IIP (E.T.S. d'Enginyeria Informàtica)
Curs 2019-2020

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

L'objectiu principal d'aquesta pràctica és el disseny d'una classe *Tipus de Dades*. Se treballaran diferents aspectes presentats en els temes 3 i 4 (*Variables: definició, tipus i ús en Java* i *Mètodes: definició, tipus i ús en Java*, respectivament). En concret, s'incidirà en:

- Declaració de l'estructura dels objectes d'una classe.
- Declaració dels constructors de la classe, mètodes consultors, modificadors i altres.
- Ús d'una classe tipus de dades: 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 `TimeInstant` perquè tinga una funcionalitat com la que s'indica en la Figura 1. Per al desenvolupament dels seus mètodes seran de gran utilitat les operacions treballades en la pràctica anterior.

Una vegada desenvolupada la classe `TimeInstant`, s'haurà d'implementar una classe programa `Test4` que execute una sèrie d'instruccions similars a les de la pràctica anterior, `Test3`, però utilitzant els objectes i mètodes de la classe `TimeInstant`.

El nom de la classe, `TimeInstant`, reflecteix el que es vol representar: una marca de temps (`Timestamp`); per tant, la classe representa el moment que defineix un instant de temps, en aquest cas, les hores i els minuts d'un dia qualsevol.

Tot i que és bo saber que un `Timestamp` o marca de temps inclou molt més detall (any, mes, dia, hores, minuts, segons i mil·lisegons o microsegons, segons la implementació), en aquesta pràctica, per simplificar, la classe `TimeInstant` únicament farà servir dos atributs, `hours` i `minutes`.

Cal notar que el terme `Timestamp` és el que s'usa en les bases de dades com un tipus de dada específica per gestionar dades que són marques de temps.

Constructor Summary

Constructors

Constructor and Description
`TimeInstant()` Crea un `TimeInstant` amb el valor de l'instant actual UTC (temps universal coordinat).
`TimeInstant(int iniHours, int iniMinutes)` Crea un `TimeInstant` amb el valor de les hores i els minuts que rep com arguments, `iniHours` i `iniMinutes` respectivament.

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`int`	`compareTo(TimeInstant tInstant)` Compara cronològicament l'instant de l'objecte en curs amb el de l'objecte de la classe `TimeInstant` referenciat per `tInstant`.
`boolean`	`equals(java.lang.Object o)` Torna `true` sii `o` és un objecte de la classe `TimeInstant` i les seues hores i minuts coincideixen amb els de l'objecte en curs.
`int`	`getHours()` Torna les hores del TimeInstant.
`int`	`getMinutes()` Torna els minuts del TimeInstant.
`void`	`setHours(int hh)` Actualitza les hores del TimeInstant.
`void`	`setMinutes(int mm)` Actualitza els minuts del TimeInstant.
`int`	`toMinutes()` Torna el número de minuts transcorreguts des de les 00:00 fins l'instant representat per l'objecte en curs.
`java.lang.String`	`toString()` Torna el TimeInstant en el format "`hh:mm`".
`static TimeInstant`	`valueOf(java.lang.String textInstant)` Torna un `TimeInstant` a partir de la descripció textual (`String`) en format "`hh:mm`".

Figura 1: Application Programming Interface (API) de la classe `TimeInstant`.

3 Activitats de laboratori

Activitat 1: creació del paquet BlueJ `pract4`

a) Descarrega els arxius `TimeInstant.java` i `TimeInstantUnitTest.class`, disponibles en la carpeta de material per a la pràctica 4 de *PoliformaT*.

   El fitxer `TimeInstant.java` conté l'esquelet de la classe a desenvolupar, incloent els comentaris que han de precedir a cadascun dels mètodes. El fitxer `TimeInstantUnitTest.class` serveix per comprovar que s'ha fet una implementació correcta de la classe `TimeInstant` (vegeu l'Activitat 6).

b) Obre el projecte *BlueJ* de treball de l'assignatura (`iip`).

c) Crea un nou paquet (Edició - Nou Paquet) de nom `pract4` i obre'l fent doble clic.

d) Afegeix al paquet `pract4` la classe `TimeInstant.java` (Edició - Agregar Classe des d'Arxiu). Comprova que la primera línia inclou la directiva del compilador per indicar que és una classe pertanyent a un paquet (`package pract4;`) i, a continuació, compila la classe `TimeInstant`.

e) Copia l'arxiu `TimeInstantUnitTest.class` a la carpeta `iip/pract4`.

f) Finalment, tanca el projecte `iip` de *BlueJ*, i reobre'l perquè tinga efecte el darrer pas.

Activitat 2: desenvolupament i prova de la classe `TimeInstant`. Atributs i mètodes constructors

Com s'ha comentat anteriorment, cada objecte de tipus `TimeInstant` manté la informació de les hores i els minuts que defineixen una marca o instant de temps. Així doncs, els atributs de la classe seran:

```
private int hours;
private int minutes;
```

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

```
/**
 *  Crea un <code>TimeInstant</code> amb el valor de
 *  les hores i els minuts que rep com arguments,
 *  <code>iniHours</code> i <code>iniMinutes</code>,
 *  respectivament.
 *  <p> Ha de complir-se la precondició:
 *  <code>0 <= iniHours < 24, 0 <= iniMinutes < 60</code>. </p>
 */
public TimeInstant(int iniHours, int iniMinutes)
```

Fixa't com en els comentaris s'inclouen alguns *tags* de `HTML` perquè es mostre millor al navegador. Per exemple, `<code>` o `<p>`.

També s'ha d'escriure el constructor per defecte que cree objectes de tipus `TimeInstant` amb els valors d'hores i minuts corresponents a l'instant actual segons 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 del temps UTC.

```
/**
 * Crea un <code>TimeInstant</code> amb el valor de l'instant
 * actual UTC (temps universal coordinat).
 */
public TimeInstant()
```

Una vegada editada i compilada aquesta part de la classe, utilitzant l'*Object Bench*, es provarà la creació d'objectes i es verificarà que són correctes, com en l'exemple de la de la Figura 2 .

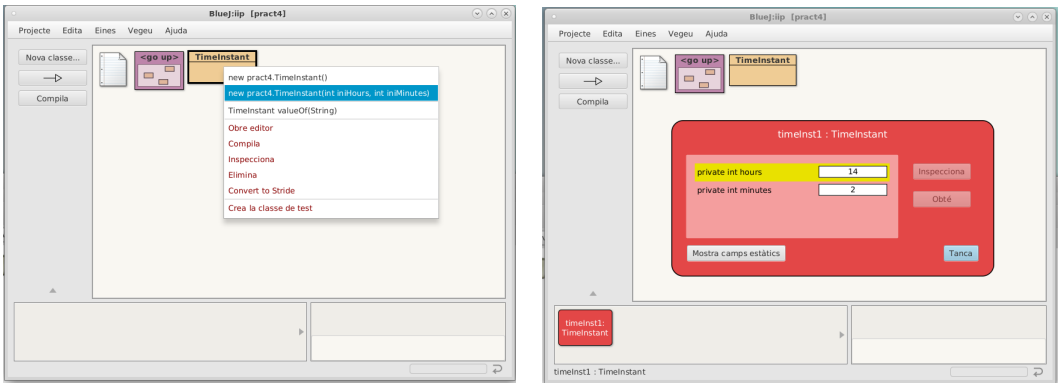

Figura 2: Exemple de com crear objectes de tipus `TimeInstant` i inspeccionar-los.

Activitat 3: desenvolupament i prova de la classe `TimeInstant`. Mètodes consultors i modificadors

Afegeix a la classe els mètodes consultors i modificadors següents:

```
/** Torna les hores del TimeInstant. */
public int getHours()

/** Torna els minuts del TimeInstant. */
public int getMinutes()

/** Actualitza les hores del TimeInstant. */
public void setHours(int hh)

/** Actualitza els minuts del TimeInstant. */
public void setMinutes(int mm)
```

Abans de seguir afegint més mètodes en la següent activitat, recompila la classe i comprova que els mètodes ja implementats són correctes. Per a açò es crearan objectes (bé en el *Object Bench*, o bé en el *Code Pad* de *BlueJ*), i se'ls aplicaran els mètodes verificant el seu resultat.

Activitat 4: desenvolupament i prova de la classe `TimeInstant`. Mètodes `toString`, `equals`, `toMinutes` i `compareTo`

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

```
/** Torna el TimeInstant en el format "<code>hh:mm</code>". */
public String toString()

/** Torna <code>true</code> sii <code>o</code> és
 *  un objecte de la classe <code>TimeInstant</code>
 *  i les seues hores i minuts coincideixen amb els
 *  de l'objecte en curs.
 */
public boolean equals(Object o)

/** Torna el número de minuts transcorreguts des de les 00:00
 *  fins l'instant representat per l'objecte en curs.
 */
public int toMinutes()

/** Compara cronològicament l'instant de l'objecte en curs
 *  amb el de l'objecte de la classe <code>TimeInstant</code>
 *  referenciat per <code>tInstant</code>.
 *  <p>
 *  El resultat és la resta entre la conversió a minuts dels
 *  dos objectes, en particular, aquest resultat serà un valor:
 *  <ul>
 *     <li> negatiu si l'instant de l'objecte en curs
 *     és anterior al de <code>tInstant</code>, </li>
 *     <li> zero si són iguals, </li>
 *     <li> positiu si l'Instant de l'objecte en curs
 *     és posterior al de l'<code>tInstant</code>. </li>
 *  </ul>
 *  </p>
 */
public int compareTo(TimeInstant tInstant)
```

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 si el paràmetre `o` es un objecte de la classe `TimeInstant` i, en cas afirmatiu, si els seus atributs coincideixen en valor amb els de l'objecte en curs:

```
o instanceof TimeInstant
&& this.hours == ((TimeInstant) o).hours
&& this.minutes == ((TimeInstant) o).minutes;
```

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

Per comprovar el comportament d'`instanceof`, es poden fer proves en el *Code Pad*, com les de la Figura 3 .

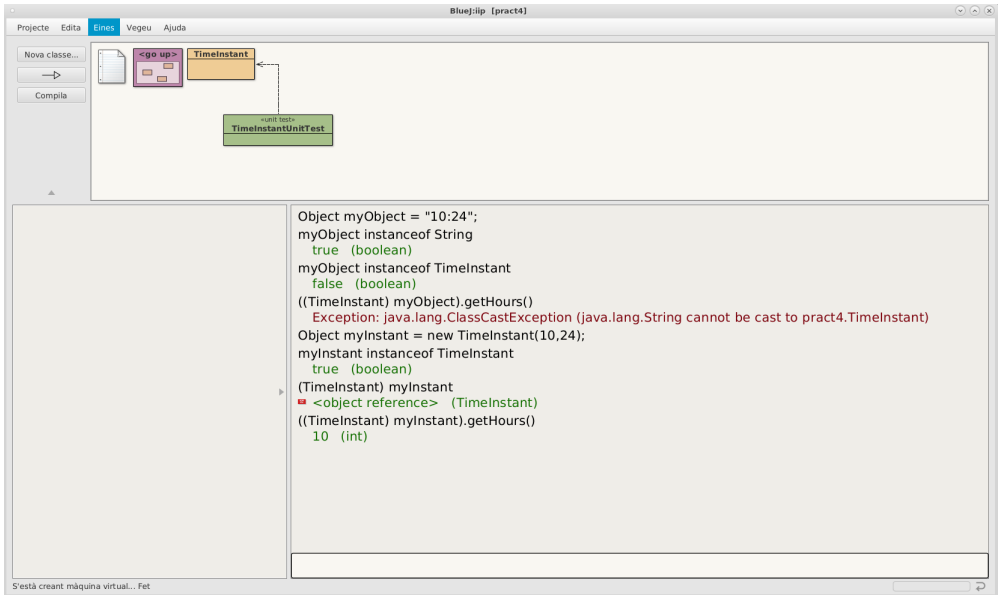

Figura 3: Exemple de com provar el comportament d'`instanceof` al *Code Pad*.

La classe s'ha de recompilar i cal provar els mètodes afegits. Per exemple, per provar `equals` i `compareTo`, es poden crear tres instants `timeInstant1`, `timeInstant2`, `timeInstant3`, corresponents a les 00:00, 12:10, 12:10 respectivament, i comprovar que:

- `timeInstant2` i `timeInstant3` són iguals,
- `timeInstant1` és anterior a `timeInstant2`,
- `timeInstant2` és posterior a `timeInstant1`.

Activitat 5: comprovació de les normes d'estil de la classe `TimeInstant`

Comprova que el codi de la classe escrita compleix les normes d'estil usant el *Checkstyle* de *BlueJ*, i corregeix-lo si no és el cas.

Activitat 6: comprovació del funcionament de la classe `TimeInstant`

Per comprovar el correcte funcionament dels mètodes de la classe `TimeInstant`, s'ha proporcionat la classe `TimeInstantUnitTest`. Aquesta classe és una `TestUnit` que s'executa com es pot observar en la Figura 4 . El resultat del test apareix reflectit en la finestra *Resultats del Test* de *BlueJ*. Si els mètodes són correctes, apareixeran marcats amb el símbol ✓ (en color verd). Si, pel contrari, algun mètode no funciona correctament, apareixerà marcat amb el símbol `X`. Amb un clic de ratolí, es mostrarà un missatge orientatiu sobre la possible causa de l'error en la part inferior de la finestra.

Activitat 7: obtenció de la documentació de la classe `TimeInstant`

Genera la documentació de la classe passant de la manera *Implementació* a la manera *Interfície*, tal i com es mostra en la Figura 5 .

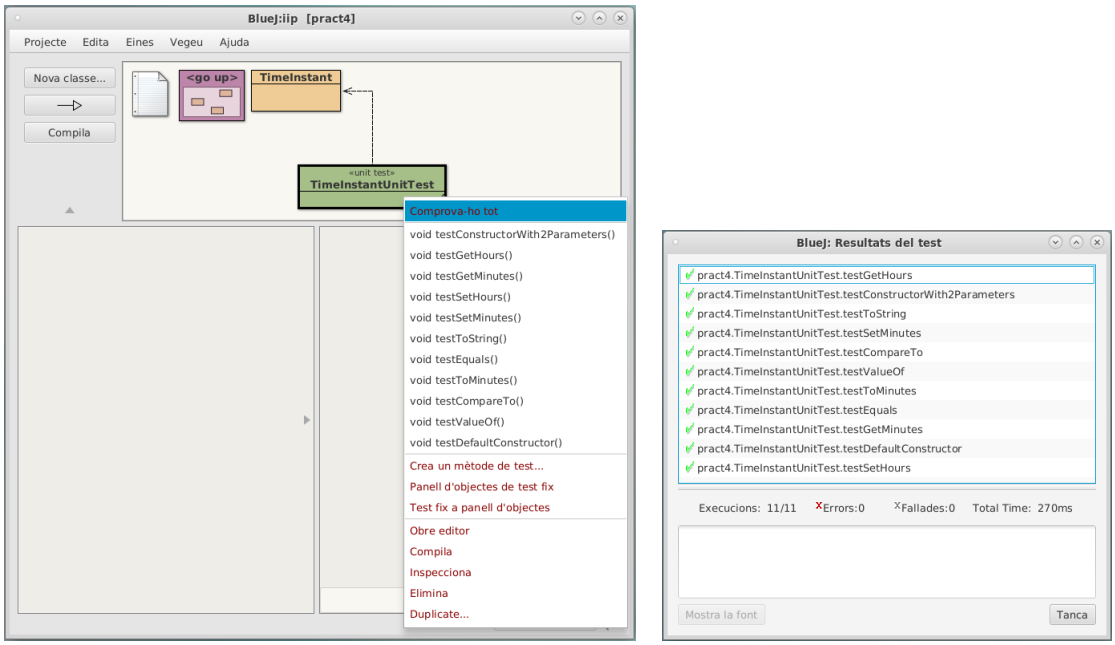

Figura 4: Execució de tots els tests inclosos en la *TestUnit* `TimeInstantUnitTest.class`.

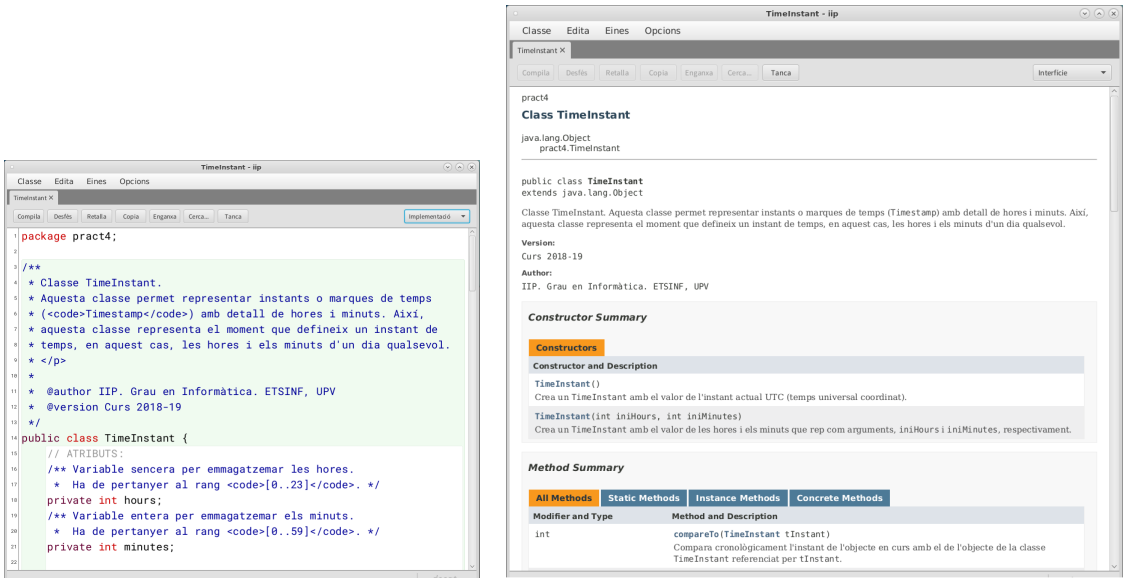

Figura 5: Exemple de com passar al mode *interfície* o *documentació*.

Activitat 8: desenvolupament de la classe `Test4`

Afegeix al paquet `pract4` una nova classe `Test4` que resolga el mateix problema que el de la pràctica 3, però aquesta vegada utilitzant la classe `TimeInstant`. És a dir, `Test4` serà en bona mida una reescriptura del codi de la pràctica 3, però tant la representació temporal que s'introdueix per teclat com el temps UTC s'han de gestionar mitjançant objectes de la classe `TimeInstant`, mostrant-se per pantalla en el format desitjat usant el mètode `toString`.

El càlcul de la diferència en minuts entre ambdos instants de temps, es pot resoldre d'una forma anàloga a com es va fer en la classe `Test3`, obtenint els atributs de cada objecte mitjançant els

mètodes consultors corresponents, o bé fent ús dels mètodes `toMinutes` o `compareTo` de la classe `TimeInstant`.

Activitat extra: ampliació de la classe `TimeInstant`. 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 `TimeInstant` un mètode amb el perfil següent:

```
/** Torna un <code>TimeInstant</code> a partir de la descripció
 *  textual (<code>String</code>) en format "<code>hh:mm</code>".
 */
public static TimeInstant valueOf(String textInstant)
```

que donat un instant de temps expressat mitjançant una cadena de text en el format `"hh:mm"`, calcule i retorne l'objecte de la classe `TimeInstant` corresponent. Cal notar que el mètode `valueOf` és un mètode estàtic (de classe) i, per tant, no es podrà invocar respecte de cap objecte preexistent, sent l'objecte `textInstant` de tipus `String` que rep com argument, la seua única dada d'entrada.

El mètode ha de calcular els valors enters corresponents a l'instant representat per `textInstant`, i amb ells, ha de crear i retornar l'objecte `TimeInstant` corresponent. Per al càlcul de tots dos valors, és convenient tenir en compte les següents consideracions:

- Els caràcters `textInstant.charAt(0)` i `textInstant.charAt(1)` corresponen, respectivament, als dígits de les desenes i unitats de les hores, mentre que els caràcters `textInstant.charAt(3)` i `textInstant.charAt(4)` corresponen als dígits anàlegs dels minuts.

- Encara que existeix compatibilitat entre el tipus `char` i `int`, atès que internament una dada de tipus `char` és un codi numèric enter, cal recordar que els codis dels caràcters `'0'` a `'9'` no es corresponen amb els 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 `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 Figura 6 .

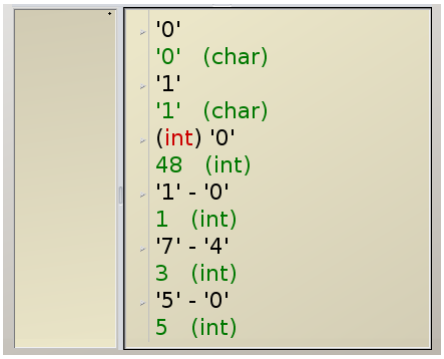

Figura 6: Exemple de com obtenir el valor enter a partir d'operacions amb valors de tipus `char`.