

IIP (E.T.S. d'Enginyeria Informàtica)

Curs 2015-2016

Pràctica 3. Elements bàsics del llenguatge i del compilador

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	1
3	Compilació de codi Java. Detecció d'errors	2
4	Errors d'execució. Prova dels programes	3
5	Normes d'estil en l'escriptura de codi	3
6	Activitats de laboratori	4

1 Objectius i treball previ a la sessió de pràctiques

L'objectiu principal d'aquesta pràctica és iniciar a l'alumne en el treball de desenvolupament de codi, compilació i prova dels programes, al mateix temps que s'exerciten alguns dels conceptes bàsics introduïts en el tema 3 (*Variables: definició, tipus i usos*) sobre la sintaxi de les classes, els tipus numèrics enters de Java i l'ús d'algunes de les seues llibreries fonamentals.

En particular, s'incidirà en:

- Introducció de dades per teclat i escriptura de resultats en el terminal.
- Manipulació de l'aritmètica d'enters.
- Compatibilitat de tipus i conversió forçada de la representació (*casting*).
- Sobrecàrrega de l'operador + (concatenació de cadenes de caràcters a més de l'operació suma).
- Consulta i ús de les llibreries `String`, `System` i `Math`.

2 Descripció del problema

És habitual en moltes aplicacions informàtiques que es precise manejar com a dada una hora, poder calcular el temps transcorregut entre diferents hores, saber si una hora donada és anterior o posterior a una altra, ...

En concret, en aquesta pràctica s'ha d'escriure una Classe-Programa que escriga en la sortida les dades corresponents a una hora (hora i minuts) en el format "`hh:mm`". El programa haurà de calcular la diferència en minuts entre un parell d'hores: una hora les dades de la qual s'hauran

d'introduir per teclat, i l'hora actual, que s'haurà de calcular automàticament. A més, ambdues hores s'hauran de mostrar en el terminal en el format indicat més amunt.

Les activitats proposades estan orientades a discutir les dificultats que poden sorgir en realitzar aquests càlculs.

3 Compilació de codi Java. Detecció d'errors

El procés d'obtenció de codi executable a partir del font precisa que aquest siga sintàcticament correcte; sinó, la traducció no és possible i la compilació falla. El programa compilador, com a conseqüència de l'anàlisi sintàctica parcial o totalment realitzada, produeix una llista de les línies en les quals ha trobat els errors i alguna indicació de l'origen de l'error. Donada la complexitat de l'anàlisi, aquestes indicacions poden no ser tot el precises que seria desitjable, necessitant-se en ocasions certa experiència per a saber interpretar adequadament aquests missatges.

Es recomana en qualsevol cas, llegir els missatges amb deteniment. A continuació se citen uns pocs exemples d'entre l'àmplia casuística que sol presentar-se:

- No es reconeix un identificador o símbol. Pot ser que s'haja teclejat malament el nom d'una variable, un mètode o una classe, i no coincidisca amb el de la seua declaració. En el cas d'una variable, pot ser que s'haja oblidat incloure aquesta declaració. En el cas d'usar mètodes d'una classe que no siga de `java.lang`, pot ser que s'haja oblidat la importació del paquet que la conté.
- En una expressió sintàcticament incorrecta es pot detectar la falta d'un parèntesi o un altre signe, la falta d'un operand, etc ... A voltes, la interpretació del missatge d'error ha d'anar més enllà d'una lectura al peu de la lletra. Considereu per exemple, la següent línia de codi en la qual es pretenia sumar `x + y` i en la qual per error s'ha teclejat un blanc en lloc de `+`:

```
int z = x y;
```

L'anàlisi sintàctica avança fins a `int z = x`, notant-se en aquest punt que la instrucció no està completa. El compilador pot donar en el missatge d'error una indicació que un `;` finalitzaria de forma sintàcticament correcta l'assignació; en aquest exemple concret, la intenció del programador era altra i l'expressió s'ha de completar amb el símbol d'operació que falta.

- En l'anàlisi d'un bloc mal parentitzat es pot detectar una clau que falta (oblidada, o que podria correspondre a una clau anterior supèrflua o malament situada). Una clau descol·locada pot també provocar que falte o sobre alguna secció de codi en un determinat lloc i que siga aquest l'error que es detecte. Per exemple:

```
import java.util.Scanner;
{ public class Hola
    public static void main(String[] args) {
        ...
    }
}
```

La primera clau no està en el seu lloc, i en l'anàlisi el que s'adverteix és que després de la clàusula d'importació hauria de seguir una classe.

Segons el cas, pot ser precís que el programador examine una zona de codi per davant o per darrere de la línia en la qual el compilador s'ha percatat de l'error.

4 Errors d'execució. Prova dels programes

Un programa sintàcticament correcte i que es compila sense problemes pot produir errors d'execució: resultats erronis, o estats de fallada o *excepcions* que provoquen la interrupció abrupta de l'execució.

La font dels errors pot ser de diversa naturalesa; en programes xicotets són habitualment deguts a una estratègia de resolució del problema equivocada, algun cas del problema no contemplat, ús incorrecte d'operacions per un coneixement imprecís del seu significat, o fins i tot errates comeses en teclejar el codi que no necessàriament impliquen incorreccions sintàctiques.

És per açò que de forma rutinària s'ha de sotmetre al programa a execucions de prova contrastant els resultats obtinguts amb els esperats.

5 Normes d'estil en l'escriptura de codi

A més de que un codi font siga sintàctica i semànticament correcte, és desitjable que estiga escrit de forma clara i ordenada per a facilitar la seua lectura i manteniment. Succeeix el mateix amb l'edició de textos convencionals, com a llibres i periòdics, que adopten normes d'estil comunament acceptades o sobreenteses, que estructuren el text i faciliten la seua comprensió.

Açò és especialment important en el cas de desenvolupament de codi, perquè al llarg de la vida dels programes, és habitual que aquests siguen mantinguts per diversos programadors. És per açò que s'han desenvolupat unes convencions d'escriptura de codi que, sense ser d'obligat compliment, es recomana que adopte la comunitat de programadors en Java.

Aquestes convencions, que afecten tant al codi com a l'ús de comentaris, van ser desenvolupades per l'anterior propietari de Java, Sun Microsystems Inc., i actualment romanen accessibles per a la seua consulta en

<http://www.oracle.com/technetwork/articles/javase/codeconvtoc-136057.html>

A continuació es repassen algunes de les més elementals.

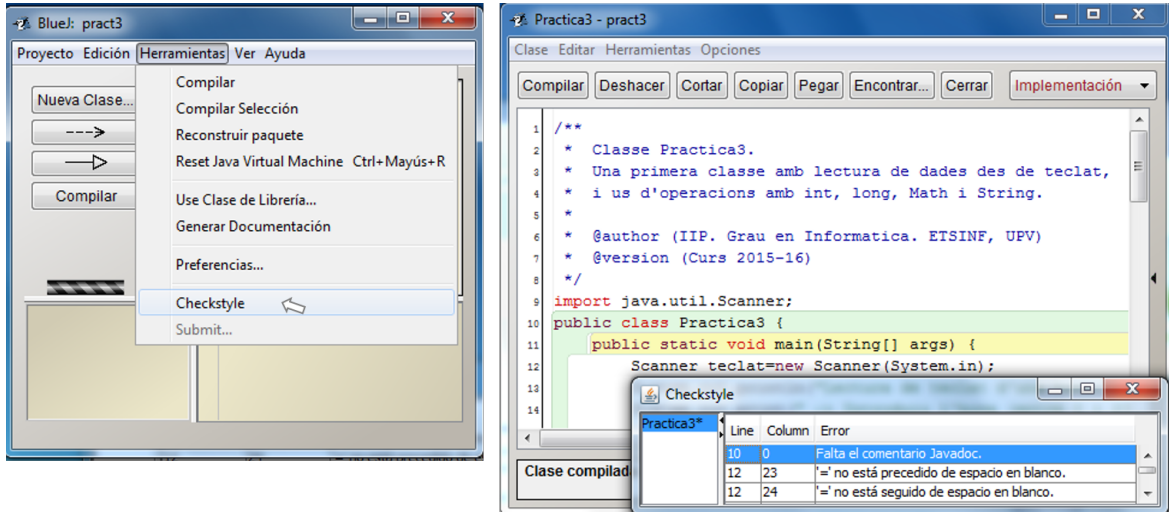
- **Noms de classes, mètodes i variables.** S'han d'escollir identificadors indicadors del seu significat. Mètodes i variables comencen en minúscula, mentre que la inicial de les classes ha d'estar en majúscula. Quan un nom prové de la juxtaposició de dues o més paraules, la segona i successives han de començar per majúscula, com per exemple en `LaPrimeraClasse`, `horaActual`.
- **Parentització de blocs i sagnat del text.** La clau que obri el bloc d'una classe o un mètode, com per exemple `main`, ha d'aparèixer al final de la línia en la qual es declara la classe o mètode. La clau que tanca el bloc ha d'aparèixer amb el mateix sagnat que la classe o mètode. Quan s'escriuen les sentències components d'un bloc, han d'aparèixer sagnades quatre espais cap a la dreta, perquè siga més recognoscible aquesta estructuració, com s'observa en el següent exemple:

```
public class Hola {  
    public static void main(String[] args) {  
        System.out.println("Hola a todos");  
        System.out.println();  
    }  
}
```

- **Escriptura d'expressions.** L'operador d'assignació `=`, i els operadors binaris aritmètics, relacionals i booleans `+`, `*`, `...`, `<`, `<=`, `...`, `&`, `&&`, `...`, hauran d'estar precedits i seguits per un espai.
- **Longitud i ruptura de les línies.** Les línies massa llargues són desaconsellables. Quan cal partir línies, s'hauran de seguir les normes que faciliten reconèixer l'estructura de les expressions i instruccions, com en el següent exemple:

```
System.out.println("La primera arrel real de l'equació val "
    + (-b + Math.sqrt(b * b - 4.0 * a * c)) / (2.0 * a));
```

Per a promoure l'ús d'aquestes convencions s'ha instal·lat l'extensió de *BlueJ Checkstyle*. En concret, si una volta resolt els errors de compilació es marca l'opció *Checkstyle* del desplegable de *Herramientas* de *BlueJ*, apareix una finestra que proporciona informació sobre les normes d'estil incomplides pel codi que s'està escrivint, como en l'exemple de la següent figura:



6 Activitats de laboratori

Activitat 1: Creació del projecte BlueJ pract3

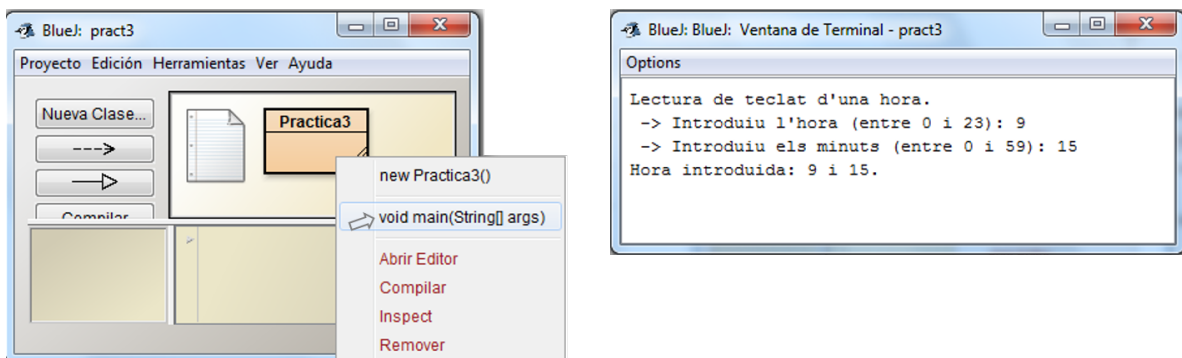
Obrir *BlueJ* en el directori de treball de l'assignatura (iip) i crear el nou projecte *pract3*. Agregar al projecte el fitxer *Practica3.java* que s'haurà descarregat prèviament de

```
poliformat>IIP:recursos/Laboratorio/Práctica 3.
```

Activitat 2: Correcció d'errors sintàctics de la classe Practica3

El codi font de la classe *Practica3.java* agregada al projecte conté deliberadament tres errors sintàctics que impedeixen obtenir el codi compilat. S'ha d'editar la classe per a cercar i corregir aquests errors atenent als missatges d'error produïts en intentar la compilació.

En la següent figura es mostra un exemple d'execució de la classe una vegada corregida i compilada correctament:

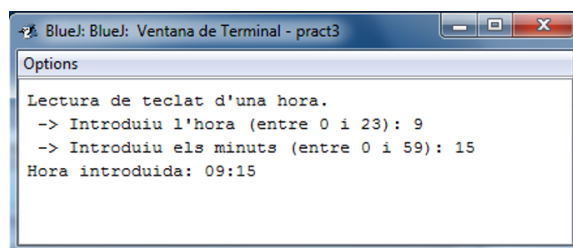


Activitat 3: Comprovació de les normes d'estil

Utilitzar l'opció de comprovar l'estil del menú *Herramientas*, i aplicar els suggeriments de millora d'estil que s'indicaren.

Activitat 3: Escriptura de l'hora introduïda en el format hh:mm

S'ha de modificar el programa anterior perquè l'hora introduïda s'escriu en l'eixida en el format de quatre dígit **hh:mm**, completant amb zeros si cal, com es mostra en l'execució exemple de la següent figura:



Per a resoldre el problema es recomana usar les operacions de la classe **String**, de manera que per a un parell hora i minuts es componga un **String** de la forma **"hh:mm"** que siga el que s'escriu en l'eixida.

Una manera directa i senzilla d'aconseguir-ho és anteposar sempre **"0"** a les hores (respectivament als minuts) quedant un **String** de dos o tres caràcters segons el cas. Exemples: si **h** val 9, l'expressió **"0" + h** dona **"09"**; si **h** val 10, el **String** resultant és **"010"**. Independentment del valor de **h**, n'hi ha prou amb obtenir el substring format pels dos últims caràcters d'aquesta **String** per a obtenir la representació en dos dígit que es precisa. Per a açò, és convenient recordar que, segons la documentació de **String**:

- El mètode **int length()** permet obtenir el nombre n de caràcters d'un **String**, que se suposen numerats de 0 a $n - 1$.
- El mètode **String substring(int i)**, obté el substring format pels caràcters del **String** des de l' i fins a l'últim, tots dos inclusivament. En aquest problema, el substring que interessa és el que s'estén des de l'índex $n - 2$ endavant.

Suposant que en la variable **String hh** s'haguera obtingut l'hora en format de dos dígit i en la variable **mm** els minuts en el mateix format, l'hora en el format sol·licitat s'obté mitjançant la concatenació **hh + ":" + mm**.

S'haurà de comprovar la correcció del programa provant a introduir per a les hores i minuts casos com els de la següent taula i verificant l'hora que s'escriu en l'eixida:

Hora	minuts	"hh:mm"
0	0	"00:00"
9	5	"09:05"
9	35	"09:35"
19	5	"19:05"
19	35	"19:35"

Activitat 5: Càlcul de l'hora actual (Temps Universal Coordinat).

El programa s'haurà de completar calculant les hores i minuts corresponents al moment d'execució del programa. Per a açò, es pot fer una crida al mètode

```
System.currentTimeMillis()
```

que, com s'indica en la documentació de la classe `System`, retorna el nombre de mil·lisegons transcorreguts entre les 00:00 UTC ¹ de l'1 de gener de 1970 i el moment actual UTC.

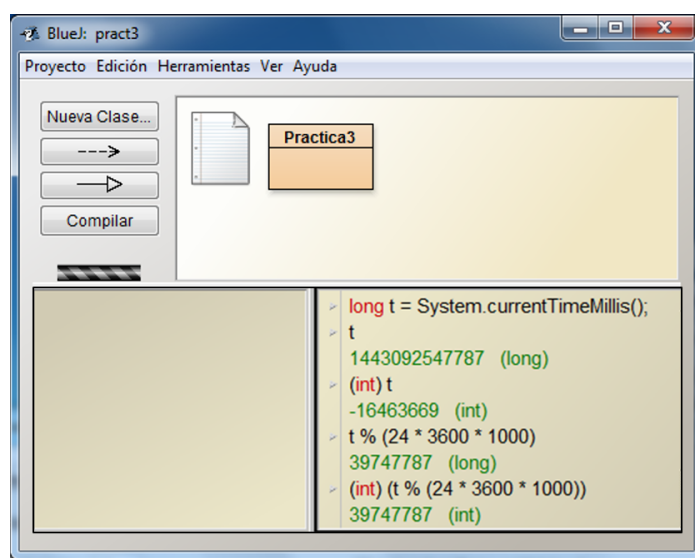
El mètode retorna una quantitat entera de tipus `long`, atès que el seu valor és molt gran i no cap en una variable de tipus `int`. La següent expressió Java de tipus `long` calcula la resta de dividir aquesta quantitat entre el nombre de mil·lisegons per dia:

```
System.currentTimeMillis() % (24 * 3600 * 1000)
```

Aquests mil·lisegons restants corresponen per tant als transcorreguts des de les 00:00 del dia actual fins al moment en que s'executa el càlcul de l'expressió. Com el valor calculat es troba en el rang $[0, 86400000]$, és possible representar-lo com `int`. En la següent instrucció es força mitjançant un *casting* aquesta transformació de representació sabent que el valor enter corresponent es preserva, i podent assignar-se a més a una variable de tipus `int` sense que es produïska un error de compilació per incompatibilitat de tipus:

```
int tRestant = (int) (System.currentTimeMillis() % (24 * 3600 * 1000));
```

En la figura que ve a continuació es mostren exemples del comportament del *casting* de `long` a `int` segons el rang del valor enter transformat:



A partir del valor de `tRestant` calculat com en la instrucció anterior, es pot obtenir mitjançant senzilles operacions aritmètiques les hores i minuts del moment actual que es precisen per a resoldre l'enunciat.

Finalment, s'ha de calcular i mostrar també per pantalla el valor absolut de la diferència en minuts entre l'hora introduïda per teclat i l'hora actual (consultar en la documentació de la classe `Math` els mètodes disponibles per a calcular el valor absolut d'enters i reals).

El resultat de l'execució ha de ser com el que es mostra en l'exemple de la figura de la següent pàgina.

¹ *Temps Universal Coordinat*, estàndard mundial equivalent en la pràctica a l'hora *Greenwich*. A Espanya, excepte Canàries, regeix l'hora *Greenwich* + 2 en horari d'estiu, i l'hora *Greenwich* + 1 en horari d'hivern.

```
BlueJ: BlueJ: Ventana de Terminal - pract3
Options
Lectura de teclat d'una hora.
-> Introduiu l'hora (entre 0 i 23): 9
-> Introduiu els minuts (entre 0 i 59): 15
Hora introduïda: 09:15
Hora actual: 11:22 (temps UTC)
Diferencia en minuts entre ambdues hores: 127
```

Activitats extra.

Una vegada resoltes les activitats bàsiques de la pràctica, es proposen les següents activitats extra que es poden resoldre en el laboratori si queda suficient temps. En qualsevol cas, constitueixen uns exercicis que poden permetre a l'alumne repassar en el seu temps d'estudi algunes peculiaritats de la concatenació de **String**, i alguns conceptes bàsics sobre la sintaxi de l'expressions de tipus **boolean**.

1. Resoldre el càlcul del format "**hh:mm**" d'escriptura d'una hora d'una manera alternativa, tenint en compte les següents consideracions:

- Donada la variable entera **h** que conté una hora entre 0 i 23, **h / 10** obté la xifra de les desenes (0, si **h < 10**), i **h % 10** obté les unitats.
- Ambdues xifres es calculen com valors **int**, però en l'expressió **h / 10 + "" + h % 10** apareixen en un context de concatenació + de **String**, pel que Java els converteix a la seua representació textual. L'ús del literal "", objecte **String** de zero caràcters, conseqüència que ambdues xifres apareguen una a continuació de l'altra en el **String** resultant.

Un càlcul anàleg és aplicable a les xifres dels minuts, amb la qual cosa es pot completar l'escriptura de l'hora en el format desitjat.

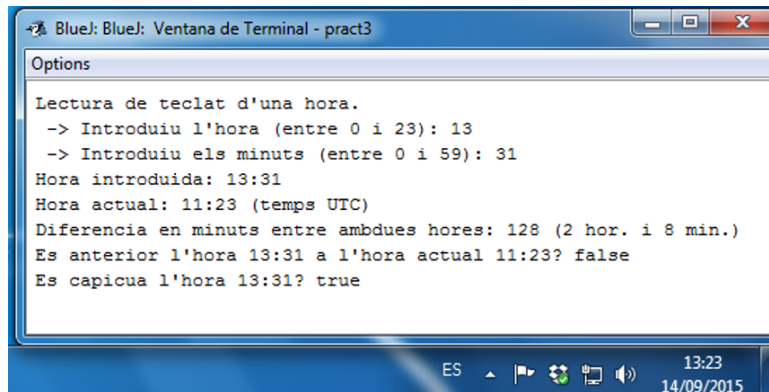
2. Modificar el mètode **main** de la classe **Practica3** perquè la diferència en minuts entre l'hora introduïda i l'actual s'expressi també en hores i minuts.
3. Afegir al mètode **main** de la classe **Practica3** les instruccions necessàries perquè s'escriguen en l'eixida els següents resultats:

- Valor de veritat corresponent al fet que l'hora introduïda és anterior a l'hora actual.
- Valor de veritat corresponent al fet que l'hora introduïda per teclat, en el seu format "**hh:mm**", siga un capicua. Consulteu en la documentació de **String** quin mètode permet conèixer el caràcter que apareix en una posició donada.

S'hauran de provar casos com els de la següent taula:

Hora	minuts	"hh:mm"	Capicua?
0	0	"00:00"	true
2	20	"02:20"	true
13	31	"13:31"	true
19	21	"19:21"	false
13	35	"13:35"	false
19	35	"19:35"	false

Els missatges resultants han de ser com els que es mostren en la següent execució exemple:



```
Blue: Blue: Ventana de Terminal - pract3
Options
Lectura de teclat d'una hora.
-> Introduiu l'hora (entre 0 i 23): 13
-> Introduiu els minuts (entre 0 i 59): 31
Hora introduida: 13:31
Hora actual: 11:23 (temps UTC)
Diferencia en minuts entre ambdues hores: 128 (2 hor. i 8 min.)
Es anterior l'hora 13:31 a l'hora actual 11:23? false
Es capicua l'hora 13:31? true
```

The screenshot shows a terminal window with a blue title bar. The window contains a program that prompts the user to enter an hour and minutes. The user enters 13 for the hour and 31 for the minutes. The program then displays the current time as 11:23 (UTC) and calculates the difference in minutes between the entered time (13:31) and the current time (11:23), which is 128 minutes (2 hours and 8 minutes). It then checks if the entered time is anterior (earlier) than the current time, returning false, and if it is a palíndrom (capicua), returning true.