CAPÍTOL 4. LA NOSTRA SOLUCIÓ

4.1. Introducció

El projecte es centra en oferir una solució software que permeti establir comunicació entre una aplicació J2ME(MIDlet) instal·lada en un telèfon mòbil, actualment el dispositiu mòbil per excel·lència, i una xarxa domòtica per tal de poder comprovar i modificar, de forma remota, l'estat dels diferents dispositius connectats a ella.

Partim d'una xarxa domòtica controlada per un ordinador central connectat a ella, que anomenarem servidor. El servidor desenvoluparà la funció de comprovació/modificació de l'estat dels diferents dispositius domòtics. També disposarà d'una connexió a Internet i permetrà la comunicació entre ell i un telèfon mòbil mitjançant el software que em desenvolupat.

En quan a control remot via telèfon mòbil, les ofertes que trobem al mercat actual es basen en les tecnologies com WAP, SMS, reconeixement de veu o introducció de codis a través del teclat. Amb aquet projecte oferim el control remot a través d'una aplicació desenvolupada amb J2ME que utilitza el protocol HTTP per comunicar-se amb el servidor de la llar.

El paquet d'aplicacions desenvolupades han estat programades amb la tecnologia JAVA. Aquesta decissió s'ha près en base a varies raons:

- L'aplicació del telèfon mòbil tenia de ser en J2ME.
- JAVA ens assegura la portabilitat de les seves aplicacions en diferents sistemes operatius.
- Interès personal d'aprendre a programar en un llenguatge del qual en coneixia molt poca cosa. Només n'havia fet un ús bàsic en alguna de les assignatures del primer curs de la carrera.

4.1.1. La Xarxa domòtica

La xarxa domòtica estarà formada per un seguit de dispositius que permetin la seva monitorització i control com per exemple: persianes, portes, finestres, llums... Estarà contro-

lada pel servidor, que gestionarà l'estat de tots els dispositius domòtics i estarà permanentment connectada a Internet. Permetrà que des del telèfon mòbil, a partir del nostre software, puguem controlar remotament els diferents dispositius de la llar.

Suposarem que la xarxa ja està muntada i funcionant correctament, sense importar-nos el tipus de protocol domòtic utilitzat(KNX, LonWorks...). Aquesta part la simularem amb el simulador de l'aplicació Casa Virtual, que estarà instal·lada al servidor. Amb aquest software, prèviament haurem creat la casa domòtica que vulguem simular.

4.1.2. Servidor

El servidor serà un ordinador que, per una banda, estarà connectat a la xarxa domòtica realitzant la funció de controlador central i, per l'altre, estarà connectat a Internet per permetre'n el control remot mitjançant un telèfon mòbil.

Dit d'una manera més precisa, disposarà d'una aplicació servlet que es comunicarà, per un cantó, amb la xarxa domòtica i, per l'altre, amb el telèfon mòbil via Internet, permetent d'aquesta manera, que és puguin controlar remotament els dispositius de la llar.

Utilitzarem Linux com a sistema operatiu i Apache Tomcat com a servidor d'aplicacions per a poder executar el servlet.

4.1.3. Telèfon mòbil

Serà el terminal mòbil que utilitzarem per controlar la llar domòtica de manera remota una vegada connectem amb el servidor. El software que l'hi permet comunicar-se amb el servidor ha estat programat amb J2ME.

4.1.4. Xarxa domòtica + Servidor + Telèfon mòbil

Resumint:

- Aplicació Casa Virtual: Instal·lada al servidor. Realitza dos funcions:
    1. Permetre la creació de diferents llars domòtiques amb els seus corresponents dispositius controlables.

2. Simular el funcionament de qualsevol de les llars domòtiques creades anteriorment i permetre'n el control remot.

- Servlet: Instal·lat al servidor. Permet la comunicació entre el telèfon mòbil i la llar domòtica simulada. Realitza la funció de "pont" entre el MIDlet i la casa domòtica. Utilitzem Internet per la comunicació Servlet-MIDlet.

- MIDlet: Instal·lat al telèfon mòbil. Permet controlar les diferents llars domòtiques que tinguem disponibles. Per a poder accedir-hi, primer de tot establirem comunicació amb el servidor i aquest farà arribar les nostres ordres al simulador de la llar.

4.2. Especificació de la nostra llar

4.2.1. Introducció

Les llars que podrem controlar podran ser des de pisos amb una única habitació, fins a grans cases de varies plantes. Estaran dividides en diferents sectors que els identificarem amb dos paràmetres: (PlantaX,HabitacióY).

Una vegada tinguem la llar dissenyada amb tots els seus sectors, ja podrem col·locar-hi els diferents aparells domòtics que tinguem instal·lats a la casa compatibles amb les nostres aplicacions. Per tant, cada dispositiu serà ubicat en un dels sectors creats anteriorment. Cada sector podrà incloure diferents dispositius del mateix o diferent tipus. N'hem implementat un total de 17 tipus:

- Alarma
- Calefacció
- Climatització
- Detector de fum
- Finestra
- Fogons
- Forn
- Llum

- Microones
- Nevera
- PC
- Persiana
- Porta
- Rentaplats
- TV
- Escalfador d'aigua elèctric
- Vídeo

4.2.2. Paquet *llistaobjectes*

Descripció informal: Paquet que està format únicament per una sola classe anomenada llista.

4.2.3. Classe *llista*

Descripció informal: Aquesta classe forma part del paquet *llistaobjectes* com a única classe. Ens permet crear, modificar i eliminar llistes de qualsevol tipus d'objecte. Com veurem més endavant, es utilitzada per les classes: *Cases, Casa, Planta i Hab*(veure figura 4.1).

Atributs:

- list: Conté un array d'objectes.
- intLastElement: Conté el nombre d'objectes inserits a l'array *list*.
- INITIAL_SIZE: Indica el tamany inicial que tindrà l'array *list* quan el creem.

Mètodes:

- Conjunt de mètodes necessaris per crear un objecte llista i consultar/modificar-ne els seus atributs.

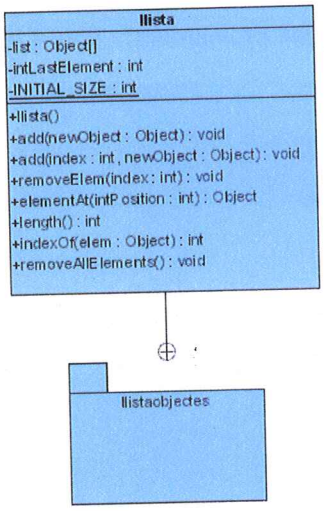

Figura 4.1: Classe *llista*

4.2.4. Paquet *Casa*

Descripció informal:

El paquet Casa conté la implementació de les llars domòtiques que podrem crear que més tard controlarem remotament a través del telèfon mòbil. Més concretament, està format per un total de 22 classes i una interfície.

El tipus de casa que ens permet construir estarà dividida en plantes i cadascuna d'aquestes, en habitacions. Les plantes hauran de contenir una habitació com a mínim i les habitacions podran incloure o no dispositius domòtics.

Totes les classes que formen part d'aquest paquet implementaran la interfície(peu de pàgina: que es una interfície) Serializable que pertany al paquet java.io de la llibreria estàndard de Java. Això ens permetrà desar i recuperar qualsevol classe mitjançant una seqüència d'objectes. Dit d'una altre manera, ens permet la serialització d'objectes. No conté cap mètode, per tant, no ens farà falta modificar el codi de les classes que la implementin.

4.2.5. Interfície *meuPersistent*

Descripció informal: Per al nostre sistema de comunicació entre el telèfon mòbil i la casa domòtica necessitarem aquesta interfície. Les classes que la implementin disposaran de mètodes per tractar la informació que vulguem enviar o la que haguem rebut(que també

haurà estat tractada per ser enviada prèviament).(veure figures 4.2 i 4.3).

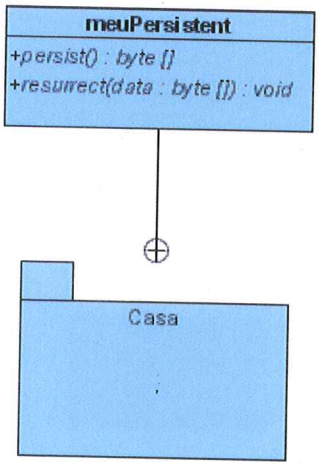

Figura 4.2: Classe *meuPersistent*

Mètodes:

- persist():byte[] : Totes les dades que vulguem transmetre primer seran tractades per aquesta funció. Com a resultat ens retornarà un array de tipus byte que després podrà ser enviat.
- resurrect(data:byte[]):void: Acció inversa a la persist():byte[]. Una vegada haguem rebut informació provinent de la casa domòtica o del telèfon mòbil a través d'Internet, la transformarem al seu estat inicial. O sigui, l'array de bytes que rebrem, el transformarem al tipus d'objecte que era abans de ser tractat i enviat.

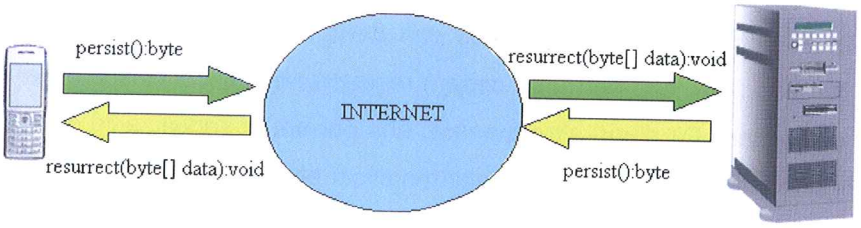

Figura 4.3: *Esquema dels mètodes persist i resurrect de la interfície meuPersistent*

4.2.6. Classe Aparell

Descripció informal: De la classe aparell en pengen tots els tipus d'aparells/dispositius que podrem instal·lar i controlar a la nostra llar.

Tots els aparells heretaran els seus atributs i mètodes. Implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.4).

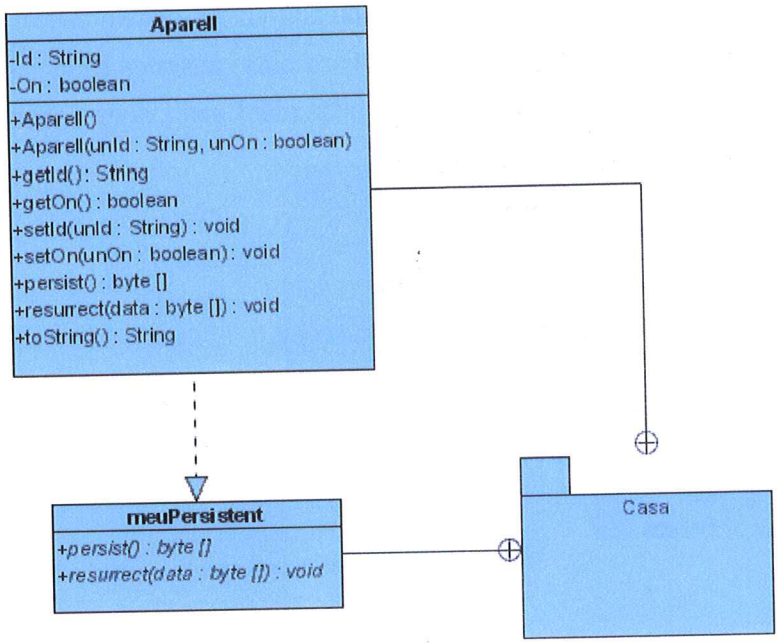

Figura 4.4: Classe *Aparell*

Atributs:

- Id: Permet emmagatzemar el nom de l'aparell.
- On: Permet emmagatzemar l'estat de l'aparell: engegat/parat (True/False)

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Aparell en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Aparell que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Aparell.

4.2.7. Classe *Alarma*

Descripció informal: Classe que implementa una alarma. La podem activar i desactivar mitjançant una contrasenya de 4 dígits. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.5).

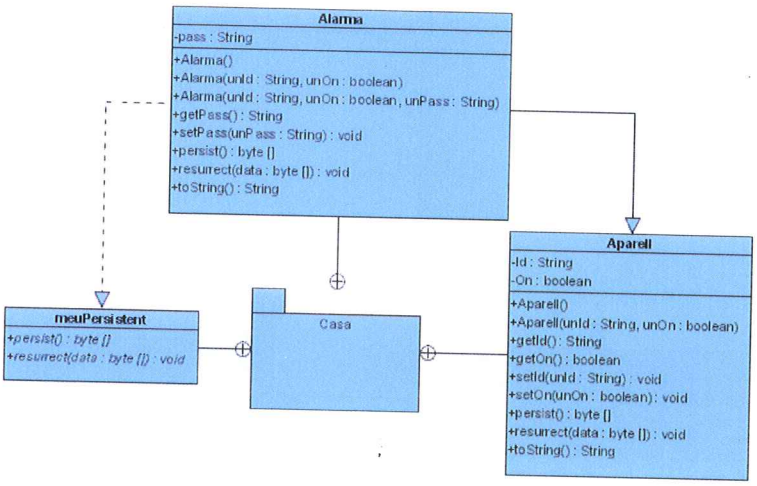

Figura 4.5: Classe *Alarma*

Atributs:

- pass: Permet emmagatzemar la contrasenya d'activació/desactivació de l'alarma.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Alarma i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir un objecte tipus Alarma en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Alarma que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Alarma.

4.2.8. Clase *Calefacció*

Descripció informal: Classe que ens permet controlar el termòstat de la calefacció. Podem especificar la temperatura que volem i també el podem programar. Per programar-lo li indicarem la data i hora en que s'iniciarà i aturarà. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.6).

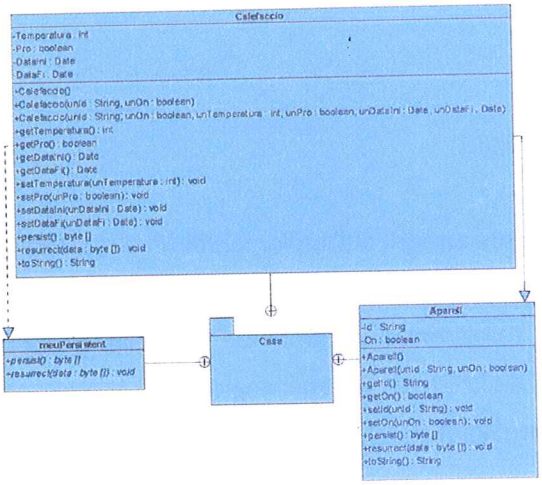

Figura 4.6: Classe *Calefacció*

Atributs:

- Temperatura: Permet emmagatzemar la temperatura que volem fixar al termòstat de la calefacció
- Pro: Permet emmagatzemar si està programada temporalment la calefacció.
- DataIni: Permet emmagatzemar la data i hora d'inici del període de funcionament programat de la calefacció.
- DataFi: Permet emmagatzemar la data i hora de finalització del període de funcionament programat de la calefacció.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Calefacció i consultar i modificar-ne els seus atributs.

- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Calefacció en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Calefacció que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Calefacció.

4.2.9. Classe *Climatització*

Descripció informal: Classe encarregada d'implementar un aire condicionat o bomba de calor. Podem marcar la temperatura que desitgem i també programar tal i com ho fem amb la classe Calefacció. De fet, té els mateixos atributs i mètodes que la classe calefacció. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.7).

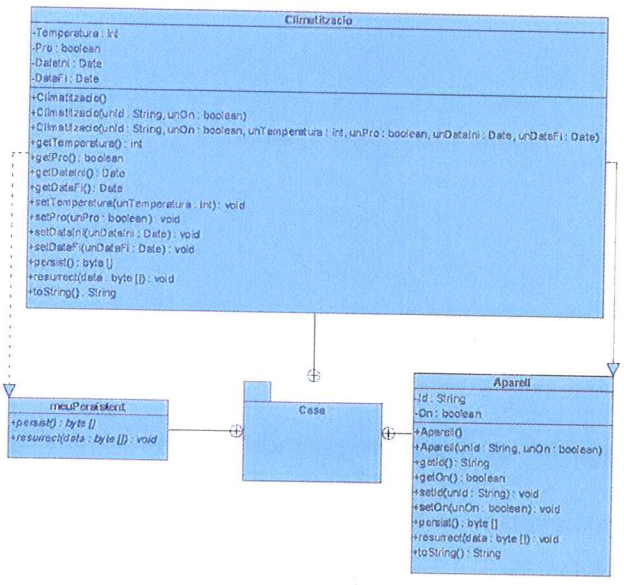

Figura 4.7: Classe *Clima*

4.2.10. Classe *DetFum*

Descripció informal: Classe que s'encarrega d'implementar un dispositiu de detecció de

fum. El podem activar i desactivar. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.8).

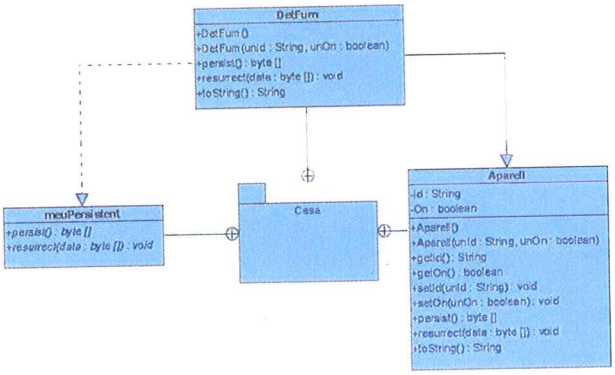

Figura 4.8: Classe *DetFum*

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus DetFum i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte DetFum en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus DetFum que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte DetFum.

4.2.11. Classe *Finestra*

Descripció informal: Classe amb la que controlem l'obertura d'una finestra. Ens indica si està oberta o tancada i ens permet canviar-ne l'estat, o sigui, obrir-la o tancar-la. Hereta

els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.9).

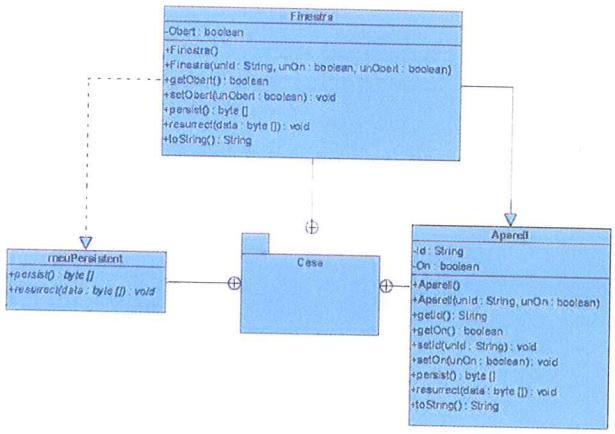

Figura 4.9: Classe *Finestra*

Atributs:

- Obert: Permet emmagatzemar l'estat de la finestra: oberta/tancada.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Finestra i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Finestra en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Finestra que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Finestra.

4.2.12. Classe *Fogons*

Descripció informal: Classe encarregada d'implementar un grup de fogons de cuina, Per defecte crea un grup de 4 fogons, un de petit, mitjà, mitjà-gran i gran i ens permet veure i regular la potència de cadascun. També podem crear grups de més o menys nombre de fogons. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.10).

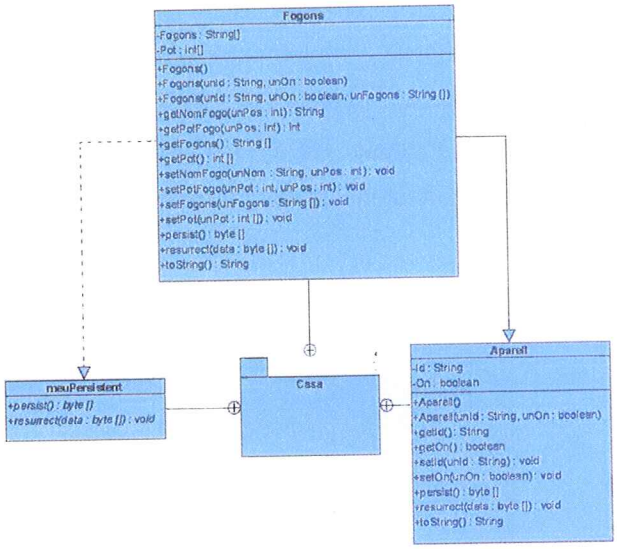

Figura 4.10: Classe *Fogons*

Atributs:

- Fogons: Permet emmagatzemar el nom dels diferents fogons que pertanyen a un mateix grup.
- Pot: Permet emmagatzemar la potència en que es troba cada fogó en el moment actual.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Fogons i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Fogons en un array de bytes.

- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Fofons que prèviament ha estat transformat i encapsulat a un array de bytes.

- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Fogons.

4.2.13. Classe *Forn*

Descripció informal: Classe encarregada d'implementar un forn convencional de cuina. Podem controlar-ne la resistència superior, inferior i el grill. També el podem programar, proporcionant-li l'hora de començar i finalitzar. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.11).

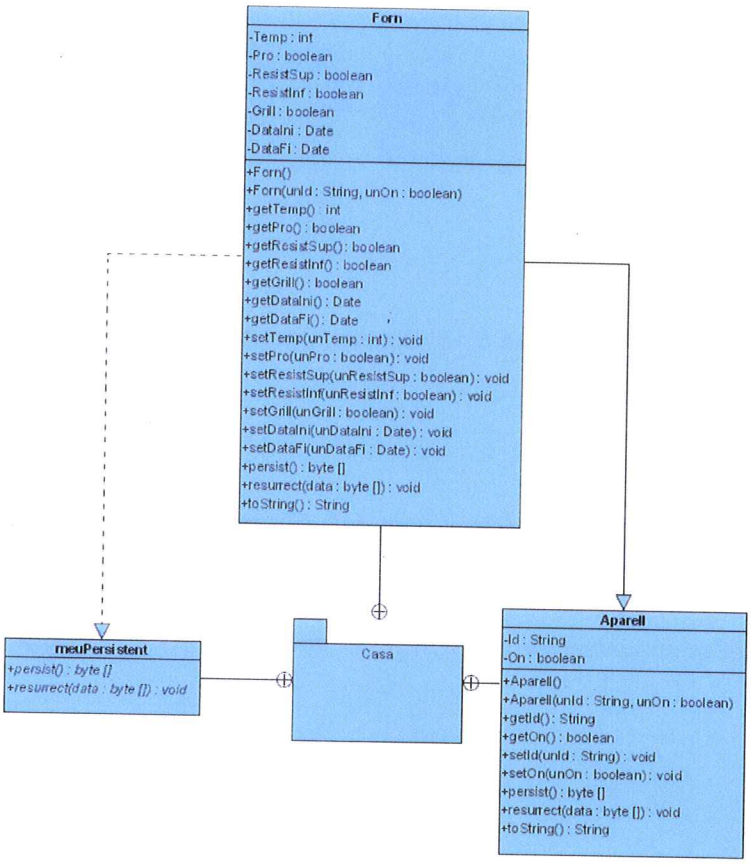

Figura 4.11: Classe *Forn*

Atributs:

- Temp: Permet emmagatzemar la temperatura a la que volem que treballi el forn.
- Pro: Ens indica si el forn es troba programat o no.
- ResistSup: Ens indica si la resistència superior del forn està funcionant.
- ResistInf: Ens indica si la resistència inferior del forn està funcionant o no.
- Grill:Ens indica si el grill està funcionant.
- DataIni: Permet emmagatzemar la data i hora d'inici del període de funcionament programat del forn
- DataFi: Permet emmagatzemar la data i hora de finalització del període de funcionament programat del forn.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Forn i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Forn en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Finestra que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Forn.

4.2.14. Classe *Llum*

Descripció informal: Classe que implementa un llum (bombeta...) que el podrem engegar, apagar i regular-ne la potència lumínica. Aquesta última la controlarem indicant-li el tant per cent que desitgem: 100% màxima potència i 0% potència nul·la, per tant llum apagat. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.12).

Atributs:

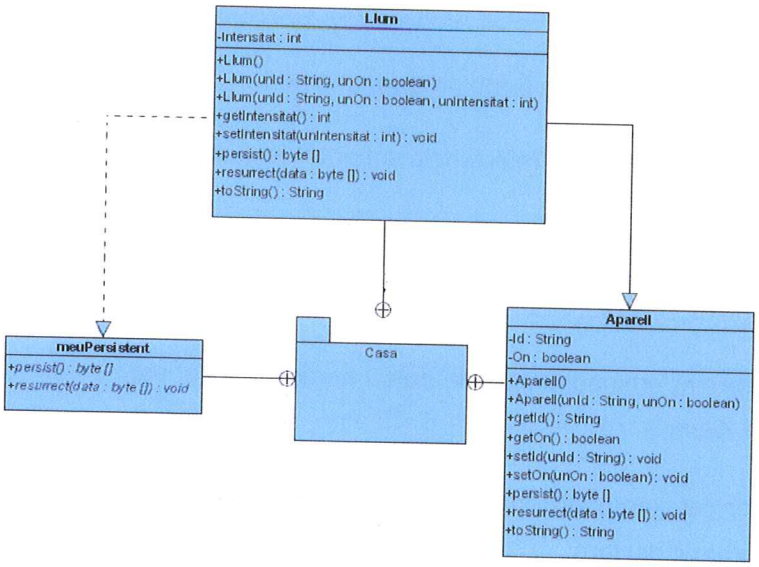

Figura 4.12: Classe *Llum*

- Intensitat: Emmagatzema la intensitat lumínica que emet el llum en aquest moment.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Llum i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Llum en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Llum que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Llum.

4.2.15. Classe *Microones*

Descripció informal: Classe que com el seu nom indica, implementa l'electrodomèstic microones. En controlarem la potència i grill. El podrem programar perquè realitzi la funció triada durant l'interval de temps que li ordenem. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.13).

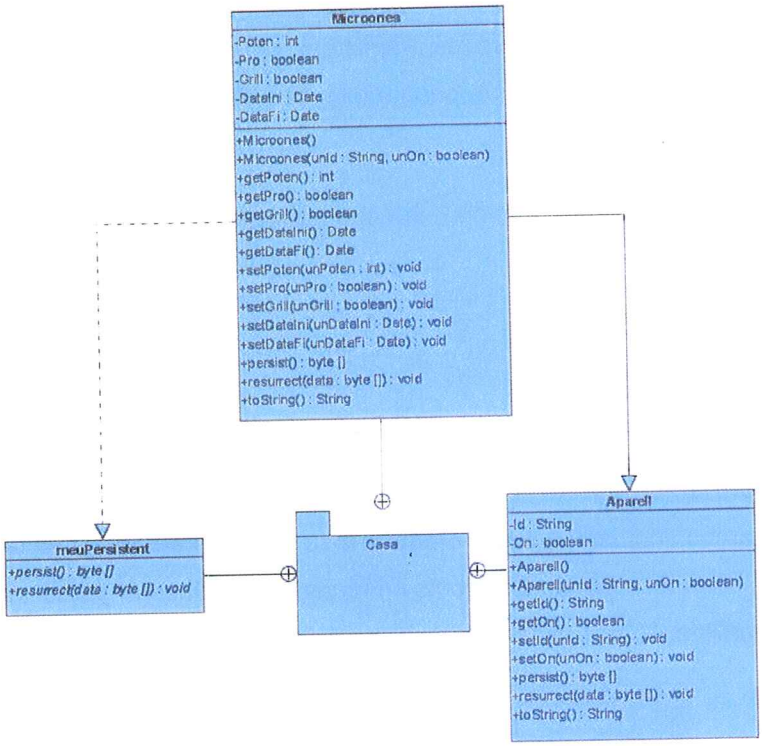

Figura 4.13: Classe *Microones*

Atributs:

- Poten: Permet emmagatzemar la potència a la que volem que treballi el microones.
- Pro: Ens indica si el microones es troba programat o no.
- Grill: Ens indica si el grill està funcionant.
- DataIni: Permet emmagatzemar la data i hora d'inici del període de funcionament programat del microones.
- DataFi: Permet emmagatzemar la data i hora de finalització del període de funcionament programat del microones.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Microones i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Microones en un array de bytes.

- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Microones que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Microones.

4.2.16. Classe *Nevera*

Descripció informal: La classe nevera implementa una nevera convencional o tipus Combi, o sigui que també tingui congelador. Podrem fixar la temperatura tant de la nevera com del congelador, si en té. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.14).

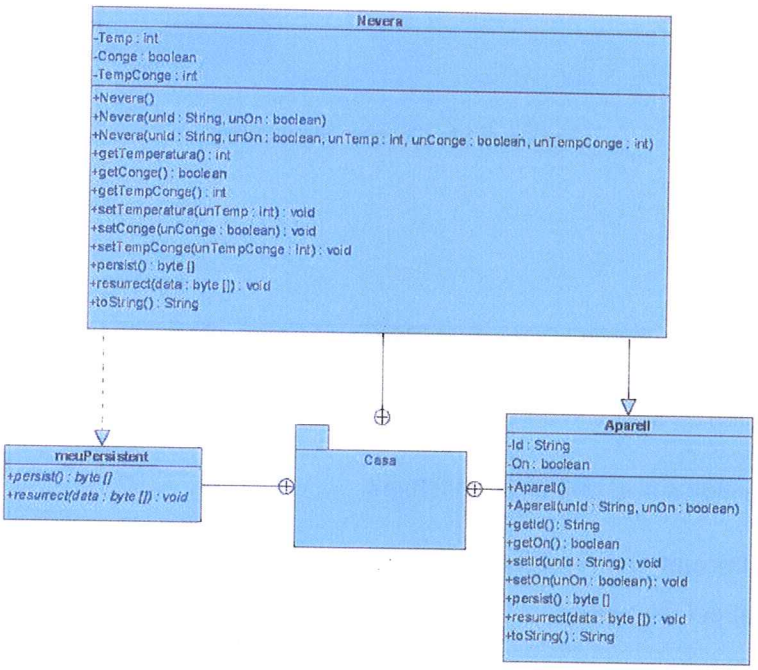

Figura 4.14: Classe *Nevera*

Atributs:

- Temp: Permet emmagatzemar la temperatura que volem que mantingui la nevera.
- Conge: Ens indica si la nevera també té congelador.
- TempConge: Permet emmagatzemar la temperatura que volem que mantingui el congelador, en el cas que la nevera en disposi d'un.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Nevera i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Nevera en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Nevera que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Nevera.

4.2.17. Classe *PC*

Descripció informal: Classe que implementa un ordinador personal. Únicament ens permet engegar i parar-lo. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.15).

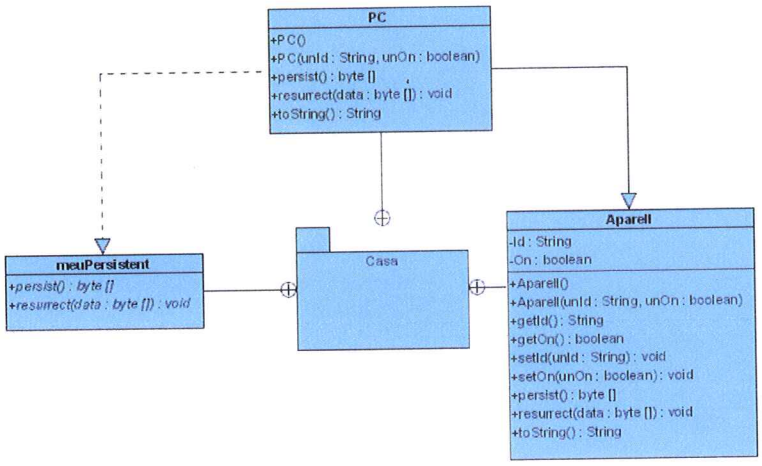

Figura 4.15: Classe *PC*

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus PC.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte PC en un array de bytes.

- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus PC que prèviament ha estat transformat i encapsulat a un array de bytes.

- toString: Mètode que proporciona una cadena que representa el valor de l'objecte PC.

4.2.18. Classe *Persiana*

Descripció informal: Classe encarregada de consultar i modificar el nivell d'obertura d'una persiana. El nivell al 0% indica que es troba totalment abaixada i el 100% completament alçada. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.16).

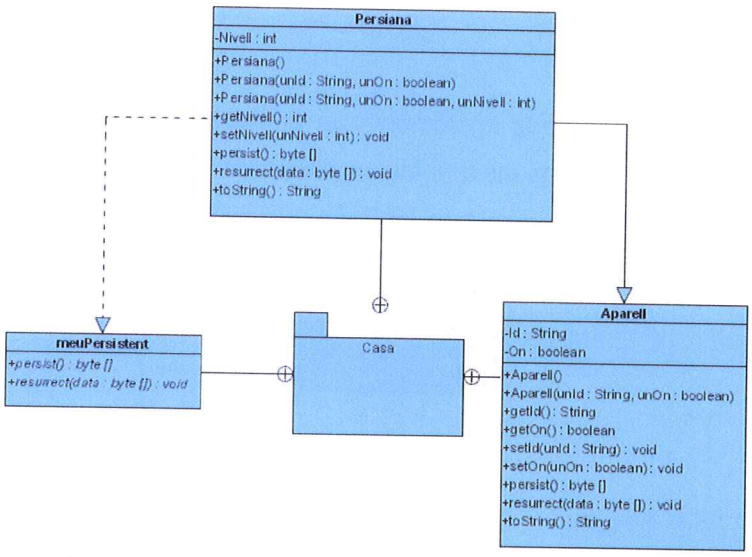

Figura 4.16: Classe *Persiana*

Atributs:

- Nivell: Ens indica el nivell d'obertura actual d'una persiana en tant per cent.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Persiana i consultar i modificar-ne els seus atributs.

- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Persiana en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Persiana que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Persiana.

4.2.19. Classe *Porta*

Descripció informal: Classe que ens permet obrir i tancar una porta. Només té dos posicions: totalment oberta o completament tancada. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.17).

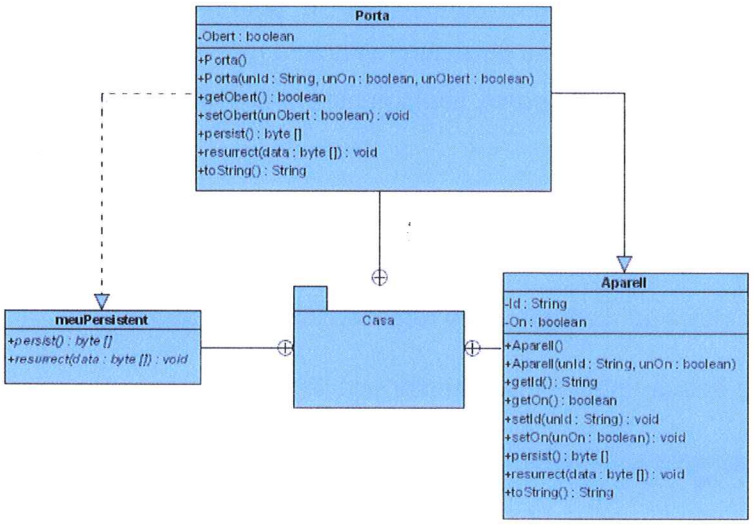

Figura 4.17: Classe *Porta*

Atributs:

- Obert: Ens indica si la porta està oberta o tancada.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Porta i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Porta en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Porta que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Porta.

4.2.20. Classe *Rentaplats*

Descripció informal: Classe que implementa un rentaplats amb temperatura graduable de l'aigua i una llista de programes de rentat. Per defecte ens crea un rentaplats amb una llista de programes i unes temperatures per defecte, però també li podem indicar quins tipus de programes i temperatures té de tenir el nostre rentaplats en qüestió. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.18).

Atributs:

- Prog: Ens indica quin programa de rentat tenim seleccionat.
- LlistaProg: Ens permet emmagatzemar una llista de diferents programes de rentat.
- Temp: Ens indica a quina temperatura volem l'aigua per al rentat.
- LlistaTemp: Ens permet emmagatzemar una llista de diferents temperatures de l'aigua que podrem fer servir amb el diferents tipus de rentats que permeti el rentaplats.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Rentaplats i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Rentaplats en un array de bytes.

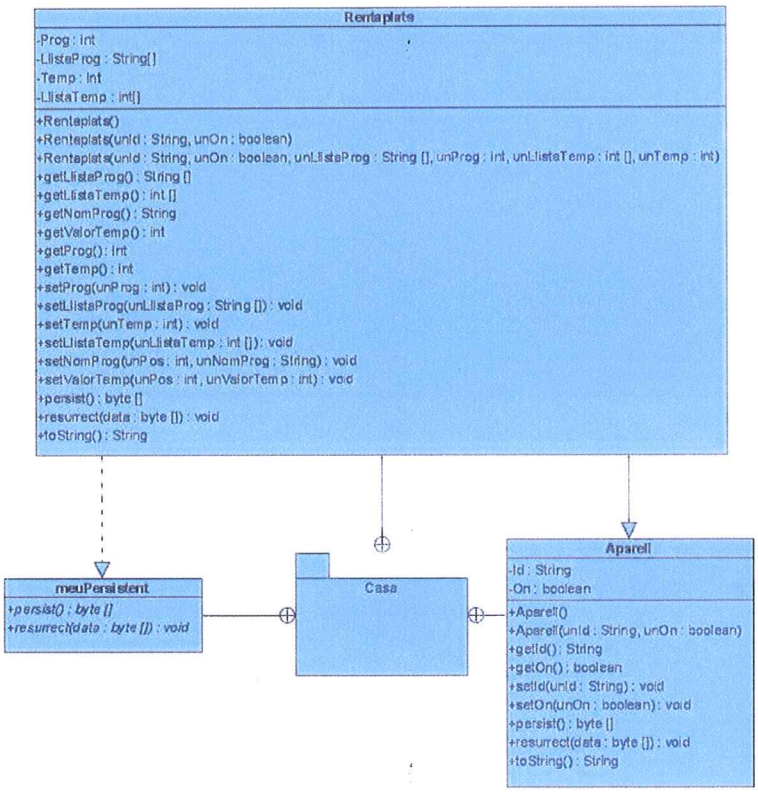

Figura 4.18: Classe *Rentaplats*

- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Rentaplats que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Rentaplats

4.2.21. Classe *TV*

Descripció informal: Classe que s'encarrega d'implementar un televisor. En podrem variar el volum de so, silenciar-lo(opció MUTE) i canviar de canal. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.19).

Atributs:

- Canal: Emmagatzema el canal que estem veient.
- Canals: Emmagatzema tots els canals que sintonitzem.

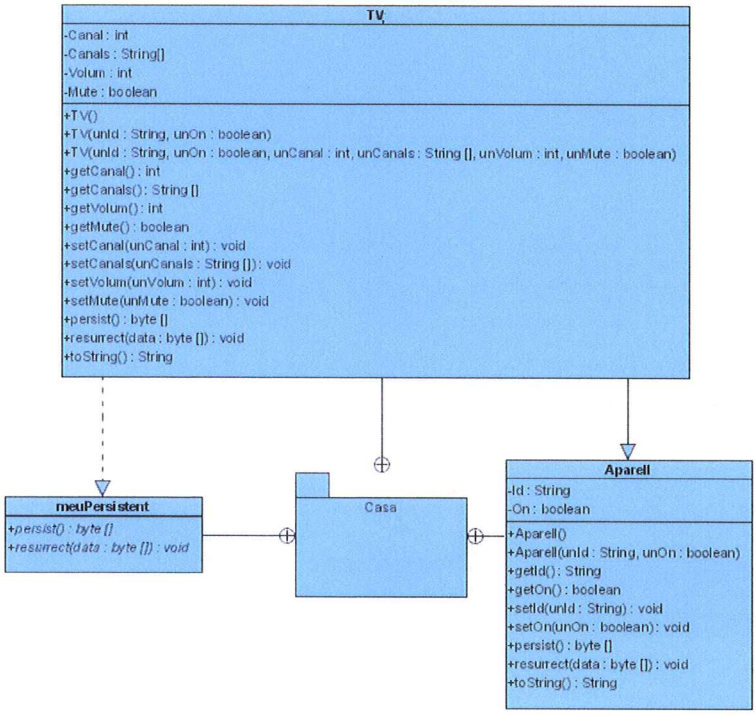

Figura 4.19: Classe *TV*

- Volum: Ens indica el volum dels altaveus del televisor en aquest moment.
- Mute: Ens indica si tenim silenciat el so del televisor.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus TV i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte TV en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus TV que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte TV.

4.2.22. Classe *TermoElectric*

Descripció informal: Aquesta classe s'encarrega de controlar un escalfador d'aigua domèstic elèctric. Únicament l'hi indiquem la temperatura que volem que escalfi l'aigua. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.20).

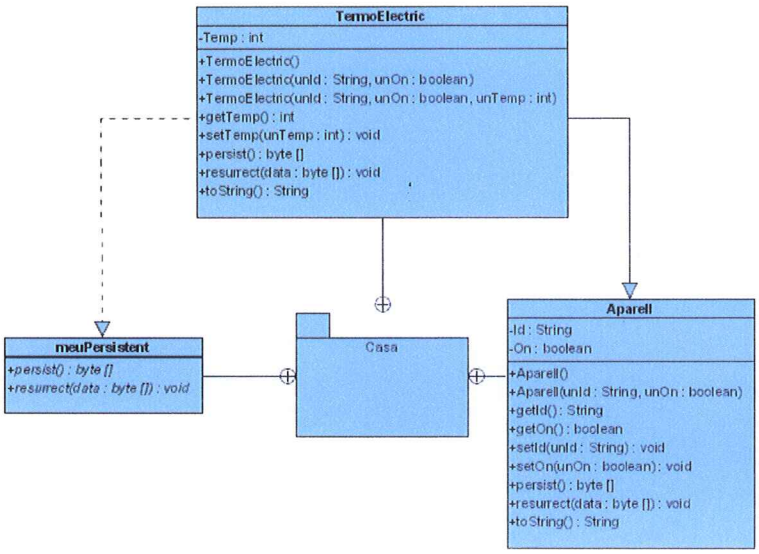

Figura 4.20: Classe *TermoElectric*

Atributs:

- Temp: Permet emmagatzemar la temperatura a la que volem que escalfi l'aigua.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus TermoElectric i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte TermoElectric en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus TermoElectric que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte TermoElectric.

4.2.23. Classe *Video*

Descripció informal: Classe que implementa un gravador de vídeo. Indicant-li el canal de televisió, podem iniciar directament la gravació en vídeo fins que l'aturem, o bé, programar-lo perquè inicii i finalitzi la gravació durant l'interval de temps que desitgem. Hereta els atributs i mètodes de la classe Aparell, implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.21).

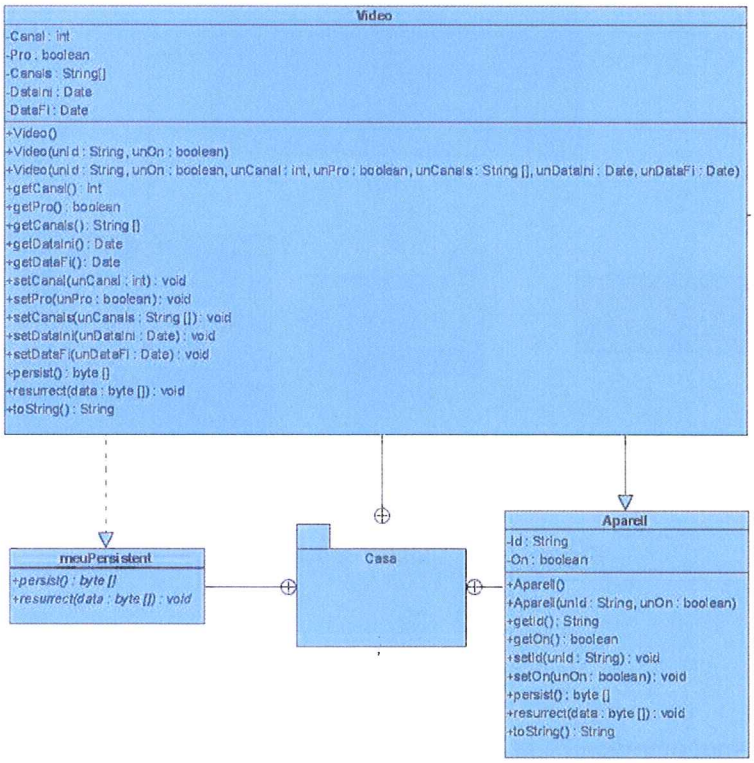

Figura 4.21: Classe *Video*

Atributs:

- Canal: Ens indica el canal que tenim seleccionat per a iniciar la gravació
- Pro: Ens indica si el vídeo està programat.
- Canals: Llista de tots els canals de televisió que tenim disponibles.
- DataIni: Permet emmagatzemar la data i hora d'inici del període gravació programat del vídeo.
- DataFi: Permet emmagatzemar la data i hora de finalització del període gravació programat del video.

Mètodes:

- Conjunt de mètodes necessaris per crear un aparell tipus Vídeo i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Video en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Video que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cádena que representa el valor de l'objecte Vídeo.

4.2.24. Classe *Hab*

Descripció informal: Cada aparell que haguem creat, el col·locarem al sector de la casa on estigui instal·lat. La llar es divideix en sectors de primer nivell: les plantes. Cada planta en sectors de segon nivell: habitacions, passadissos, terrasses.... En el cas que una casa tingui jardí, considerem que aquest també formarà part d'una de les plantes de la casa. Amb la classe Hab implementem els sectors de segon nivell. Tindran un identificador i hi col·locarem els aparells creats. Cadascun on correspongui. Implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.22).

Atributs:

- Id: Emmagatzema el nom del sector de segon nivell.
- Aparells: Emmagatzema tots els aparells que es troben instal·lats en aquest sector.

Mètodes:

- Conjunt de mètodes necessaris per crear un objecte tipus Hab i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Hab en un array de bytes.

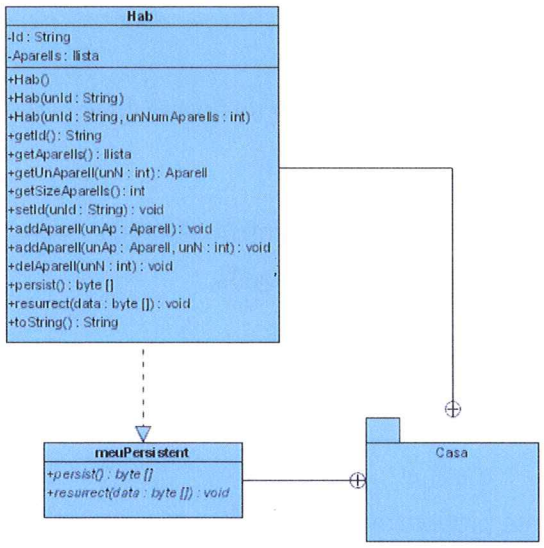

Figura 4.22: Classe *Hab*

- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Hab que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Hab.

4.2.25. Classe *Planta*

Descripció informal: Aquesta classe implementa els sectors de primer nivell (les plantes de la casa). Els diferenciarem amb l'identificador que desitgem i comptaran com a mínim amb un sector de segon nivell. Implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.23).

Atributs:

- Id: Ens permet emmagatzemar el nom del sector de primer nivell.
- Habs: Ens permet emmagatzemar tots els sectors de segon nivell que formen part de l'objecte Planta.

Mètodes:

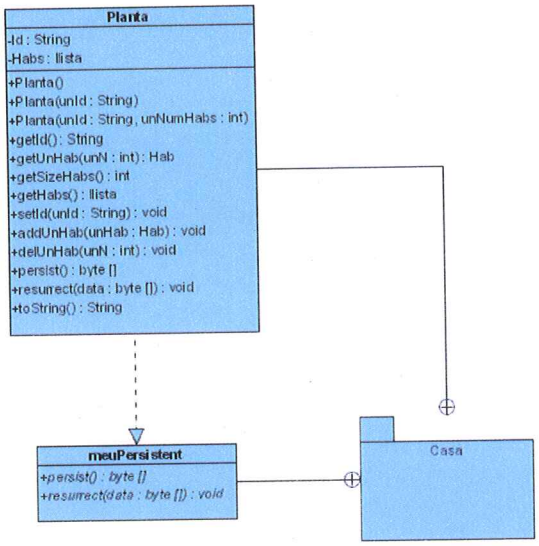

Figura 4.23: Classe *Planta*

- Conjunt de mètodes necessaris per crear un objecte tipus Planta i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Planta en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Planta que prèviament ha estat transformat i encapsulat a un array de by
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Planta.

4.2.26. Classe *Casa*

Descripció informal: Com el seu nom indica, implementarà la casa creada. Tindrà un identificador i els sectors de primer nivell(les plantes) que en formin part. Per tant comptarà amb tota la informació sobre la nostra llar: inclou els sectors de primer nivell, aquests els de segon i cadascun d'ells els aparells que hi tenen instal·lats. Implementa la interfície meuPersistent i pertany al paquet Casa (veure figura 4.24).

Atributs:

- Id: Emmagatzema el nom de la casa.

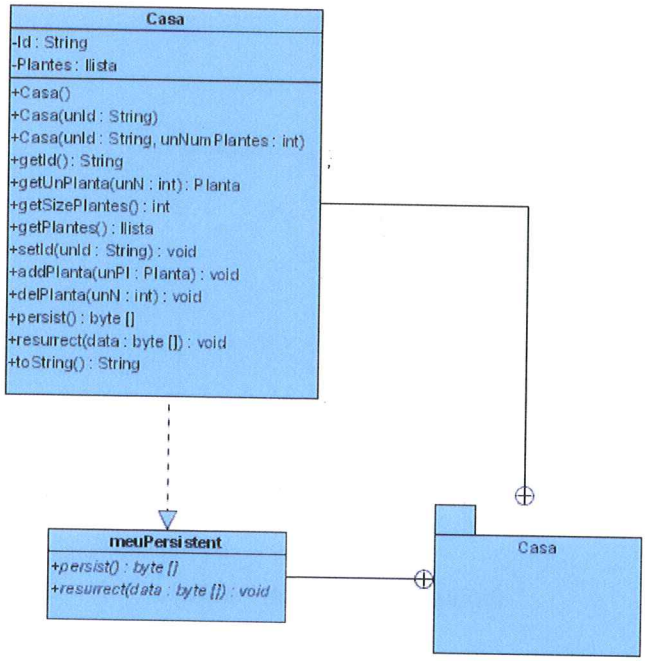

Figura 4.24: Classe *Casa*

- Plantes: Emmagatzema tots els sectors de primer nivell que té la casa.

Mètodes:

- Conjunt de mètodes necessaris per crear un objecte tipus Casa i consultar i modificar-ne els seus atributs.
- persist: pertany a la interfície meuPersistent. Mètode invers al resurrect. Permet convertir l'objecte Casa en un array de bytes.
- resurrect: pertany a la interfície meuPersistent. Mètode invers al persist. Restaurem un objecte tipus Casa que prèviament ha estat transformat i encapsulat a un array de bytes.
- toString: Mètode que proporciona una cadena que representa el valor de l'objecte Casa.

4.2.27. Classe *Cases*

Descripció informal: Aquesta classe va néixer davant la necessitat d'agrupar totes les cases creades en un únic objecte. Totes elles estaran emmagatzemades en un objecte de la classe Cases. Pertany al paquet Casa (veure figura 4.25).