IIP (E.T.S. d'Enginyeria Informàtica)
Curs 2015-2016

Pràctica 4. Desenvolupament i reutilització de classes Java

Professors d'IIP
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Índex

1 Objectius i treball previ a la sessió de pràctiques **1**

2 Descripció del problema **2**

3 Activitats de laboratori **2**

1 Objectius i treball previ a la sessió de pràctiques

L'objectiu principal d'aquesta pràctica és el disseny d'una classe tipus de dada; en concret, s'incidirà en els següents aspectes presentats en els temes 3 i 4 (*Variables: definició, tipus i usos* i *Mètodes: definició, tipus i usos en Java*, respectivament):

- Declaració de l'estructura dels objectes d'una classe.
- Declaració dels mètodes constructors de la classe, mètodes consultors i modificadors i uns altres.
- Ús d'una classe tipus de dada: declaració de variables referència, creació d'objectes i aplicació dels seus mètodes.

2 Descripció del problema

En aquesta pràctica es va a desenvolupar la classe `Hora` per a que tinga una funcionalitat com la que s'indica en la següent figura:

Constructor Summary

`Hora()`	Hora (hores i minuts) actual UTC (temps coordinat universal).
`Hora(int hh, int mm)`	Hora corresponent a les hh hores i mm minuts.

Method Summary

`int`	`aMinuts()` Retorna el nombre de minuts transcorreguts des de les 00:00 fins l'Hora.
`int`	`compareTo(Hora altraHora)` Compara cronologicament l'Hora i altraHora.
`boolean`	`equals(java.lang.Object o)` Retorna true sii o es una Hora que coincideix en hores i minuts amb l'Hora.
`int`	`getHores()` Retorna les hores de l'Hora.
`int`	`getMin()` Retorna els minuts de l'Hora.
`void`	`setHores(int hh)` Modifica les hores de l'Hora.
`void`	`setMin(int mm)` Modifica els minuts de l'Hora.
`java.lang.String`	`toString()` Retorna l'Hora en format "hh:mm".

Per al desenvolupament dels seus mètodes seran de gran utilitat les operacions treballades en la pràctica anterior. Una vegada desenvolupada aquesta classe `Hora`, s'haurà d'escriure una classe programa `Practica4` que execute una sèrie d'instruccions similar a la de la pràctica anterior, `Practica3`, però utilitzant els objectes i mètodes d'`Hora`.

3 Activitats de laboratori

Activitat 1: Desenvolupament i prova de la classe `Hora`. Atributs i mètodes constructors

Obrir *BlueJ* en el directori de treball de l'assignatura `iip` i crear un nou projecte `pract4`. Afegir al projecte el fitxer `Hora.java` disponible en la carpeta de material per a la pràctica de `poliformaT`, que conté l'esquelet de la classe a desenvolupar, incloent el comentari que ha de precedir a cadascun dels mètodes de manera que aquests queden adecuadament documentats.

Els objectes d'aquesta classe contindran tota la informació d'una hora, hores i minuts:

Així doncs, els atributs de la classe hauran de ser

```
private int h;
private int m;
```

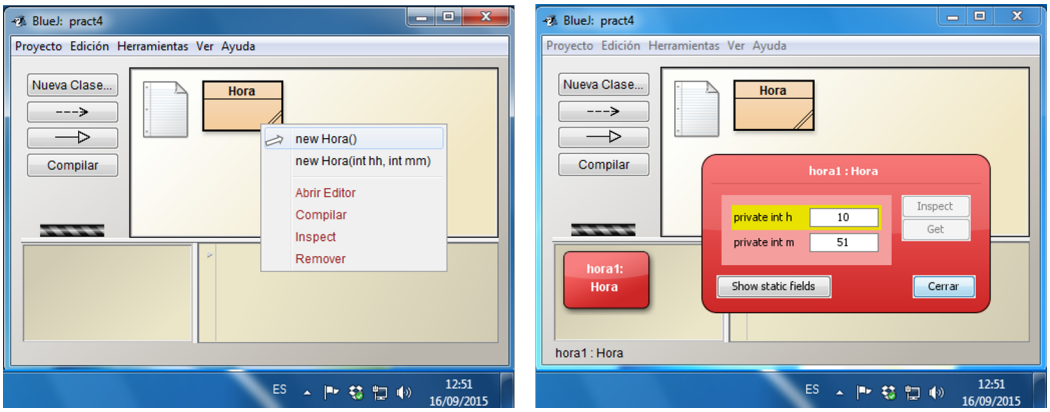
En la classe s'inclourà un primer mètode constructor amb la capçalera o perfil:

```
/**  Hora corresponent a les hh hores i mm minuts.
 *   Precondicio: 0<=hh<24, 0<=mm<60.
 */
  public Hora(int hh, int mm)
```

i s'escriurà la constructora per defecte que cree l'hora amb els valors corresponents a l'hora actual UTC. És a dir, aquest mètode haurà d'encapsular els càlculs que es van realitzar en la pràctica anterior per a calcular hores i minuts de l'hora UTC.

```
/** Hora (hores i minuts) actual UTC (temps coordinat universal).
 */
public Hora()
```

Una vegada editada i compilada aquesta part de la classe, es provarà la creació d'objectes i es verificarà que són correctes, com en l'exemple de la següent figura:



Activitat 2: Desenvolupament i prova de la classe `Hora`. Mètodes consultors i modificadors

Afegir a la classe els mètodes consultors i modificadors les capçaleres dels quals es mostren a continuació:

```
/** Retorna les hores de l'Hora. */
public int getHores()

/** Retorna els minuts de l'Hora. */
public int getMin()

/** Modifica les hores de l'Hora. */
public void setHores(int hh)

/** Modifica els minuts de l'Hora. */
public void setMin(int mm)
```

Abans de seguir afegint més mètodes en la següent activitat, es recompilarà la classe i es provarà que els mètodes ja escrits són correctes. Per a açò es crearan objectes, bé en el *workbench* bé en el *Code Pad* de *BlueJ*, i se lis aplicaran els mètodes verificant el seu resultat.

Activitat 3: Desenvolupament i prova de la classe `Hora`. Mètodes `toString`, `equals`, `aMinuts` i `compareTo`

Afegir a la classe els mètodes les capçaleres dels quals es mostren a continuació:

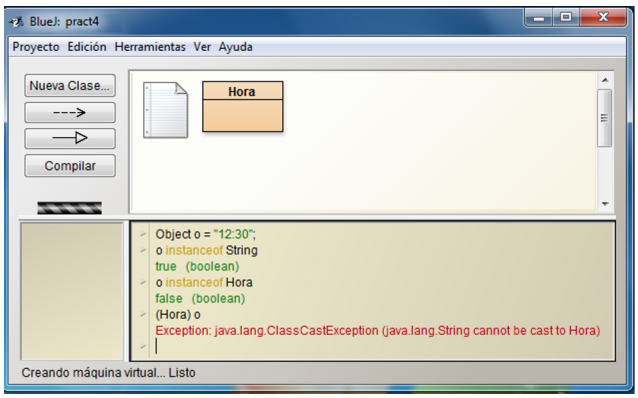
```
/** Retorna l'Hora en format "hh:mm".
 */
public String toString()

/** Retorna true sii o es una Hora que coincideix
 *  en hores i minuts amb l'Hora.
 */
public boolean equals(Object o)

/** Retorna el nombre de minuts transcorreguts
 *  des de les 00:00 fins l'Hora.
 */
public int aMinuts()

/** Compara cronologicament l'Hora i altraHora. El resultat es un valor:
 *     - negatiu si l'Hora es anterior a altraHora,
 *     - zero si son iguals,
 *     - positiu si l'Hora es posterior a altraHora.
 */
public int compareTo(Hora altraHora)
```

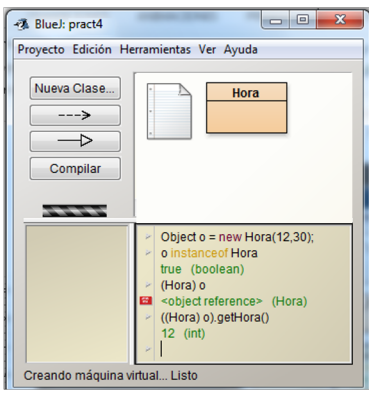
A l'hora d'implementar el mètode `equals` es recorda que, a causa de l'operador curtcircuitat `&&`, és important l'ordre dels operands en l'expressió que compara `o` i la `Hora`:

```
o instanceof Hora  &&  this.h == ((Hora) o).h  &&  this.m == ((Hora) o).m
```

D'aquesta manera, només s'avaluen el segon i tercer operands de l'expressió si `o` és efectivament una `Hora`, i per tant se li pot aplicar el *casting* que permet a Java tractar `o` com un objecte `Hora` i accedir als seus atributs.

Per a comprovar el comportament d'`instanceof` es poden fer proves en el *Code Pad*, com les de la figura següent:





La classe s'haurà de recompilar i provar els mètodes afegits. Per exemple, per a provar `equals` i `compareTo`, es poden crear tres hores `hora1`, `hora2`, `hora3`, corresponents a les 00:00, 12:10, 12:10 respectivament, i comprovar que:

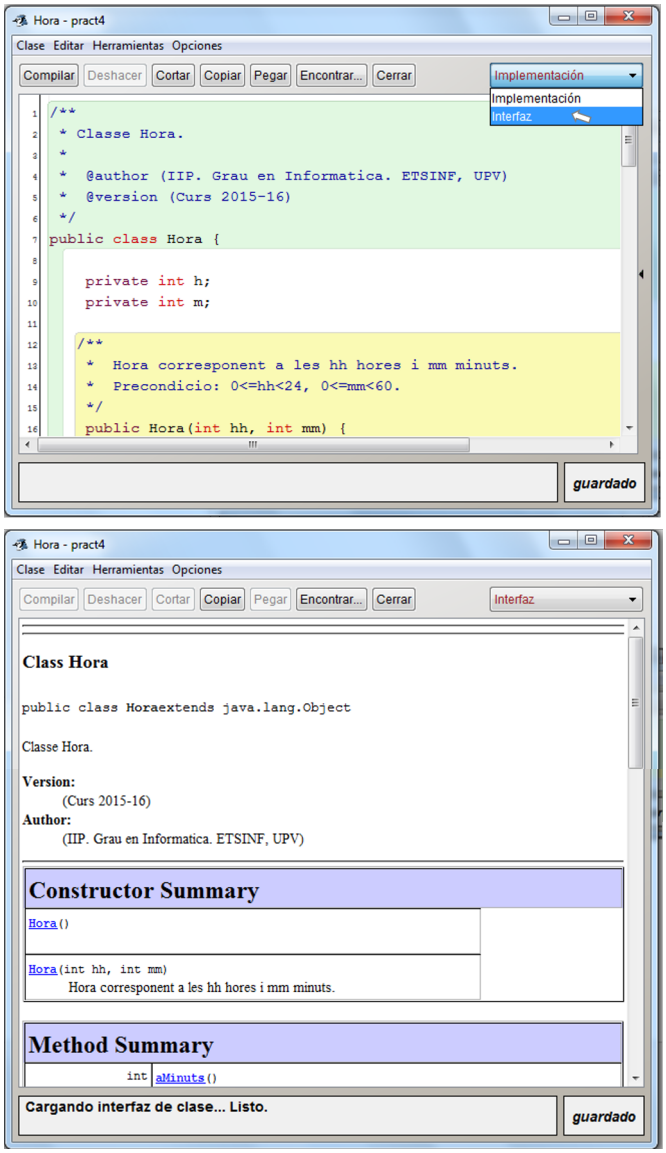
- `hora2` i `hora3` són iguals,
- `hora1` és anterior a `hora2`,
- `hora2` és posterior a `hora1`.

Activitat 4: Comprovació de les normes d'estil de la classe `Hora`

Comprovar que el codi de la classe escrita compleix les normes d'estil usant el `Checkstyle` de *BlueJ*, i corregir-lo si és el cas.

Activitat 5: Obtenció de la documentació de la classe `Hora`

Obtenir la documentació de la classe passant de la manera d'edició o *Implementació* a la manera *Interfície*, com es mostra en la següent figura:



Activitat 6: Desenvolupament de la classe `Practica4`

Afegir al projecte una nova classe `Practica4` que resolga el mateix problema que en la classe `Practica3`, però utilitzant la classe `Hora`. És a dir, `Practica4` serà en bona mida una reescriptura literal de la classe `Practica3`, però tant l'hora les dades de la qual s'introdueixen per teclat, com l'hora actual seran objectes de la classe `Hora`, i s'escriuran en el terminal en el format desitjat usant el mètode `toString` de la classe.

Per a calcular la diferència en minuts entre ambdues hores, es pot resoldre d'una forma anàloga a com es resol en `Practica3`, obtenint les dades de cada hora mitjançant els mètodes consultors, o bé usar el mètode `aMinuts` de la classe.

Activitat extra: Ampliació de la classe `Hora`. Mètode `valueOf`

Una vegada resoltes les activitats anteriors que contemplen els objectius bàsics de la pràctica, es proposa la següent activitat extra que es pot resoldre en el laboratori si queda suficient temps. En qualsevol cas, constitueix un exercici que pot permetre a l'alumne repassar en el seu temps d'estudi alguns conceptes bàsics sobre els tipus `char` i `String`.

En aquest exercici es demana afegir a la classe `Hora` un mètode amb el perfil:

```
/** Retorna una Hora a partir de la descripcio textual en format "hh:mm".
 */
 public static Hora valueOf(String hhmm)
```

que donada la representació en format `"hh:mm"` d'una hora, calcule i retorne l'hora corresponent. Cal notar que el mètode és estàtic, atès que no s'aplica a cap objecte preexistent, i l'única dada amb la que treballa el mètode és un `String`.

El mètode haurà de calcular els valors enters corresponents a l'hora que representa el paràmetre `hhmm`, i amb ells, crearà i retornarà l'objecte `Hora` corresponent. Per al càlcul del valor de tots dos valors, és convenient tenir en compte les següents consideracions:

- Els caràcters `hhmm.charAt(0)` i `hhmm.charAt(1)` corresponen respectivament als dígits de les desenes i unitats de les hores, mentre que els caràcters `hhmm.charAt(3)` i `hhmm.charAt(4)` corresponen als dígits anàlegs dels minuts.
- Encara que existeix compatibilitat entre `char` i `int` atès que per a Java internament un `char` és un codi numèric enter, cal recordar que els codis dels caràcters `'0'` a `'9'` no es corresponen als valors enters 0 a 9. No obstant açò, com aquests codis són consecutius, si `d` és un `char` que conté un dígit qualsevol, l'expressió entera `d - '0'` calcula el valor enter corresponent. Així, si `d` val `'0'` aquesta expressió val 0, si `d` val `'1'`, l'expressió s'avalua a 1, etc ..., com s'observa en els exemples del *Code Pad* de la següent figura: