



Fondamenti di Informatica L-B

Esercitazione n°5



Java: Interfaccia Comparable, Array

A.A. 2007/08

Tutor: Barbara Pettazzoni

barbara.pettazzoni@studio.unibo.it



Interfaccia Comparable

- L'altra volta abbiamo visto il metodo **equals** per poter confrontare se due oggetti sono uguali o meno.
- Ma... e se volessimo ordinare una serie di oggetti? Ovvero, non ci interessa soltanto l'uguaglianza, ma se un oggetto viene prima o dopo un altro, secondo un certo criterio d'ordinamento...definito da noi!
- Per far cio', Java mette a disposizione l'interfaccia **Comparable**



Interfaccia Comparable: esempio!

```
public class Contatore implements Comparable
{
    private int valore;
    ...
    public int compareTo(Contatore c)
    {return valore-c.valore;}

    public boolean equals(Contatore c)
    {return (valore == c.valore);}

    public int compareTo(Object o)
    {return compareTo((Contatore) o);}

    public boolean equals(Contatore c)
    {return equals((Contatore) o);}

}
```

3



Interfaccia Comparable: esempio!

```
public class TestComparableContatore
{
    public static void main(String args[])
    {
        Contatore c1 = new Contatore();
        Contatore c2 = new Contatore();
        c1.incrementa();
        if(c1.compareTo(c2) > 0)
            System.out.println("c1 e' piu' grande
                                di c2");
        else    System.out.println("c2 e' piu' grande
                                di c1");
    }
}
```

4



Gli array in Java

- Gli array in Java sono *oggetti*, istanze di una particolare classe, `[]`
- Quindi, devono essere trattati come oggetti normali, cioè devono essere definiti...

```
int [] tantiInteri;  
Contatore [] contatori;
```

- ...e quindi devono essere *inizializzati*, creando dinamicamente l'oggetto:

```
tantiInteri = new int[3];  
contatori   = new Contatore[6];
```

5



Gli array in Java

- Durante la *definizione*, le parentesi quadre si possono mettere sia dopo il nome della variabile che dopo il tipo (*a differenza del C!!*)

```
int [] tantiInteri;  
int tantiInteri [];
```

É esattamente
la stessa cosa!!

- La *dimensione* dell'array viene definita durante l'inizializzazione del vettore!

6



Gli array in Java

- Attenzione però! L'inizializzazione non é ancora finita!

```
int [] tantiInteri = new int[3];
```

In questo caso, trattandosi di un array di *tipi primitivi*, ogni elemento del vettore é una **variabile!**

```
Contatore [] contatori = new Contatore[3];
```

In questo caso, trattandosi di un array di *oggetti*, ogni elemento del vettore é un **referimento ad un (futuro) oggetto...quindi al momento punta a null!!** **Bisogna creare l'oggetto per usarlo!!**

7



Un esempio

- Provate il seguente codice:

```
public class TestArrayContatore
{
    public static void main(String args[])
    {
        Contatore [] contatori = new Contatore[3];
        contatori[0].incrementa();
    }
}
```

- Vi verrà lanciata una ***NullPointerException*** : infatti, state cercando di accedere ad un oggetto non ancora inizializzato!

8



Una domanda!

- Perché il seguente codice lancia anche lui una `NullPointerException`?

```
public class TestArrayContatore
{
    public static void main(String args[])
    {
        Contatore [] contatori = new Contatore[2];
        contatori[0] = new Contatore();
        contatori[1] = new Contatore();
        contatori = new Contatore[3];
        contatore[0].incrementa();
    }
}
```

Quando facciamo di nuovo **new**, contatori punta a una nuova zona di memoria: le inizializzazioni che abbiamo fatto prima non sono più valide!

9



Gli array in Java

- Gli array sono oggetti, e come tutti gli oggetti hanno proprietà e metodi che possiamo usare. In particolare, la proprietà **length** indica la lunghezza dell'array

```
tantiInteri = new int[3];
contatori    = new Contatore[6];
tantiInteri.length //vale 3
Contatori.length  //vale 6
```

- Per quello visto in precedenza, non si può estendere un array (a meno di perdere ciò che si era già memorizzato).
- Per far ciò, esistono delle classi apposite, che vedremo nella prossima esercitazione.

10



Una piccolissima parentesi sull'ereditarietà

- In Java, qualunque classe é figlia della classe Object (questa é il padre di tutti gli oggetti): mediante un sistema detto *ereditarietà*, noi estendiamo la classe Object aggiungendo metodi e attributi!
- La classe Object ha però dei suoi metodi, che quindi qualunque oggetto ha! Se non vengono *ridefiniti*, verranno richiamati i metodi definiti nella classe Object
- Noi abbiamo già visto un metodo del genere...ve lo ricordate?
- Il metodo `equals(Object o)` se non viene ridefinito, verifica l'uguaglianza fra due riferimenti...che é appunto ciò che fa il metodo equals della classe Object!

11

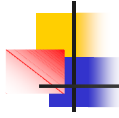


Il metodo `toString`

- Si tratta di un altro metodo che é presente all'interno della classe Object; se non viene ridefinito, significa che si utilizzerá il metodo proprio di questa classe.
- Definiamalo sempre per la classe contatore.

```
public class Contatore
{
    private int valore;
    ...
    public String toString()
    {
        return "Il valore del contatore é' " + valore;
    }
}
```

12



Esempio

```
public class TestToStringContatore
{
    public static void main(String args[])
    {
        Contatore c = new Contatore(5);
        System.out.println(c);
        c.incrementa();
        System.out.println(c);
    }
}
```

Java richiama automaticamente il metodo toString!!

13



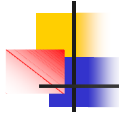
Esercizi

- Scrivete un programma che sia in grado di stampare tutti (e i soli) parametri che vengono passati al main (ovvero il vettore di String args). Se non vengono passati parametri, stampare un messaggio adatto.
- Scaricate dal sito i file trama.txt, trama2.txt e trama3.txt. Dovete realizzare un programma che conti in ognuno le parole presenti e le occorrenze della parola "Anello". Si deve quindi creare un file risultati.txt che contenga delle linee del tipo:

*trama.txt rispetto a trama2.txt contiene più parole.
trama2.txt rispetto a trama.txt contiene più occorrenze della parola Anello.*

Seguite appunto dai valori dei contatori corrispondenti.

14



Soluzione primo esercizio

```
public class StampaParametri
{
    public static void main(String args[])
    {
        if(args.length == 0)
            System.out.println("Nessun parametro!");
        else
            for(int i = 0; i < args.length; i++)
            {
                System.out.println("Parametro "+i+
                                   "simo: " + args[i]);
            }
    }
}
```

15

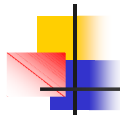


Soluzione secondo esercizio (1)

```
import fiji.io.*;
import java.io.*;
import java.util.*;

public class ContaJava2
{
    public static void main(String args[])
    {
        try
        {
            String fileNames[] =
                {"trama.txt", "trama2.txt", "trama3.txt"};
            Contatore contatoriParole[] = new Contatore[3];
            Contatore contatoriAnello[] = new Contatore[3];
            ...
        }
    }
}
```

16



Soluzione secondo esercizio (2)

```
...
for(int i=0; i < 3; i++)
{
    contatoriParole[i] = new Contatore();
    contatoriAnello[i] = new Contatore();
    Lettore input = new Lettore(new FileReader(fileNames[i]));
    while(!input.eof())
    {
        String linea = input.leggiLinea();
        StringTokenizer tokenizer = new StringTokenizer(linea, " ");
        while(tokenizer.hasMoreTokens())
        {
            String parola = tokenizer.nextToken();
            if(parola.equals("Anello"))
                contatoriAnello[i].incrementa();
            contatoriParole[i].incrementa();
        }
    }
}
...
```

17



Soluzione secondo esercizio (3)

```
...
FileWriter fw = new FileWriter("risultati.txt");
BufferedWriter bw = new BufferedWriter(fw);
for(int i=0; i < 2; i++)
{
    for(int j = i + 1; j < 3; j++)
    {
        if(contatoriParole[i].compareTo(contatoriParole[j]) > 0)
            bw.write(fileNames[i] + " rispetto a " + fileNames[j] +
                " contiene piu' parole: " +
                contatoriParole[i].getValore() +
                ">" + contatoriParole[j].getValore());
        else
            bw.write(fileNames[j] + " rispetto a " + fileNames[i] +
                " contiene piu' parole: " +
                contatoriParole[j].getValore() + ">" +
                contatoriParole[i].getValore());
    }
}
...
```

18



Soluzione secondo esercizio (4)

```
...  
        bw.flush();  
        bw.close();  
    }  
    catch (Exception e)  
    {  
        e.printStackTrace();  
    }  
}
```