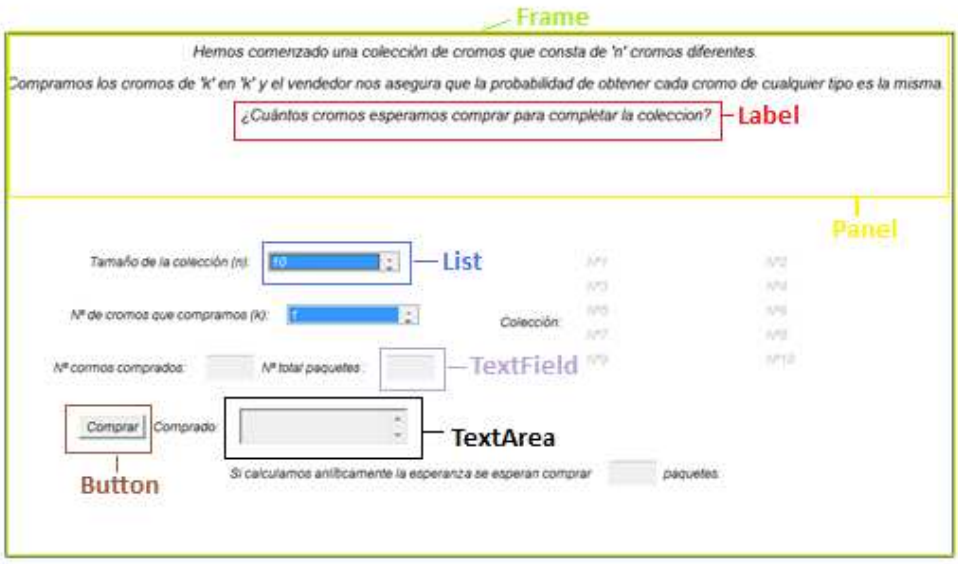


Figura 4. Ejemplo de un applet (Colección de cromos 1)

4.1. Desarrollo de un programa informático

En este apartado trataremos de explicar, paso a paso, cual es el proceso que se sigue para realizar un programa informático.

4.1.1. Formalización del problema y análisis de requerimientos

El primer paso en la realización de cualquier programa informático es la comprensión del problema y la transformación de las preguntas generales en preguntas más concretas. Antes de empezar a programar deberemos tener claras las especificaciones del problema.

Una parte importante es tener un esquema de la parte gráfica que interactuará con el usuario. Hacer un dibujo de los elementos que después saldrán por pantalla, ayudará a colocar los elementos cuando se esté programando, aunque puede haber modificaciones al traducirlo al lenguaje de programación.

A continuación (Figura 5) se muestra el borrador inicial del diseño al problema de 'Jurado Popular o Juez', si nos fijamos en la versión final se notan cambios significativos en el diseño.

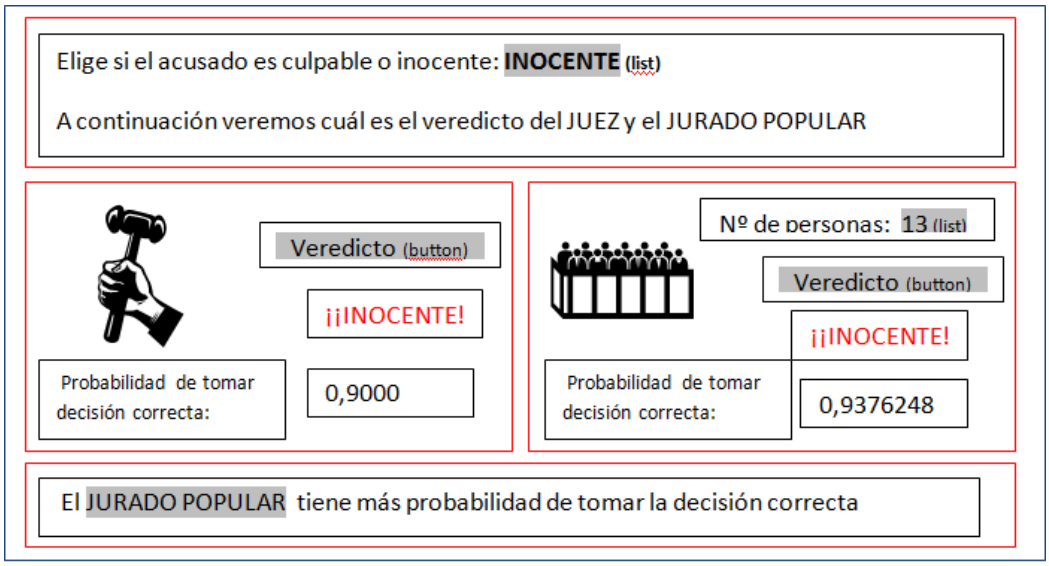


Figura 5. Diseño inicial del problema 'Jurado Popular o Juez'

4.1.2. Diseño del algoritmo

En este paso transformaremos las especificaciones del problema en un conjunto de operaciones y procedimientos definidas en pseudocódigo. Todo algoritmo debe tener en cuenta estas propiedades:

Carácter finito: Un algoritmo siempre debe terminar después de un número finito de pasos.

Precisión: Cada paso de un algoritmo debe estar precisamente definido; las operaciones a llevar a cabo deben ser especificadas de manera rigurosa y no ambigua para cada caso.

Entrada: Un algoritmo tiene cero o más entradas: cantidades que le son dadas antes de que el algoritmo comience, o dinámicamente mientras el algoritmo corre. Estas entradas son tomadas de conjuntos específicos de objetos.

Salida: Un algoritmo tiene una o más salidas: cantidades que tienen una relación específica con las entradas.

Eficacia: También se espera que un algoritmo sea eficaz, en el sentido de que todas las operaciones a realizar en un algoritmo deben ser suficientemente básicas como para que en principio puedan ser hechas de manera exacta y en un tiempo finito por un hombre usando lápiz y papel.

4.1.3. Codificación

La codificación es la operación de escribir la solución del problema (de acuerdo a la lógica del diagrama de flujo o pseudocódigo), en una serie de instrucciones detalladas, en un código reconocible por la computadora. La serie de instrucciones detalladas se le conoce como código fuente, el cual se escribe en un lenguaje de programación o lenguaje de alto nivel.

4.1.4. Prueba y Depuración

Los errores humanos dentro de la programación de computadoras son muchos y aumentan considerablemente con la complejidad del problema. El proceso de identificar y eliminar errores, para dar paso a una solución sin errores se le llama depuración.

La prueba consiste en la captura de datos hasta que el programa no presente errores (los más comunes son los sintácticos y lógicos).

4.2. Applet Java

Hay tres puntos importantes que se deben conocer para realizar el primer applet: los métodos básicos de la clase applet, la interfaz gráfica y la disposición espacial de los objetos (Layout Manager).

4.2.1. Métodos básicos de la clase applet

Para escribir applets Java, hay que utilizar una serie de métodos que controlan la ejecución del applet. Incluso para realizar el applet más sencillo, necesitaremos definir varios métodos:

- **init()**

Este método, se llama automáticamente en cuanto el applet se hace visible. Es llamado sólo una vez. Se encarga de la inicialización de los datos, cargar imágenes o asignación de valores a las variables globales, que solamente debe realizarse una vez.

- **destroy()**

Se llama a este método cuando el *applet* va a ser descargado para liberar los recursos que tenga reservados (excepto la memoria).

- **start()**

Este método es llamado automáticamente en cuanto el applet se hace visible, después de haber sido inicializado.

- **stop()**

Este método es llamado para detener el applet. Se llama cuando el applet desaparece de la pantalla, con objeto de no consumir recursos que no se están usando en ese momento.

4.2.2. Interfaz gráfica (AWT[10])

La interfaz gráfica la forman los eventos, los componentes y la disposición espacial (Layout Manager) de los objetos. A continuación se tratan por separado cada uno, y se explican los distintos modelos que existen.

4.2.2.1. Componentes

Un componente es un objeto con una representación gráfica que puede ser visualizada en la pantalla y que puede interactuar con el usuario. Ejemplo de componentes son los botones, casillas de verificación y las barras de desplazamiento de una típica interfaz gráfica de usuario.

La Figura 6 muestra todos los componentes del AWT y los eventos específicos de cada uno de ellos, así como una breve explicación en qué consiste cada tipo de evento.

COMPONENTE	EVENTOS GENERADOS	SIGNIFICADO
Button	ActionEvent	Clicar en el botón
Checkbox	ItemEvent	Seleccionar o deseleccionar un item
CheckboxMenuItem	ItemEvent	Seleccionar o deseleccionar un item
Choice	ItemEvent	Seleccionar o deseleccionar un item
Component[11]	ComponentEvent	Mover, cambiar tamaño, mostrar u ocultar un componente
	FocusEvent	Obtener o perder el focus
	KeyEvent	Pulsar o soltar una tecla
	MouseEvent	Pulsar o soltar un botón del ratón; entrar o salir de un componente; mover o arrastrar el ratón (tener en cuenta que este evento tiene dos Listener)
Container[12]	ContainerEvent	Añadir o eliminar un componente de un container
List	ActionEvent	Hacer doble click sobre un item de la lista
	ItemEvent	Seleccionar o deseleccionar un item de la lista
MunuItem	ActionEvent	Seleccionar un item de un menú
Scrollbar	AdjustementEvent	Cambiar el valor de la scrollbar
TextComponent[13]	TextEvent	Cambiar el texto
TextField	ActionEvent	Terminar de editar un texto pulsando Intro
Window	WindowEvent	Acciones sobre una ventana: abrir, cerrar, iconizar, restablecer e iniciar el cierre

Figura 6. Relación entre componentes y eventos

[10] Abstract Windows Toolkit es la parte de Java que se ocupa de construir interfaces gráficas de usuario.

[11] La clase Label, Canvas y Graphics heredan los métodos de la clase Component, por lo tanto pueden generar los mismos eventos.

[12] La clase Panel hereda los métodos de la clase Container y puede generar los eventos de la clase Container.

[13] La clase TextField y la clase TextArea heredan de la clase TextComponent, utilizan los mismos eventos y generan eventos TextEvent

A continuación se muestran los componentes utilizados en la construcción de los distintos applets:

Container

Es una clase muy general. Sus métodos los heredan las clases como Frame y Panel, que se suelen utilizar con mucha frecuencia para crear ventanas. Mantiene una lista de objetos que se les va añadiendo. Cuando se le añade un nuevo objeto se incorpora al final de esta lista, salvo que se especifique una posición determinada. En la Figura 7 muestra algunos métodos de Container.

MÉTODOS DE CONTAINER	FUNCIÓN QUE REALIZAN
Component add(Component)	Añade un componente al container
doLayout()	Ejecuta el algoritmo de ordenación del layout manager
Component getComponent(int)	Obtiene el n-ésimo componente en el container
Component getComponentAt(int, int), Component getComponentAt(Point)	Obtiene el componente que contiene un determinado punto
int getComponentCount()	Obtiene el número de componentes en el container
Component[] getComponents()	Obtiene los componentes en este container.
remove(Component), remove(int), removeAll()	Elimina el componente especificado.
setLayout(LayoutManager)	Determina el layout manager para este container

Figura 7. Métodos clase Container

Panel

Es un Container de propósito general. Se puede utilizar tal cual para contener otros componentes y también crear una sub-clase para alguna finalidad específica.

Por defecto, el Layout Manager de Panel es Flow Layout pero en algunos casos se pueden utilizar el GridLayout. Los métodos más importantes que se utilizan con la clase Panel, son métodos heredados de Componet y Container, pues la clase Panel no tiene métodos propios, obviando el constructor (Figura 8).

MÉTODOS DE PANEL	FUNCIÓN QUE REALIZA
Panel(), Panel(LayoutManager miLM)	Constructores de Panel

Figura 8. Métodos clase Panel

Canvas

Es una zona rectangular de la pantalla en la que se puede dibujar y en la que se pueden generar eventos. Desde los objetos de esta clase se pueden llamar a los métodos paint() y repaint() de su super-clase Component. Tampoco tiene eventos propios pero pueden recibir los eventos ComponentEvent que al igual que los métodos perteneces a su super-clase Component.

MÉTODOS DE CANVAS	FUNCIÓN QUE REALIZA
Canvas()	Es el único constructor de esta clase
paint(Graphics g);	Dibuja un rectángulo con el color de background. Lo normal es que las sub-clases de Canvas redefinan este método.

Figura 9. Métodos clase Canvas

Label

Introduce en un container un texto no seleccionable y que no puede editar, por defecto se alinea por la izquierda. La elección de la fuente (Font), de los colores, de la tamaño y posición de Label se realiza con los métodos heredaos de la clase Component. La Figura 10 muestra algunos métodos de la clase Label.

Métodos de Label	Función que realizan
Label(String lbl), Label(String lbl, int align)	Constructores de Label
setAlignement(int align), int getAlignement()	Establecer u obtener la alineación del texto
setText(String txt), String getText()	Establecer u obtener el texto del Label

Figura 10. Métodos clase Label

TextField y TextArea

Muestra texto seleccionable y editable. Se pueden especificar la fuente y los colores de la fuente (foregorund) y de fondo (background). Esta clase se puede generar ActionEvents, y al heredar de la clase Text Component también puede recibir TextEvents. La Figura 11 muestra algunos métodos de la clase TextField y TextArea.

MÉTODOS DE TEXTFIELD	FUNCIÓN QUE REALIZAN
TextField(), TextField(int ncol), TextField(String s), TextField(String s, int ncol)	Constructores de TextField
int getColumns(), setColoumns(int)	Obtiene o establece el número de columnas del TextField
setEchoChar(char c), char getEchoChar(),	Establece, obtiene o pregunta por el carácter utilizado para
boolean echoCharIsSet()	passwords, de forma que no se pueda leer lo tecleado por el usuario
MÉTODOS DE TEXTAREA	**FUNCIÓN QUE REALIZAN**
TextArea(), TextArea(int nfil, int ncol), TextArea(String text), TextArea(String text, int nfil, int ncol)	Constructores de TextArea
setRows(int), setColumns(int), int getRows(), int getColumns()	Establecer y/u obtener los números de filas y de columnas
append(String str), insert(String str, int pos), replaceRange(String s, int i, int f)	Añadir texto al final, insertarlo en una posición determinada y reemplazar un texto determinado

Figura 11. Métodos clase TextField y clase TextArea

Button

Es un componente que al clicar sobre él, se genera un evento de la case ActionEvent. Su aspecto depende de su plataforma (PC, Mac, Unix), pero su funcionalidad siempre se la misma. Los métodos más importantes, están recogidos en la Figura 12.

MÉTODOS DE LA CLASE BUTTON	FUNCIÓN QUE REALIZA
Button(String label) y Button()	Constructores
setLabel(String str), String getLabel()	Permite establecer u obtener la etiqueta del Button
addActionListener(ActionListener al), removeActionListener(ActionListener al)	Permite registrar el objeto que gestionará los eventos, que deberá implementar ActionListener
setActionCommand(String cmd), String getActionCommand()	Establece y recupera un nombre para el objeto Button independiente del label y del idioma

Figura 12. Métodos clase Button

List

La clase List viene definida por una zona de pantalla con varias líneas de las que se muestran sólo algunas y entre las que se pueden hacer la selección simple o múltiple.

Las listas generan eventos de la clase Action Event (al clicar dos veces sobre un ítem o al pulsar return) e ItemEvents (al seleccionar o deseleccionar un ítem). Al gestionar el evento ItemEvent puede preguntar si el usuario estaba pulsando a la vez alguna tecla (Alt,Ctrl,Shift), por ejemplo para hacer una selección multiple. La siguiente tabla (Figura 13) muestra los principales métodos de la clase List.

MÉTODOS DE LIST	FUNCIÓN QUE REALIZA
List(), List(int nl), List(int nl, boolean mult)	Constructor: por defecto una línea y selección simple
add(String), add(String, int), addItem(String), addItem(String, int)	Añadir un item. Por defecto se añaden al final
addActionListener(ActionListener), addItemListener(ItemListener)	Registra los objetos que gestionarán los dos tipos de eventos soportados
insert(String, int)	Inserta un nuevo elemento en la lista
replaceItem(String, int)	Sustituye el item en posición int por el String
delItem(int), remove(int), remove(String), removeAll()	Eliminar uno o todos los items de la lista
int getItemCount(), int getRows()	Obtener el número de items o el número de items visibles
String getItem(int), String[] getItems()	Obtiene uno o todos los elementos de la lista
int getSelectedIndex(), String getSelectedItem(), int[] getSelectedIndexes(), String[] getSelectedItems()	Obtiene el/los elementos seleccionados
select(int), deselect(int)	Selecciona o elimina la selección de un elemento
boolean isIndexSelected(int), boolean isItemSelected(String)	Indica si un elemento está seleccionado o no
boolean isMultipleMode(), setMultipleMode(boolean)	Pregunta o establece el modo de selección múltiple
int getVisibleIndex(), makeVisible(int)	Indicar o establecer si un item es visible

Figura 13. Métodos clase List

Graphics

El único argumento de los métodos update() y paint() es un objeto de esta clase que dispone de otros métodos para soportar dos tipos de gráficos:

- Dibujo de gráficas primitivas (texto, líneas, círculos, rectángulos, polígonos,).
- Presentación de imágenes en formatos *.gif y *.jpeg.

Además, esta clase mantiene un contexto gráfico: un área de dibujo actual, un color de dibujo del fondo (background) y otro de los elementos (foregorund), una fuente con todas sus propiedades, etc. Los ejes están situados en la esquina superior izquierda y las coordenadas se miden siempre en pixels.

Java dispone de métodos para realizar dibujos sencillos, llamados a veces "primitivas" gráficas. La clase Graphics dispone de los métodos para primitivas gráficas reseñados en la Figura 14.

MÉTODO GRÁFICO	FUNCIÓN QUE REALIZAN
drawLine(int x1, int y1, int x2, int y2)	Dibuja una línea entre dos puntos
drawRect(int x1, int y1, int w, int h)	Dibuja un rectángulo (w-1, h-1)
fillRect(int x1, int y1, int w, int h)	Dibuja un rectángulo y lo rellena con el color actual
clearRect(int x1, int y1, int w, int h)	Borra dibujando con el background color
draw3DRect(int x1, int y1, int w, int h, boolean raised)	Dibuja un rectángulo resaltado (w+1, h+1)
fill3DRect(int x1, int y1, int w, int h, boolean raised)	Rellena un rectángulo resaltado (w+1, h+1)
drawRoundRect(int x1, int y1, int w, int h, int arcw, int arch)	Dibuja un rectángulo redondeado
fillRoundRect(int x1, int y1, int w, int h, int arcw, int arch)	Rellena un rectángulo redondeado
drawOval(int x1, int y1, int w, int h)	Dibuja una elipse
fillOval(int x1, int y1, int w, int h)	Dibuja una elipse y la rellena de un color
drawArc(int x1, int y1, int w, int h, int startAngle, int arcAngle)	Dibuja un arco de elipse (ángulos en grados)
fillArc(int x1, int y1, int w, int h, int startAngle, int arcAngle)	Rellena un arco de elipse
drawPolygon(int x[], int y[], int nPoints)	Dibuja y cierra el polígono de modo automático
drawPolyline(int x[], int y[], int nPoints)	Dibuja un polígono pero no lo cierra
fillPolygon(int x[], int y[], int nPoints)	Rellena un polígono

Figura 14. Métodos clase List

4.2.2.2. Eventos

Un evento es una acción que realiza el usuario mientras se está ejecutando el applet. Pueden ser eventos, acciones como clicar en un botón, pasar el ratón por encima de algún componente o utilizar el teclado.

A continuación se muestran los eventos utilizados en la construcción de los distintos applets:

ActionEvent

Los eventos ActionEvent se producen al clicar con el ratón en un botón, al elegir un comando de un menú (MenuItem), al hacer doble clic en un elemento de una lista (List) y al pulsar Intro para introducir un texto en un campo de texto (TextField).

ComponentEvent

Los eventos ComponentEvent se generan cuando un Component de cualquier tipo se muestra, se oculta, o cambia de posición o de tamaño. Los eventos de mostrar u ocultar ocurren cuando se llama el método setVisble(boolean) del componente, pero no cuando se minimiza la ventana.

ContainerEvent

Los containerEvent se generan cada vez que un Component se añade o se retira de un Container. Estos eventos sólo tienen un papel de aviso y no es necesario gestionarlos para que se realice la operación.

TextEvent

Se produce un TextEvent cada vez que cambia algo en un componente de texto (TetArea y TextField). Se puede desear evitar ciertos caracteres y para eso hay que gestionar los eventos correspondientes.

WindowEvent

Se produce un WindowEvent cada vez que se abre, cierra, ioniza, restaura, activa o desactiva una ventana.

4.2.2.3. Layout Manager

El Layout Manager es un objeto que controla como los componentes se sitúan en un Container.

Se debe elegir el Layout Manager que mejor se adecue a las necesidades de la aplicación que se desea desarrollar porque cada uno utiliza un método distinto, para ello se explicarán los cinco tipos que existen.

FlowLayout

FlowLayout es el Layout Manager por defecto para Panel. FlowLayout coloca los componentes uno detrás de otro, en una fila, de izquierda a derecha y de arriba a abajo.

BorderLayout

El BorderLayout define cinco áreas: North, South, East, West y Center. El componente se pude colocar en cada una de las aéreas definidas. Una vez que se determina el tamaño de la ventana, los componentes añadidos en cada zona tratan de ocupar todo el espacio disponible.

GridLayout

Los componentes en el GridLayout se colocan en una matriz de celdas del mismo tamaño y cada componente utiliza todo el espacio disponible de la celda.

CardLayout

Este Layout Manager permite compartir una misma ventana para que distintos componentes puedan ser visualizados sucesivamente. El mismo sistema que aparece en una presentación de diapositivas.

GridBagLayout

El GridBagLayout tiene similitud con el GridLayout, ya que los componentes se colocan en un matriz de celdas con la peculiaridad que cada celda puede tener altura y anchura distintas.

4.3. Ejemplo de un applet

En este apartado trataremos de explicar todo lo que contiene, un código de un applet. Este applet es la solución del problema de la 'Colección de cromos' para una colección.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.util.Random;
```

La palabra **import** indica que son declaraciones que le avisan al compilador que este programa usará diferentes clases.

Por ejemplo, en el caso de *java.applet.** se llaman a todas las clases que pertenecen al paquete (packages) applet.

```
public class Coleccion_cromos extends Applet implements ActionListener {
```

En esta línea se define una nueva clase, donde se especifica que es un applet, y dispone de todos los métodos de la clase applet.

```
private Panel panelPrimero, panelSegundo, panelSegundoPosicion1,
panelSegundoPosicion1Forma1,panelSegundoPosicion1Forma2,
panelSegundoPosicion1Forma3,panelSegundoPosicion1Forma4,
panelSegundoPosicion2,panelSegundoPosicion2Forma1,panelSegundoPosicion2Forma2,
panelTercero, panelTerceroForma1, panelTerceroForma2;

private Label labelColeccion,labelPregunta1,labelPregunta2,labelPregunta3,
labelCromos,labelCompra,labelTamañoColeccion,labelEsperanza1,labelEsperanza2,
labelContado,labelError,labelContadorPaquetes,labelCompleto,label1,
label2,label3,label4,label5,label6,label7,label8,label9,label10;

private List listColeccion,listNumeroCromos;

private Button buttonComprar;

private TextField textfieldContador,textfieldCromos,textfieldEsperanza,
textfieldContadorPaquetes;
```

```
private TextArea textareaCompra,textareaColeccion;

public static Random aux = new Random ();
static final int max_compra=100;
static final int max_coleccion=10;
int contadorCompra=0;
int contadorPaquetes=0;
int contadorColeccion=0;
int [] compra = new int [max_compra];
int [] coleccion= new int [max_coleccion];
```

En esta parte, se declaran todas las variables que se van a utilizar a nivel global. Se pueden ver todos los **Panel, Label, List, Button TextField** y **TexArea** que se utilizarán. Todas estas variables forman parte de la interfaz gráfica. El resto de variables que no son propiamente de la interfaz gráfica son utilizadas para hacer los cálculos propios del programa, y obtener la solución del problema.

A continuación se explica el contenido del método init(). Este método incluye todos los componentes que se cargarán en cuanto se ejecute el applet. No se ha incluido la totalidad del método por la densidad de código.

```
public void init(){
        Font fuenteGrande = new Font( "SeansSerif",Font.ITALIC,15 );
        Font fuentePequeña=new Font ( "SeansSerif",Font.ITALIC,12 );
        this.setLayout(new GridLayout(3,1));
        this.setSize(900,600);
        this.setFont(fuentePequeña);

        panelPrimero =this.createPanel(this);
        panelPrimero.setFont(fuenteGrande);
        labelPregunta1 =addLabelToPanel(panelPrimero,"Hemos comenzado una
        colección de cromos que consta de 'n' cromos diferentes.");
        labelPregunta2 =addLabelToPanel(panelPrimero,"Compramos los cromos de
        'k'  en 'k' y el vendedor nos asegura que la probabilidad de obtener
        cada cromo de cualquier tipo es la misma.");
        labelPregunta3 =addLabelToPanel(panelPrimero,"¿Cuántos cromos
        esperamos comprar para completar la coleccion?");

        …}
```

Este método está definido para incluir un evento, es decir, que se ejecutara cuando el usuario ejecute un evento, en nuestro caso es un botón. Dentro de esta función se ha creado el código que resuelve el problema y devuelve al usuario un valor. Este método se encarga el de gestionar el resto de métdos básicos, como start () y destroy ().

```
public void actionPerformed(ActionEvent event) {
        int i;
        double esp;

        if (event.getActionCommand().equals("buttonComprar")) {

             String listaColeccion=listColeccion.getSelectedItem();
             int numColeccion=Integer.valueOf(listaColeccion).intValue();

             String listaNumeroCromos=listNumeroCromos.getSelectedItem();
             int numCromos=Integer.valueOf(listaNumeroCromos).intValue();

             if (numColeccion>=numCromos){
                esp=esperanza(numColeccion,numCromos);
```

```
                textfieldEsperanza.setText(""+esp);
                 if (numColeccion==contadorColeccion){
                     labelCompleto.setVisible(true);
                 }
                 else{
                     for (i=1;i<=numCromos;i++){
                          compra[i-1]=aleatorio(numColeccion);
                          textareaCompra.append(""+compra[i-1]+", ");
                          contadorCompra++;
                          textfieldContador.setText(""+contadorCompra);
                                 int j=0;

                           while((coleccion[j]!=compra[i1])&&(j<contadorColeccion))
                                    {j++;}
                                    if (coleccion[j]!=compra[i-1]) {
                                        coleccion[j]=compra[i-1];
                                        contadorColeccion++;
                                    }
                          }
                          contadorPaquetes++;
                          textfieldContadorPaquetes.setText(""+contadorPaquetes);
                          ordenarVector(contadorColeccion, coleccion);
                          for (i=0; i<contadorColeccion; i++){
                              if (coleccion[i]==1)label1.setForeground(Color.black);
                              if (coleccion[i]==2)label2.setForeground(Color.black);
                              if (coleccion[i]==3)label3.setForeground(Color.black);
                              if (coleccion[i]==4)label4.setForeground(Color.black);
                              if (coleccion[i]==5)label5.setForeground(Color.black);
                              if (coleccion[i]==6)label6.setForeground(Color.black);
                              if (coleccion[i]==7)label7.setForeground(Color.black);
                              if (coleccion[i]==8)label8.setForeground(Color.black);
                              if (coleccion[i]==9)label9.setForeground(Color.black);
                             if(coleccion[i]==10)label10.setForeground(Color.black);
                          }
                      }
                  }
                  else{
                   panelTerceroForma2.setVisible(true);
                  }
            }
        }
```

Siguiendo un diseño descendente se han incorporado, funciones y acciones auxiliares, para facilitar la programación. Pueden acceder al código de las funciones auxiliares y al resto de métodos de todos los applets en el anexo de este proyecto.

4.4. Incluir un applet en una página HTML

Una característica de los applets es que puede ser utilizado dentro de una página HTML. En este proyecto se han incluido todos los applets en una de ellas.

```
<HTML>
<HEAD>
<TITLE>
Coleccion_cromos_1
</TITLE>
</HEAD>
<BODY>
<APPLET
CODEBASE = "."
CODE     = "Coleccion_cromos_1.class"
WIDTH    = 900
HEIGHT   = 600
ARCHIVE="file:/C:/…/bluejcore.jar, file:/C:…/junit.jar,
file:/C:i/Desktop/PFC/"
>
</APPLET>
</BODY>
</HTML>
```

Para llamar a un applet desde una página HTML hay que utilizar la etiqueta **<**APPLET**>** al comienzo de la declaración y otra al final **</**APPLET**>.**

Dentro de la etiqueta applet el parámetro más importante es CODE que señala el nombre del archivo cuya extensión es .class, y cuyo nombre coincide con el de la clase que describe el applet.

Los valores de los parámetros WIDTH y HEIGHT determinan las dimensiones del applet. En este caso el applet tiene una anchura de 900 y una altura de 600 píxeles.

El atributo ARCHIVE se utiliza para localizar los ficheros .jar que utiliza el applet.

5. Implementación de los applets

Se presenta a continuación el resultado final de los applets desde el punto de vista de un usuario. Estos programas están alojados en una página web[14] dedicada al proyecto (Figura 15). A partir de ahí se pueden acceder a las secciones especificas de cada problema donde se podrá encontrar los applets que se han creado.

Figura 15. Página de inicio de la web

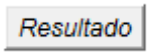
5.1. El problema del Jurado Popular o Juez

Lo primero que nos encontramos en cuanto vemos la primera imagen del applet, es el enunciado del problema. Como nos encontramos con una aplicación que interactúa con el usuario, permite cierta flexibilidad en la resolución del problema, por lo que podemos resolver el problema del enunciado original o cualquier variante que permite, basada en el original.

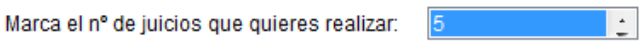
[14] La pueden visitar en: http://www-eio.upc.es/~delicado/docencia/Daniel_Alcaide/inicio.html

Figura 15. Situación Inicial del applet Jurado Popular o Juez

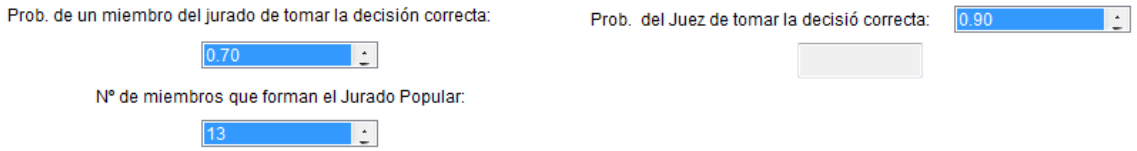
La manera de ejecutar el programa para que empiece a hacer cálculos es utilizando el botón **'Resultado'**



Si no se modifica algún parámetro anteriormente, el programa realizará 5 juicios consecutivos.



Con las probabilidades para cada parte del problema según el problema inicial.



La probabilidad de que un miembro del jurado popular tome la decisión correcta es de 0,7, sabiendo que el jurado popular lo forman 13 miembros y que toman las decisiones por mayoría simple.

Una vez se acabé el proceso de cálculo por parte del applet, aparecen los resultados en la siguiente tabla (Figura 16).

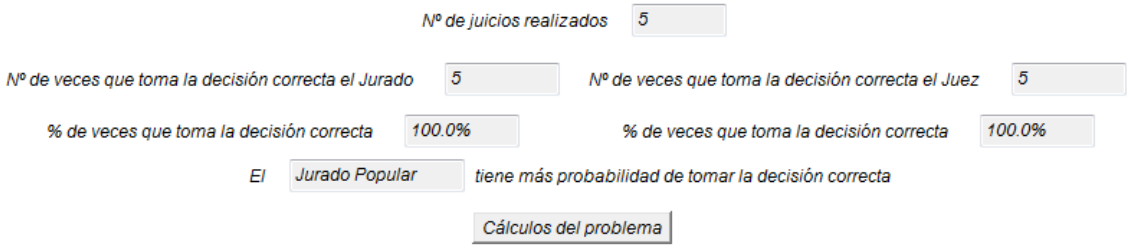
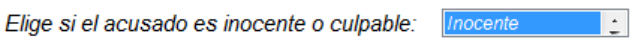


Figura 16. Parte inferior después de ejecutar el applet por defecto

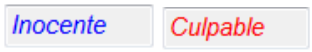
Justo debajo del botón 'Resolver', aparecerá el 'Nº de juicios realizados' hasta el momento, y tanto para el Jurado como para el Juez el 'Nº de veces que toman la decisión correcta' junto con el porcentaje sobre el total.

Al final de esta tabla podemos ver a modo de conclusión una comparación entre quien tiene más probabilidad de tomar la decisión correcta. En definitiva esta conclusión contesta directamente la respuesta del problema, cuál de los dos tiene más probabilidad de tomar la decisión correcta.

Por otra parte, el acusado sobre el que tomarán un veredicto tanto el jurado como el juez aparece 'inocente'.



A raíz de esta decisión, en cada juicio realizado darán un veredicto que saldrá por pantalla:



La decisión que aparece por pantalla solo se refiere al último juicio realizado, ya que sería más difícil representar todos los juicios realizados en este formato, sobretodo si el número de juicios es grande. Por lo tanto si queremos ver la evolución de todos los juicios, mediante este sistema, deberemos realizar un juicio cada vez.

5.2. El problema de la colección de cromos

Para este problema se presentan dos applets. El primero se ciñe al problema original presentado algunas variantes para que se pueda analizar desde distintos puntos de vista. El segundo representa el número de cromos necesario para completar diversas colecciones del mismo tamaño.

Para una colección:

Encontramos al inicio de este applet una breve explicación del enunciado. Este enunciado, tiene la misma temática que el original aunque está modificando para adecuarse a las prestaciones del applet.

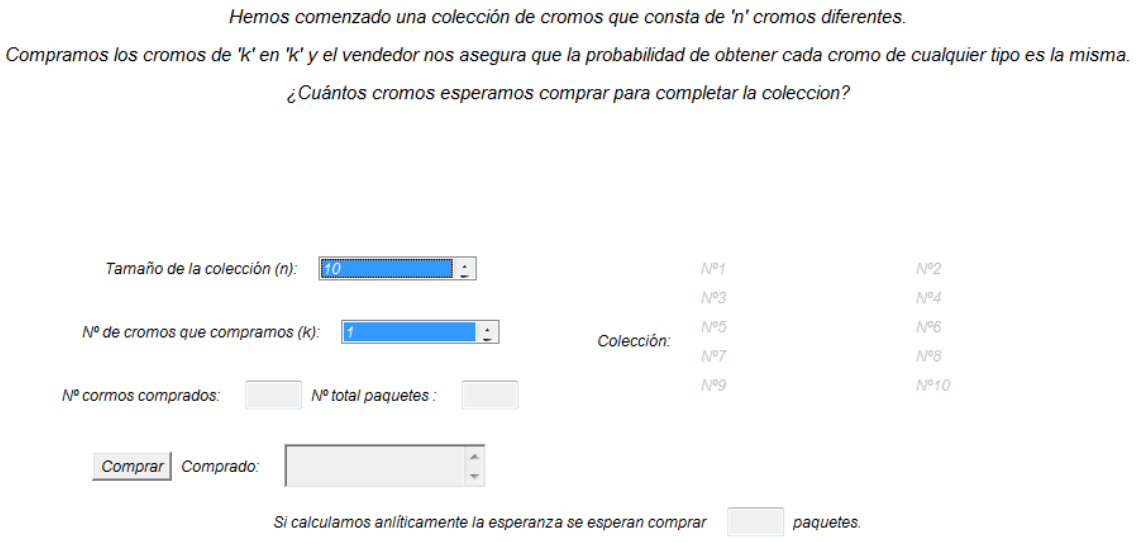
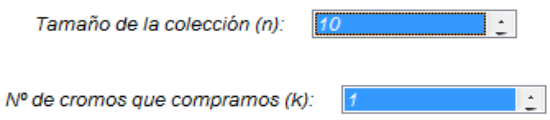


Figura 17. Situación inicial del applet 'colección de cromos' para una colección

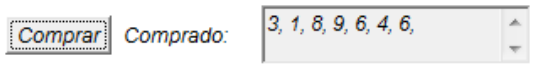
Inicialmente, los parámetros del problema están colocados siguiendo el original.



El 'Tamaño de la colección' es de 10 cromos que se compran en paquetes de un cromo cada vez. Se pueden modificar los parámetros tanto del tamaño de la colección y el 'Nº de cromo que compramos' cada vez.

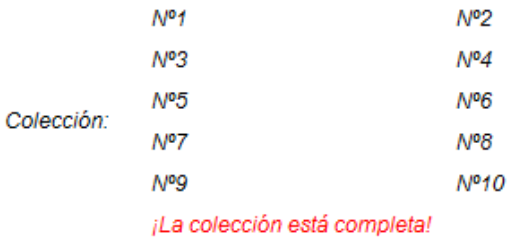
Para ejecutar el programa hay que utilizar el botón **'Comprar'**, que simulará la compra de un paquete de cromos ('Nº de cromos que compramos (k)).

Tanto los cromos repetidos como los que no lo son aparecerán en el recuadro de compra.

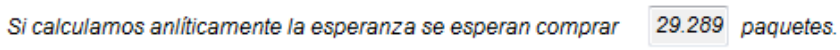


Cada nuevo cromo que no estuviera repetido aparecerá resaltado de color negro, en la tabla colección, el resto de números que todavía no se tengan aparecerán en gris.

Cuando la colección se completa, el programa avisará que se ha completado, y no permite comprar más paquetes.



Por último, y contestando la pregunta del enunciado aparece la esperanza matemática calculada a partir, de los parámetros indicados.



Para diversas colecciones:

Este applet simula ‘c’ colecciones de cromos y las representa en un histrograma el número de cromos comprados para cada colección. Automáticamente calcula la media, la mediana y la desviación tipo de estos datos.

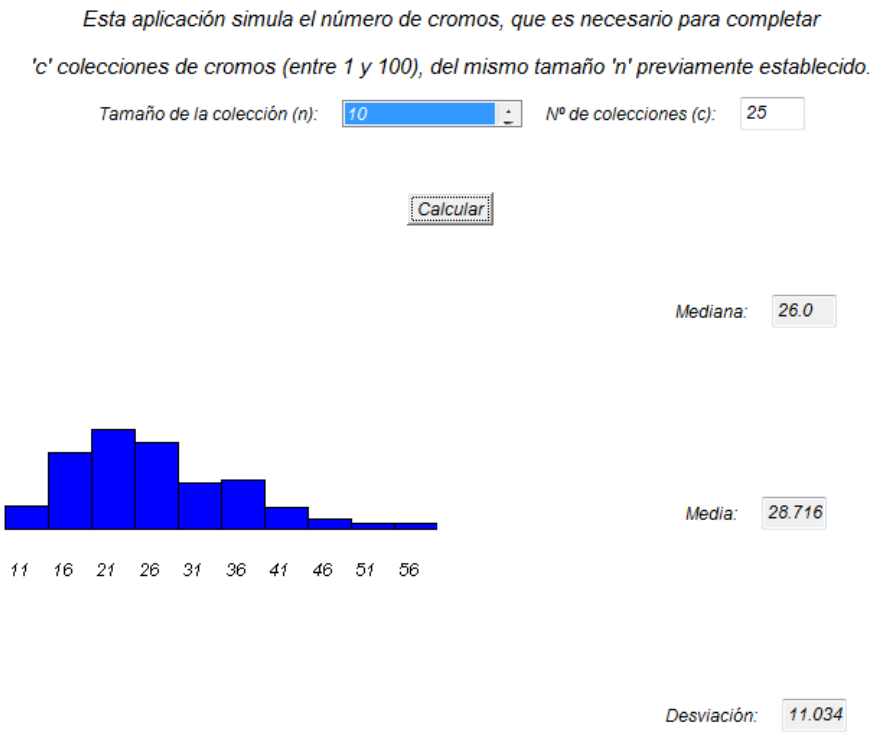


Figura 18. Applet de la simulación de n colecciones

En la parte superior del applet encontramos el enunciado del ejercicio que realiza. Como en el programa anterior se puede modificar el ‘Tamaño de la colección (n)’ y el ‘Nº de colecciones (c)’ que realizan.

Para ejecutar el programa y ver la evolución del histograma debe utilizarse el botón **‘Calcular’**

6. Conclusiones

En este apartado se tratan de exponer cuáles han sido los nuevos conceptos que se han aprendido desarrollando este trabajo y por otro lado, se planten posibles extensiones del proyecto realizado.

6.1. Aportaciones

Este proyecto ha aportado nuevos conceptos y ha fortalecido otros que ya se conocían

- Fortalecer los conocimientos de cálculo de probabilidad

  Al tratar de construir distintos algoritmos que dan solución a un problema de cálculo de probabilidad, se ha fortalecido los conceptos tratados en los problemas.

- Profundizar en lenguaje JAVA

  En la construcción de los applets, se han tenido que aprender nuevas clases propias de los applets, junto a la parte del lenguaje dedicada a la parte estética y la interacción con el usuario.

- Conocer un lenguaje de marcado XHTML

  Para favorecer la difusión de los applets, se decidió crear una página web donde se pudiera ejecutar los applet. Para ello se tuvo que aprender este nuevo lenguaje a un nivel básico para cumplir el objetivo marcado.

6.2. Posibles extensiones del trabajo

Por un lado, se está satisfecho por este proyecto, ya que se han cumplido las expectativas iniciales. Por otro lado, hay aspectos que podrían ampliarse.

- Realizar un mayor número de applets

  Sería interesante poder tratar más problemas de probabilidad y implementarlos en un applet.

- Gráficos más elaborados

  Se podría trabajar el aspecto de los gráficos consiguiendo gráficos más atractivos como son los gráficos en tres dimensiones.

- Utilizar más tipos de componentes y eventos

  Existen muchos tipos de componentes y eventos que podrían figurar en un applet y que en esta ocasión no se han tratado.

7. Bibliografía

- Elena Briones (Diplomatura de Estadística - UPC) (2008). Herramientas informáticas interactivas para la didáctica de la probabilidad.
- Mosteller, F (1965). Fifty Challenging Problems in Probability with Solutions. Dover Publication
- Gómez , G y Delicado, P (2008) Curs d'Estadística Matemàtica 1
- García de Jalón, J (2000) Aprenda Java como si estuviera en primero. Escuela de Ingenieros Industriales de San Sebastián(Universidad de Navarra)
- Plasencia, L (2008) Dreamweaver Cs3. Anaya
- http://www.ub.edu/stat/GrupsInnovacio/Statmedia/demo/Statmedia.htm
- http://www.shodor.org/interactivate/activities/simpleMontyHall
- http://www.ruf.rice.edu/~lane/stat_sim/descriptive/index.html
- http://www.roseindia.net/java/example/java/applet/testURL.shtml

8. Anexo

8.1. Código del applet 'Jurado Popular o Juez'

```
import java.applet.Applet;
import java.applet.AppletContext;
import java.awt.*;
import java.awt.event.*;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Random;

/**
 * Class JuradoPopular_o_Juez - write a description of the class here
 *
 * @author (daniel.alcaide)
 * @version (1.0)
 */
public class JuradoPopular_o_Juez extends Applet implements ActionListener {

        private Panel  panelPrimero,panelPrimeroForma1,panelPrimeroForma2,
             panelPrimeroForma3,panelSegundo,panelSegundoPosicion1,
             panelSegundoPosicion1Forma1,panelSegundoPosicion1Forma11,
             panelSegundoPosicion1Forma2,panelSegundoPosicion1Forma3,
             panelSegundoPosicion1Forma31,panelSegundoPosicion1Forma4,
             panelSegundoPosicion2,panelSegundoPosicion2Forma1,
             panelSegundoPosicion2Forma2,panelTercero,panelTerceroForma1,
             panelTerceroForma2,panelTerceroPos1,panelTerceroPos11,
             panelNumDecJurado,panelTerceroPos12,panelProDecJurado,
             panelTerceroPos2,panelTerceroPos21,panelNumDecJuez,
             panelTerceroPos22,panelProDecJuez,panelTerceroForma3,
             panelTerceroForma31,panelTerceroForma11,panelTerceroForma12;

    private Label labelMargen,labelJuicios,labelEnunciado1,labelEnunciado2,labelElegir,
                  labelProbabilidadJuez,labelProbabilidadJurado,labelNumJurado,

labelProbbilidadJuradoPopular,labelDecision1,labelDecision2,labelNumJuicios,

labelNumDecJurado,labelporcentajeJurado,labelNumDecJuez,labelporcentajeJuez;

    private Button  buttonResultado,buttonMasInformacion;

    private TextField textfieldVeredictoJuez,textfieldVeredictoJurado,

textfieldDecision,textfieldProbabilidadJurado,textfieldNumDecJurado,
                      textfieldPorDecJurado,textfieldNumDecJuez,textfieldPorDecJuez,
                      textfieldNumJuicios;

    private List listVeredicto, listJuicios,listProbabilidadJuez,listProbabilidadJurado,
                 listParticipantesJurado;

    public static Random aux = new Random ();

    float porcentajeJuezCorrecto,porcentajeJuradoCorrecto;

    int  contadorJuezCorrecto=0; int contadorJuradoCorrecto=0; int
contadorTotalJuicios=0;

     public void init(){
        Font fuenteGrande = new Font( "SeansSerif",Font.ITALIC,15 );
        Font fuentePequeña=new Font ( "SeansSerif",Font.ITALIC,12 );
        this.setLayout(new GridLayout(3,1));
        this.setSize(800,650);

        panelPrimero =this.createPanel(this);
        panelPrimero.setLayout(new GridLayout(3,1));

        panelPrimeroForma1=this.addPanelToPanel (panelPrimero);
        panelPrimeroForma1.setFont(fuenteGrande);
        panelPrimeroForma1.setLayout(new GridLayout(3,1));

        labelMargen = addLabelToPanel(panelPrimeroForma1,"");
```

```
        labelEnunciado1 = addLabelToPanel(panelPrimeroForma1,"En un sistema  judicial
se plantea la posibilidad de instaurar un jurado popular.");
        labelEnunciado2= addLabelToPanel(panelPrimeroForma1,"Se quiere saber si el
Jurado popular tendrá más probabilidad de tomar la decisión correcta que el Juez.");
       labelEnunciado1.setAlignment(Label.CENTER);
       labelEnunciado2.setAlignment(Label.CENTER);


        panelPrimeroForma2=this.addPanelToPanel (panelPrimero);
        panelPrimeroForma2.setFont(fuenteGrande);

        labelElegir= addLabelToPanel(panelPrimeroForma2,"Elige si el acusado es inocente
o culpable:");
        listVeredicto=addListStringToPanel (panelPrimeroForma2,1, this);
        listVeredicto.setFont(fuentePequeña);

        panelPrimeroForma3=this.addPanelToPanel (panelPrimero);

        labelJuicios= addLabelToPanel(panelPrimeroForma3,"Marca el nº de juicios que
quieres realizar:");
        listJuicios= addListIntegerToPanel (panelPrimeroForma3, 1, 30,5, this);

        panelSegundo=this.createPanel(this);
        panelSegundo.setLayout(new GridLayout(1,2));


        panelSegundoPosicion1=this.addPanelToPanel (panelSegundo);
        panelSegundoPosicion1.setLayout(new GridLayout(4,1));


        panelSegundoPosicion1Forma1=this.addPanelToPanel (panelSegundoPosicion1);
        panelSegundoPosicion1Forma1.setLayout(new GridLayout(2,1));

        labelProbabilidadJurado= addLabelToPanel (panelSegundoPosicion1Forma1, "Prob. de
un miembro del jurado de tomar la decisión correcta:");
        panelSegundoPosicion1Forma11=this.addPanelToPanel (panelSegundoPosicion1Forma1);

        listProbabilidadJurado = addListToPanel (panelSegundoPosicion1Forma11, 100,
4,this);

        panelSegundoPosicion1Forma2=this.addPanelToPanel (panelSegundoPosicion1);

        labelNumJurado= addLabelToPanel (panelSegundoPosicion1Forma2, "Nº de miembros
que forman el Jurado Popular: ");
        listParticipantesJurado= addListIntegerToPanel (panelSegundoPosicion1Forma2, 1,
30,13, this);


        panelSegundoPosicion1Forma3=this.addPanelToPanel (panelSegundoPosicion1);
        panelSegundoPosicion1Forma3.setLayout(new GridLayout(2,1));

        labelProbbilidadJuradoPopular= addLabelToPanel (panelSegundoPosicion1Forma3,
"Prob. Jurado Popular de tomar la decision correcta:");
        panelSegundoPosicion1Forma31=this.addPanelToPanel (panelSegundoPosicion1Forma3);

        textfieldProbabilidadJurado = addTextFieldToPanel (panelSegundoPosicion1Forma31,
4);
        textfieldProbabilidadJurado.setEditable(false);
        textfieldProbabilidadJurado.setColumns(3);

        panelSegundoPosicion1Forma4=this.addPanelToPanel (panelSegundoPosicion1);

        textfieldVeredictoJurado = addTextFieldToPanel (panelSegundoPosicion1Forma4, 8);
        textfieldVeredictoJurado.setFont(fuenteGrande);
        textfieldVeredictoJurado.setEditable(false);

        panelSegundoPosicion2=this.addPanelToPanel (panelSegundo);
        panelSegundoPosicion2.setLayout(new GridLayout(2,1));

        panelSegundoPosicion2Forma1=this.addPanelToPanel (panelSegundoPosicion2);

        labelProbabilidadJuez= addLabelToPanel (panelSegundoPosicion2Forma1, "Prob.  del
Juez de tomar la decisió correcta:");
        listProbabilidadJuez = addListToPanel (panelSegundoPosicion2Forma1, 100,
8,this);

        panelSegundoPosicion2Forma2=this.addPanelToPanel (panelSegundoPosicion2);
```

```
        textfieldVeredictoJuez = addTextFieldToPanel (panelSegundoPosicion2Forma1, 8);
        textfieldVeredictoJuez.setFont(fuenteGrande);
        textfieldVeredictoJuez.setEditable(false);

        panelTercero =this.createPanel(this);
        panelTercero.setLayout(new GridLayout(3,1));
        panelTercero.setVisible (true);
        panelTercero.setFont(fuentePequeña);

        panelTerceroForma1 =this.addPanelToPanel (panelTercero);
        panelTerceroForma1.setLayout(new GridLayout(2,1));

        panelTerceroForma11=this.addPanelToPanel (panelTerceroForma1);

        buttonResultado=addButtonToPanel(panelTerceroForma11, "Resultado",
"buttonResultado", this);

        panelTerceroForma12=this.addPanelToPanel (panelTerceroForma1);

        labelNumJuicios=addLabelToPanel (panelTerceroForma12, "Nº de juicios
realizados");
        textfieldNumJuicios=addTextFieldToPanel (panelTerceroForma12,6);
        textfieldNumJuicios.setEditable(false);

        panelTerceroForma2=this.addPanelToPanel (panelTercero);
        panelTerceroForma2.setLayout(new GridLayout(1,2));

        panelTerceroPos1=this.addPanelToPanel (panelTerceroForma2);
        panelTerceroPos1.setLayout(new GridLayout(2,1));

        panelTerceroPos11=this.addPanelToPanel (panelTerceroPos1);

        labelNumDecJurado=addLabelToPanel (panelTerceroPos11, "Nº de veces que toma la
decisión correcta el Jurado");
        panelNumDecJurado=addPanelToPanel (panelTerceroPos11);
        textfieldNumDecJurado= addTextFieldToPanel (panelNumDecJurado, 8);
        textfieldNumDecJurado.setEditable(false);

        panelTerceroPos12=this.addPanelToPanel(panelTerceroPos1);

        labelporcentajeJurado=addLabelToPanel (panelTerceroPos12, "% de veces que toma
la decisión correcta");
        panelProDecJurado=addPanelToPanel (panelTerceroPos12);
        textfieldPorDecJurado= addTextFieldToPanel (panelProDecJurado, 8);
        textfieldPorDecJurado.setEditable(false);

        panelTerceroPos2=this.addPanelToPanel (panelTerceroForma2);
        panelTerceroPos2.setLayout(new GridLayout(2,1));

        panelTerceroPos21=this.addPanelToPanel (panelTerceroPos2);

        labelNumDecJuez=addLabelToPanel (panelTerceroPos21, "Nº de veces que toma la
decisión correcta el Juez");
        panelNumDecJuez=addPanelToPanel (panelTerceroPos21);
        textfieldNumDecJuez= addTextFieldToPanel (panelNumDecJuez, 8);
        textfieldNumDecJuez.setEditable(false);

        panelTerceroPos22=this.addPanelToPanel(panelTerceroPos2);

        labelporcentajeJuez=addLabelToPanel (panelTerceroPos22, "% de veces que toma la
decisión correcta");
        panelProDecJuez=addPanelToPanel (panelTerceroPos22);
        textfieldPorDecJuez= addTextFieldToPanel (panelProDecJuez, 8);
        textfieldPorDecJuez.setEditable(false);

        panelTerceroForma3=this.addPanelToPanel (panelTercero);

        panelTerceroForma31=this.addPanelToPanel (panelTerceroForma3);
        labelDecision1 = addLabelToPanel(panelTerceroForma31,"  El");
        textfieldDecision = addTextFieldToPanel (panelTerceroForma31, 14);
        textfieldDecision.setEditable(false);
        labelDecision2= addLabelToPanel (panelTerceroForma31, "tiene más probabilidad de
tomar la decisión correcta");

    }
```

```
     public void actionPerformed(ActionEvent event) {
         double num_aleatorioJuez,num_aleatorioJurado;

         String stringAcusado=listVeredicto.getSelectedItem(); //Obtine el valor de la
lista, inocente o culpable

         int numeroJuicios=listJuicios.getSelectedIndex()+1;

         String numero_listaJuez=listProbabilidadJuez.getSelectedItem(); //Obtine la
probabilidad del juez
         double probabilidadJuez=Double.valueOf(numero_listaJuez).doubleValue();

         int numeroJurado=listParticipantesJurado.getSelectedIndex()+1; //El número de
personas que forman parte del jurado popular
         int mayoriaJurado=((numeroJurado)/2)+1;

         String numero_listaJurado=listProbabilidadJurado.getSelectedItem();
         double probabilidadPersona=Double.valueOf(numero_listaJurado).doubleValue();

         double
probabilidadJuradoPopular=binomialAcum(mayoriaJurado,numeroJurado,numeroJurado,probabili
dadPersona);//Calcula probabilidad

         if (event.getActionCommand().equals("buttonResultado")) {
               contadorTotalJuicios=contadorTotalJuicios+numeroJuicios;
               for (int i=0; i<numeroJuicios;i++){
                 num_aleatorioJuez = probabilidad ();
                 num_aleatorioJurado = probabilidad ();
                probabilidadJuradoPopular = (double)Math.round(probabilidadJuradoPopular
* 100)/100;
               textfieldProbabilidadJurado.setText(""+probabilidadJuradoPopular);
             if ((stringAcusado=="Inocente")&&(probabilidadJuez>=num_aleatorioJuez)) {
               textfieldVeredictoJuez.setText("Inocente");
               textfieldVeredictoJuez.setForeground(Color.blue);
               contadorJuezCorrecto++;
           }
             if ((stringAcusado=="Inocente")&&(probabilidadJuez<num_aleatorioJuez)) {
               textfieldVeredictoJuez.setText("Culpable");
               textfieldVeredictoJuez.setForeground(Color.red);
           }
             if ((stringAcusado=="Culpable")&&(probabilidadJuez>=num_aleatorioJuez)) {
               textfieldVeredictoJuez.setText("Culpable");
               textfieldVeredictoJuez.setForeground(Color.blue);
               contadorJuezCorrecto++;
           }
             if ((stringAcusado=="Culpable")&&(probabilidadJuez<num_aleatorioJuez)) {
               textfieldVeredictoJuez.setText("Inocente");
               textfieldVeredictoJuez.setForeground(Color.red);
           }
             if
((stringAcusado=="Inocente")&&(probabilidadJuradoPopular>=num_aleatorioJurado)) {
               textfieldVeredictoJurado.setText("Inocente");
               textfieldVeredictoJurado.setForeground(Color.blue);
               contadorJuradoCorrecto++;
           }
             if
((stringAcusado=="Inocente")&&(probabilidadJuradoPopular<num_aleatorioJurado)) {
               textfieldVeredictoJurado.setText("Culpable");
               textfieldVeredictoJurado.setForeground(Color.red);
           }
             if
((stringAcusado=="Culpable")&&(probabilidadJuradoPopular>=num_aleatorioJurado)) {
               textfieldVeredictoJurado.setText("Culpable");
               textfieldVeredictoJurado.setForeground(Color.blue);
               contadorJuradoCorrecto++;
           }
             if
((stringAcusado=="Culpable")&&(probabilidadJuradoPopular<num_aleatorioJurado)) {
               textfieldVeredictoJurado.setText("Inocente");
               textfieldVeredictoJurado.setForeground(Color.red);
           }

             if (probabilidadJuez>=probabilidadJuradoPopular) {
               textfieldDecision.setText("Juez");
               textfieldDecision.setForeground(Color.blue);
           }
             else {
```

```
                textfieldDecision.setText("Jurado Popular");
                textfieldDecision.setForeground(Color.black);
            }

        }

porcentajeJuradoCorrecto=(Float.valueOf(contadorJuradoCorrecto).floatValue()/Float.value
Of(contadorTotalJuicios).floatValue())*100;

porcentajeJuezCorrecto=(Float.valueOf(contadorJuezCorrecto).floatValue()/Float.valueOf(c
ontadorTotalJuicios).floatValue())*100;
         textfieldNumJuicios.setText(""+contadorTotalJuicios);
         textfieldNumDecJurado.setText(""+contadorJuradoCorrecto);
         textfieldNumDecJuez.setText(""+contadorJuezCorrecto);
         textfieldPorDecJurado.setText (porcentajeJuradoCorrecto+"%");
         textfieldPorDecJuez.setText (porcentajeJuezCorrecto+"%");
         }

    }
    public void start()
    {
        // provide any code requred to run each time
        // web page is visited
    }


    public void stop()
    {
        // provide any code that needs to be run when page
        // is replaced by another page or before JApplet is destroyed
    }

    public void destroy()
    {
        // provide code to be run when JApplet is about to be destroyed.
    }

    public Panel createPanel (Applet applet) {
        Panel panel = new Panel();
        applet.add(panel);
        return panel;
    }

    public Panel addPanelToPanel (Panel panelp) {
        Panel panel = new Panel();
        panelp.add(panel);
        return panel;
    }

    public List addListStringToPanel (Panel panel, int fil_vis, ActionListener listener)
{
        List list=new List (fil_vis, false);
        list.add("Inocente");
        list.add("Culpable");
        list.select (0);
        panel.add(list);
        list.addActionListener(listener);
        return list;
    }

    public List addListToPanel
                (Panel panel, int fil_in, int selec, ActionListener listener ) {
        int i;
        List list = new List (1, false);
        for (i=50; i<fil_in; i=i+5){
                list.add("0."+(i), i);
       }
        if (i==100){
           list.add("1", i);
        }
       list.makeVisible (selec);
       list.select (selec);
       list.addActionListener(listener);
       panel.add(list);
       return list;
   }
```

```
   public Label addLabelToPanel(Panel panel, String text) {
       Label label = new Label(text);
       panel.add(label);
       return label;
    }

    public Button addButtonToPanel
                    (Panel panel, String text, String actionCommand, ActionListener
listener) {
        Button button = new Button(text);
        button.setActionCommand(actionCommand);
        button.addActionListener(listener);
        panel.add(button);
        return button;
    }

    public TextField addTextFieldToPanel(Panel panel, int npositions) {
        TextField textfield = new TextField(npositions);
        panel.add(textfield);
        return textfield;
    }

    public List addListIntegerToPanel
                  (Panel panel, int fil_vis, int fil_in,int defecto, ActionListener
listener ) {
        int i;
        List list = new List (fil_vis, false);

        for (i=1; i<=fil_in; i++){
            list.add(""+(i), i);
        }
       list.makeVisible (defecto-1);
       list.select (defecto-1);
       list.addActionListener(listener);
       panel.add(list);
       return list;
    }

    public double probabilidad () {
       double p = aux.nextDouble();
       return (p);
    }

    public  double binomial(int x, int n, double p) {
        long comb = combinatoria(n,x);
        return (double) comb * Math.pow(p,x) * Math.pow(1-p,n-x);
    }

    public  long combinatoria(int n, int x ) {
        int dif, term;
        double comb = 1;

        dif =  Math.max(x , n - x);
        term = Math.min(x , n - x);

        for (int i = 1;  i <= term ; i++) {
            int aux = i + dif;
            comb *= (double) ( i + dif ) / i;
        }

        return (long) comb;
    }

    public double binomialAcum(int xi, int xf, int n, double p) {
         double sum = 0.0;
         for (int x = xi; x <= xf; x++) {
             sum =sum+ binomial(x,n,p);
         }
         return sum;
    }


}
```

8.2. Código del applet 'Cromos 1'

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.util.Random;

/**
 * Class Cromos - write a description of the class here
 *
 * @author (your name)
 * @version (a version number)
 */
public class Cromos_1 extends Applet implements ActionListener {

       private Panel
panelPrimero,panelSegundo,panelSegundoPosicion1,panelSegundoPosicion1Forma1,

panelSegundoPosicion1Forma2,panelSegundoPosicion1Forma3,panelSegundoPosicion1Forma4,

panelSegundoPosicion2,panelSegundoPosicion2Forma1,panelSegundoPosicion2Forma2,
            panelTercero,panelTerceroForma1,panelTerceroForma2;

    private Label labelColeccion,labelPregunta1,labelPregunta2,labelPregunta3,

labelCromos,labelCompra,labelTamañoColeccion,labelEsperanza1,labelEsperanza2,
            labelContador,labelError,labelContadorPaquetes,labelCompleto,
            label1,label2,label3,label4,label5,label6,label7,label8,label9,label10;

    private List listColeccion,listNumeroCromos;

    private Button buttonComprar;

    private TextField textfieldContador,textfieldCromos,textfieldEsperanza,
                      textfieldContadorPaquetes;

    private TextArea textareaCompra,textareaColeccion;

    public static Random aux = new Random ();
    static final int max_compra=100;
    static final int max_coleccion=10;
    int contadorCompra=0;
    int contadorPaquetes=0;
    int contadorColeccion=0;
    int [] compra = new int [max_compra];
    int [] coleccion= new int [max_coleccion];

    public void init(){
        Font fuenteGrande = new Font( "SeansSerif",Font.ITALIC,15 );
        Font fuentePequeña=new Font ( "SeansSerif",Font.ITALIC,12 );
        this.setLayout(new GridLayout(3,1));
        this.setSize(900,600);
        this.setFont(fuentePequeña);

        panelPrimero =this.createPanel(this);
        panelPrimero.setFont(fuenteGrande);
        labelPregunta1 =addLabelToPanel(panelPrimero,"Hemos comenzado una colección de
cromos que consta de 'n' cromos diferentes.");
        labelPregunta2 =addLabelToPanel(panelPrimero,"Compramos los cromos de 'k' en 'k'
y el vendedor nos asegura que la probabilidad de obtener cada cromo de cualquier tipo es
la misma.");
        labelPregunta3 =addLabelToPanel(panelPrimero,"¿Cuántos cromos esperamos comprar
para completar la coleccion?");

        panelSegundo=this.createPanel (this);
        panelSegundo.setLayout(new GridLayout(1,2));

        panelSegundoPosicion1= this.addPanelToPanel (panelSegundo);
        panelSegundoPosicion1.setLayout(new GridLayout(4,1));

        panelSegundoPosicion1Forma1=this.addPanelToPanel(panelSegundoPosicion1);
```

```
        labelTamañoColeccion =addLabelToPanel (panelSegundoPosicion1Forma1, "Tamaño de
la colección (n):");
        listColeccion= addListIntegerToPanel (panelSegundoPosicion1Forma1, 1, 10,10,
this);

        panelSegundoPosicion1Forma2=this.addPanelToPanel(panelSegundoPosicion1);

        labelCromos =addLabelToPanel (panelSegundoPosicion1Forma2, "Nº de cromos que
compramos (k):");
        listNumeroCromos= addListIntegerToPanel (panelSegundoPosicion1Forma2, 1, 10,1,
this);

        panelSegundoPosicion1Forma3=this.addPanelToPanel(panelSegundoPosicion1);

        labelContador=addLabelToPanel (panelSegundoPosicion1Forma3, "Nº cormos
comprados: ");
        textfieldContador=addTextFieldToPanel (panelSegundoPosicion1Forma3, 3);
        textfieldContador.setEditable(false);
        labelContadorPaquetes=addLabelToPanel (panelSegundoPosicion1Forma3, "Nº total
paquetes : ");
        textfieldContadorPaquetes=addTextFieldToPanel (panelSegundoPosicion1Forma3, 3);
        textfieldContadorPaquetes.setEditable(false);

        panelSegundoPosicion1Forma4=this.addPanelToPanel(panelSegundoPosicion1);

        buttonComprar=addButtonToPanel(panelSegundoPosicion1Forma4, "Comprar",
"buttonComprar", this);
        labelCompra=addLabelToPanel (panelSegundoPosicion1Forma4, "Comprado: ");
        textareaCompra=addTextAreaToPanel (panelSegundoPosicion1Forma4,1,20);
        textareaCompra.setEditable(false);

        panelSegundoPosicion2=this.addPanelToPanel(panelSegundo);

        panelSegundoPosicion2Forma1=this.addPanelToPanel(panelSegundoPosicion2);

        labelColeccion=addLabelToPanel (panelSegundoPosicion2Forma1, "Colección:");

        panelSegundoPosicion2Forma2=this.addPanelToPanel(panelSegundoPosicion2);
        panelSegundoPosicion2Forma2.setLayout(new GridLayout(6,2));

       label1=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº1");
        label1.setForeground(Color.lightGray);

        label2=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº2");
        label2.setForeground(Color.lightGray);

        label3=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº3");
        label3.setForeground(Color.lightGray);

        label4=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº4");
        label4.setForeground(Color.lightGray);

        label5=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº5");
        label5.setForeground(Color.lightGray);

        label6=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº6");
        label6.setForeground(Color.lightGray);

        label7=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº7");
        label7.setForeground(Color.lightGray);

        label8=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº8");
        label8.setForeground(Color.lightGray);

        label9=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº9");
        label9.setForeground(Color.lightGray);

        label10=addLabelToPanel (panelSegundoPosicion2Forma2,"Nº10");
        label10.setForeground(Color.lightGray);

        labelCompleto=addLabelToPanel (panelSegundoPosicion2Forma2,"¡La colección está
completa!");
        labelCompleto.setForeground(Color.red);
        labelCompleto.setVisible(false);

        panelTercero=this.createPanel (this);
        panelTercero.setLayout(new GridLayout(2,1));
```

```
        panelTerceroForma1=this.addPanelToPanel(panelTercero);

        labelEsperanza1=addLabelToPanel (panelTerceroForma1, "Si calculamos
anlíticamente la esperanza se esperan comprar");
        textfieldEsperanza=addTextFieldToPanel (panelTerceroForma1, 3);
        textfieldEsperanza.setEditable(false);
        labelEsperanza2=addLabelToPanel (panelTerceroForma1, "paquetes.");

        panelTerceroForma2=this.addPanelToPanel(panelTercero);
        panelTerceroForma2.setVisible(false);
        labelError=addLabelToPanel(panelTerceroForma2,"¡El número de cromos comprados de
un paquete, no puede superar al número de la coleccion!");
        labelError.setForeground(Color.red);

    }
    public void actionPerformed(ActionEvent event) {
        int i;
        double esp;

        if (event.getActionCommand().equals("buttonComprar")) {

             String listaColeccion=listColeccion.getSelectedItem();
             int numColeccion=Integer.valueOf(listaColeccion).intValue(); //Nº cromos de
la colección

             String listaNumeroCromos=listNumeroCromos.getSelectedItem();
             int numCromos=Integer.valueOf(listaNumeroCromos).intValue(); //Nº de cromos
que compramos cada vez

             if (numColeccion>=numCromos){
                esp=esperanza(numColeccion,numCromos);// calcula cuantos paquetes
esperamos comprar
               textfieldEsperanza.setText(""+esp);
                if (numColeccion==contadorColeccion){
                    labelCompleto.setVisible(true);
                }
                else{
                    for (i=1;i<=numCromos;i++){
                         compra[i-1]=aleatorio(numColeccion);
                         textareaCompra.append(""+compra[i-1]+", ");
                         contadorCompra++;
                         textfieldContador.setText(""+contadorCompra);
                                  int j=0;
                         while ((coleccion[j]!=compra[i-
1])&&(j<contadorColeccion)){j++;}
                                if (coleccion[j]!=compra[i-1]) {
                                    coleccion[j]=compra[i-1];
                                    contadorColeccion++;
                                }
                    }
                    contadorPaquetes++;
                    textfieldContadorPaquetes.setText(""+contadorPaquetes);
                    ordenarVector(contadorColeccion, coleccion);
                    for (i=0; i<contadorColeccion; i++){
                          if (coleccion[i]==1) label1.setForeground(Color.black);
                          if (coleccion[i]==2) label2.setForeground(Color.black);
                          if (coleccion[i]==3) label3.setForeground(Color.black);
                          if (coleccion[i]==4) label4.setForeground(Color.black);
                          if (coleccion[i]==5) label5.setForeground(Color.black);
                          if (coleccion[i]==6) label6.setForeground(Color.black);
                          if (coleccion[i]==7) label7.setForeground(Color.black);
                          if (coleccion[i]==8) label8.setForeground(Color.black);
                          if (coleccion[i]==9) label9.setForeground(Color.black);
                          if (coleccion[i]==10) label10.setForeground(Color.black);
                      }
                }
             }
             else{
                panelTerceroForma2.setVisible(true);
             }
        }
    }
    public void start()
    {
        // provide any code requred to run each time
        // web page is visited
```

```
    }

    /**
     * Called by the browser or applet viewer to inform this JApplet that
     * it should stop its execution. It is called when the Web page that
     * contains this JApplet has been replaced by another page, and also
     * just before the JApplet is to be destroyed.
     */
    public void stop()
    {
        // provide any code that needs to be run when page
        // is replaced by another page or before JApplet is destroyed
    }

    public void destroy()
    {
        // provide code to be run when JApplet is about to be destroyed.
    }
    public Panel createPanel (Applet applet) {
        Panel panel = new Panel();
        applet.add(panel);
        return panel;
    }
    public Panel addPanelToPanel (Panel panelp) {
        Panel panel = new Panel();
        panelp.add(panel);
        return panel;
    }
    public Label addLabelToPanel(Panel panel, String text) {
       Label label = new Label(text);
       panel.add(label);
       return label;
    }
    public List addListIntegerToPanel(Panel panel, int fil_vis, int fil_in, int ver,
ActionListener listener ) {
        int i;
        List list = new List (fil_vis, false);

        for (i=1; i<=fil_in; i++){
            list.add(""+(i), i);
        }
       list.makeVisible (ver-1);
       list.select (ver-1);
       list.addActionListener(listener);
       panel.add(list);
       return list;
    }
    public Button addButtonToPanel(Panel panel, String text, String actionCommand,
ActionListener listener) {
        Button button = new Button(text);
        button.setActionCommand(actionCommand);
        button.addActionListener(listener);
        panel.add(button);
        return button;
    }
    public TextField addTextFieldToPanel(Panel panel, int npositions) {
        TextField textfield = new TextField(npositions);
        panel.add(textfield);
        return textfield;
    }
    public TextArea addTextAreaToPanel(Panel panel, int nfil,int ncol) {
        TextArea textarea = new TextArea(nfil,ncol);
        panel.add(textarea);
        return textarea;
    }
    public int aleatorio (int grado)
        {
            int a =(int) (aux.nextDouble()*grado);
            return a+1;
    }
    public void ordenarVector (int maxn,int [] vector){
        int min,i,imin,j;
        for (i=0; i< maxn-1; i++){
            min=vector[i];
            imin=i;
            for (j=i+1;j<maxn; j++){
                if (vector[j]<min){
```

```
                    min=vector[j];
                    imin=j;
                }
            }
            vector[imin]=vector[i];
            vector[i]=min;
        }
    }

    public double esperanza (int nColeccion,int nCromos){
        double sum=0;
        for (int i=1;i<=nColeccion;i++){
                sum=sum+(1.0/i);
        }
        return (sum*nColeccion)/nCromos;
    }


}

//--- Clase para ejecutarlo sin necesidad de navegador

class Main_Cromos_1 {

    private static Cromos_1 myApplet = new Cromos_1();

    // Programa principal (permite standalone)
    public static void main (String[] args) {
        Frame myFrame = new Frame("Cromos para una una colección");
        // Acciones al cerrar la aplicacion
        myFrame.addWindowListener(new WindowAdapter() {
            // Redefine el metodo windowClosing
            public void windowClosing(WindowEvent event) {
                myApplet.stop();
                myApplet.destroy();
                System.exit(0);
            }
        });
        myFrame.add(myApplet);//, BorderLayour.CENTER);
        myFrame.setSize(1000,600);
        myFrame.setLocation(100,50);
        myApplet.init();
        myApplet.start();
        myFrame.setVisible(true);
    }

}
```

8.3. Código del applet 'Cromos 2'

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.io.*;
import java.util.Random;
import java.awt.Image;


public class cromos_2 extends Applet implements ActionListener {

        static final int max_totalColecciones=10000;
        static final int max_coleccion=10;

        private Panel panelPrimero, panelPrimeroForma1,
panelPrimeroForma2,panelPrimeroForma3,

panelSegundo,panelSegundoPosicion1,panelSegundoPosicion2,panelSegundoPosicion2Forma1,

panelSegundoPosicion2Forma2,panelSegundoPosicion1Forma1,panelSegundoPosicion2Forma3,
                    panelPrimeroForma31;

        public Label labelTamañoColeccion,labelNumeroColecciones,labelMediana,labelMedia,
```

```
                    labelDesviacion,labelError,labelEnunciado1,labelEnunciado2;

        public TextField
textfieldNumeroColecciones,textfiedlMediana,textfiedlMedia,textfiedlDesviacion;

        public List listColeccion;

        public Button calcular;

        public static Random aux = new Random ();

        int contadorTotalColecciones=0;

        int [] totalColecciones= new int [max_totalColecciones];

    private CanvasHisto canvasHistograma;
    public void init(){

        Font fuenteGrande = new Font( "SeansSerif",Font.ITALIC,15 );
         Font fuentePequeña=new Font ( "SeansSerif",Font.ITALIC,12 );
         this.setSize(800,700);
            this.setFont(fuentePequeña);

            panelPrimero=this.createPanel(this);
               panelPrimero.setLayout(new GridLayout(3,1));


               panelPrimeroForma1=addPanelToPanel (panelPrimero);
               panelPrimeroForma1.setFont(fuenteGrande);
            panelPrimeroForma1.setLayout(new GridLayout(2,1));
            labelEnunciado1=addLabelToPanel(panelPrimeroForma1,"Esta aplicación simula el
número de cromos, que es necesario para completar");
            labelEnunciado2 =addLabelToPanel(panelPrimeroForma1,"'c' colecciones de
cromos (entre 1 y 100), del mismo tamaño 'n' previamente establecido.");
            labelEnunciado1.setAlignment(Label.CENTER);
            labelEnunciado2.setAlignment(Label.CENTER);

            panelPrimeroForma2=addPanelToPanel(panelPrimero);
               labelTamañoColeccion =addLabelToPanel (panelPrimeroForma2, "Tamaño de la
colección (n):");
            listColeccion= addListIntegerToPanel (panelPrimeroForma2, 1, 10,10, this);
            labelNumeroColecciones=addLabelToPanel (panelPrimeroForma2,"   Nº de
colecciones (c):");
            textfieldNumeroColecciones=addTextFieldToPanel (panelPrimeroForma2,3);
            textfieldNumeroColecciones.setText("1");


            panelPrimeroForma3=addPanelToPanel (panelPrimero);
            panelPrimeroForma3.setLayout(new GridLayout(2,1));
            panelPrimeroForma31=addPanelToPanel (panelPrimeroForma3);
            calcular=addButtonToPanel(panelPrimeroForma31, "Calcular", "Calcular", this);
            labelError=addLabelToPanel (panelPrimeroForma3, "¡Error!: Debe introducir un
'Nº de colección' entero, entre 1 y 100");
            labelError.setAlignment(Label.CENTER);
         labelError.setForeground(Color.red);
         labelError.setVisible(false);

               panelSegundo=this.createPanel(this);
               panelSegundo.setLayout(new GridLayout(1,2));

               panelSegundoPosicion1=addPanelToPanel (panelSegundo);
               panelSegundoPosicion1Forma1=addPanelToPanel (panelSegundoPosicion1);

        canvasHistograma=addCanvasHistogramaToPanel(panelSegundoPosicion1Forma1,400,400);

               panelSegundoPosicion2=addPanelToPanel (panelSegundo);
               panelSegundoPosicion2.setLayout(new GridLayout(3,1));
               panelSegundoPosicion2Forma1=addPanelToPanel(panelSegundoPosicion2);
               labelMediana =addLabelToPanel (panelSegundoPosicion2Forma1, "Mediana:");
               textfiedlMediana=addTextFieldToPanel(panelSegundoPosicion2Forma1,3);
               textfiedlMediana.setEditable(false);

               panelSegundoPosicion2Forma2=addPanelToPanel(panelSegundoPosicion2);
               labelMedia =addLabelToPanel (panelSegundoPosicion2Forma2, "Media:");
               textfiedlMedia=addTextFieldToPanel(panelSegundoPosicion2Forma2,3);
               textfiedlMedia.setEditable(false);
```

```
                panelSegundoPosicion2Forma3=addPanelToPanel(panelSegundoPosicion2);
                labelDesviacion =addLabelToPanel (panelSegundoPosicion2Forma3,
"Desviación:");
                textfiedlDesviacion=addTextFieldToPanel(panelSegundoPosicion2Forma3,3);
                textfiedlDesviacion.setEditable(false);



        }
        public void actionPerformed(ActionEvent event) {

           int [] coleccion= new int [max_coleccion];

           int contadorColeccion=0;
           int contadorCompra=0;
           int compra=-1;

          if (event.getActionCommand().equals("Calcular")) {
               try {
                   String listaColeccion=listColeccion.getSelectedItem();
             int numColeccion=Integer.valueOf(listaColeccion).intValue(); //Nº cromos de
la colección
             String textNumeroColecciones = textfieldNumeroColecciones.getText();
             int numColecciones = Integer.valueOf(textNumeroColecciones).intValue();
              if ((numColecciones>=1)&&(numColecciones<=100)){
                   for (int i=1;i<=numColecciones;i++){ //iteraciones que se harán en
para cada colección de cromos
                     for (int k=0; k<10; k++) coleccion[k]=-1;
                     contadorCompra=0;
                   contadorColeccion=0;
                  while (numColeccion!=contadorColeccion){ //bucle, compra de cromos
hasta completar una colección
                        compra=aleatorio(numColeccion);
                        contadorCompra++;
                        int j=0;
                        while ((coleccion[j]!=compra)&&(j<contadorColeccion)){j++;}
//buscar si ya se ha se ha insertado el mismo número previamente
                      if (coleccion[j]!=compra) {//si el número no lo habíamos
comprado, se inserta en el vector 'coleccion'
                         coleccion[j]=compra;
                         contadorColeccion++;
                      }
                  }
                  totalColecciones[contadorTotalColecciones]=contadorCompra;
                  contadorTotalColecciones++;

               }
                 ordenarVector(contadorTotalColecciones,totalColecciones);
                 int minimo=totalColecciones[0];
                 int maximo=totalColecciones[contadorTotalColecciones-1];
                 double nmedia=media(totalColecciones,contadorTotalColecciones);
                 double nmediana=mediana(totalColecciones,contadorTotalColecciones);
                 double
ndesviacion=desviacion(totalColecciones,contadorTotalColecciones);

                 textfiedlMedia.setText(""+nmedia);
             textfiedlMediana.setText(""+nmediana);
             textfiedlDesviacion.setText(""+ndesviacion);
             grafico (totalColecciones,contadorTotalColecciones,maximo,minimo);
            }
            else labelError.setVisible(true);


             }catch (NumberFormatException e){ labelError.setVisible(true);}
          }
     }
        public void start()
    {
        // provide any code requred to run each time
        // web page is visited
    }

    /**
     * Called by the browser or applet viewer to inform this JApplet that
     * it should stop its execution. It is called when the Web page that
     * contains this JApplet has been replaced by another page, and also
     * just before the JApplet is to be destroyed.
```

```
     */
    public void stop()
    {
        // provide any code that needs to be run when page
        // is replaced by another page or before JApplet is destroyed
    }

    public void destroy()
    {
        // provide code to be run when JApplet is about to be destroyed.
    }
    public Panel createPanel (Applet applet) {
        Panel panel = new Panel();
        applet.add(panel);
        return panel;
    }
    public Panel addPanelToPanel (Panel panelp) {
        Panel panel = new Panel();
        panelp.add(panel);
        return panel;
    }
    public Label addLabelToPanel(Panel panel, String text) {
       Label label = new Label(text);
       panel.add(label);
       return label;
    }
    public List addListIntegerToPanel(Panel panel, int fil_vis, int fil_in, int ver,
ActionListener listener ) {
        int i;
        List list = new List (fil_vis, false);

        for (i=3; i<=fil_in; i++){
            list.add(""+(i), i);
        }
       list.makeVisible (ver-3);
       list.select (ver-3);
       list.addActionListener(listener);
       panel.add(list);
       return list;
    }
    public Button addButtonToPanel(Panel panel, String text, String actionCommand,
ActionListener listener) {
        Button button = new Button(text);
        button.setActionCommand(actionCommand);
        button.addActionListener(listener);
        panel.add(button);
        return button;
    }
    public TextField addTextFieldToPanel(Panel panel, int npositions) {
        TextField textfield = new TextField(npositions);
        panel.add(textfield);
        return textfield;
    }
    public TextArea addTextAreaToPanel(Panel panel, int nfil,int ncol) {
        TextArea textarea = new TextArea(nfil,ncol);
        panel.add(textarea);
        return textarea;
    }
    public int aleatorio (int grado)
        {
            int a =(int) (aux.nextDouble()*grado);
            return a+1;
    }
    public void ordenarVector (int maxn,int [] vector){
        int min,imin;
        for (int i=0; i< maxn-1; i++){
            min=vector[i];
            imin=i;
            for (int j=i+1;j<maxn; j++){
                if (vector[j]<min){
                    min=vector[j];
                    imin=j;
                }
            }
            vector[imin]=vector[i];
            vector[i]=min;
        }
```

```
    }

    public double esperanza (int nColeccion,int nCromos){
        double sum=0;
        for (int i=1;i<=nColeccion;i++){
                sum=sum+(1.0/i);
        }
        return (sum*nColeccion)/nCromos;
    }
    public CanvasHisto addCanvasHistogramaToPanel(Panel panel, int sizex, int sizey) {
         CanvasHisto canvas = new CanvasHisto();
         canvas.setSize(sizex,sizey);
         canvas.setBackground (Color.white);
         panel.add(canvas);
         return canvas;
    }
        public double media(int[] array,int max){

                double media = 0.0;
                for (int i=0; i < max; i++) {
                  media = media + array[i];
                }
                media = media / max;
                return media;
        }
        public double desviacion(int[] array,int max){

                double sum=0.0;
                double nmedia = media(array,max);
                for (int i=0; i < max; i++) {
                        double dif=array[i]-nmedia;
                  sum=sum+Math.pow(dif,2);
                }
                sum=sum/(max-1);
                return Math.sqrt(sum);
        }
        public double mediana(int[] array,int max) {//mediana de un vector ordenado
            int middle = max/2;
            if (max == 1) {
                return array[middle];
            } else {
               return (array[middle-1] + array[middle]) / 2.0;
            }

        }
    public void grafico (int[] array,int max,int maximo,int minimo){
        int [] vectorGrafico=new int[(maximo/5)];//se agrupan de 5 en cinco las
colecciones, por lo tanto el tamaño del vector también se reducira

        for(int i=0;i<max;i++){
                int aux=(array [i]-minimo)/5;
                vectorGrafico[aux]++;
        }
        canvasHistograma.dibujarHistograma (vectorGrafico, 1,minimo);

    }

}

class Main_cromos_2 {

    private static cromos_2 myApplet = new cromos_2();

    // Programa principal (permite standalone)
    public static void main (String[] args) {
        Frame myFrame = new Frame("Cromos para varias colecciones");
        // Acciones al cerrar la aplicacion
        myFrame.addWindowListener(new WindowAdapter() {
            // Redefine el metodo windowClosing
            public void windowClosing(WindowEvent event) {
                myApplet.stop();
                myApplet.destroy();
                System.exit(0);
            }
        });
        myFrame.add(myApplet);//, BorderLayour.CENTER);
        myFrame.setSize(1000,600);
```

```
        myFrame.setLocation(100,50);
        myApplet.init();
        myApplet.start();
        myFrame.setVisible(true);
    }

}
```

8.4. Código de la clase 'CanvasHisto'

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

class CanvasHisto extends Canvas {

    private Graphics g;
    // No importa que sea de longitud 0, se carga en dibujarHistograma
    private int[] tab = new int[0];
    private int scale;
    private int iescala; // Inico de la las modalidades


    public void paint(Graphics g) {
        int offsetx=100;
        int offsety= 150;
        int width=30;
        int auxmod; //auxiliar para modificar las modalidades
        int imaxh = tab.length/2;
        String mod;
        g.setColor (Color.white);
        g.fillRect (0,0, 400, 400);

        for (int i=0; i<tab.length; i++) {
            auxmod=iescala+(i*5);
            mod = ""+auxmod;
            g.setColor(Color.blue);
            g.fillRect(offsetx+i*width,offsety+(tab[imaxh]-
tab[i])*scale,width,tab[i]*scale);
            g.setColor(Color.black);
            g.drawRect(offsetx+i*width,offsety+(tab[imaxh]-
tab[i])*scale,width,tab[i]*scale);
            g.drawString (mod, offsetx+i*width+3,offsety+40);
        }
    }

    public void blanco(Graphics g) {
        g.setColor (Color.white);
        g.fillRect (0,0, 400, 400);
    }

    public void limpiarHist (){
    g=this.getGraphics();
    this.blanco (g);
}

    public void dibujarHistograma(int[] t, int s, int a) {
    tab=t;
    scale =s;
    iescala = a;
    g = this.getGraphics();
    this.paint(g);
    }
}
```