Varie:
final, visibilità, parametri di ingresso

Classi e metodi `final`

È possibile impedire la creazione di sottoclassi di una certa classe definendola **`final`**

Esempio:

```
final class C {...}
class C1 extends C // errato
```

Similmente, è possibile impedire l'overriding di un metodo definendolo **`final`**

Esempio:

```
class C { final void f() {...} }
class C1 extends C {
  void f() {...} // errato
}
```

Information hiding in Java

- La visibilità degli attributi e metodi di una classe `C` può essere:
  - **`public`**
    - visibili a tutti
    - vengono ereditati
  - **`protected`**
    - visibili solo alle sottoclassi
    - vengono ereditati
  - **«package» (nessun modificatore specificato)**
    - visibili solo alle classi dichiarate nel package di `C`
  - **`private`**
    - visibili solo all'interno della stessa classe
    - non visibili nelle sottoclassi

Modificatori di visibilità

	visibilità			
modificatore	**classe**	**package**	**sottoclasse**	**mondo**
`private`	Y	N	N	N
non dichiarato	Y	Y	N	N
`protected`	Y	Y	Y	N
`public`	Y	Y	Y	Y

Fondamenti di Java

Static

Modificatori: static

Variabili e metodi associati ad una Classe anziche' ad un Oggetto sono definiti "static".

Le variabili statiche servono come singola variabile condivisa tra le varie istanze

I metodi possono essere richiamati senza creare una istanza.

Variabili “static”: esempio 1

```
public class  S {
   static int instanceCount = 0; //variabile “di classe”
   S() {instanceCount++;}

}
public class  A {
   public static void main(String a[]) {
        new A();
   }
   A() {
      for (int  i = 0;  i < 10;  ++i) {
         S instance=new S();
      }
      System.out.println("# of instances:
        "+S.instanceCount);
   }
}
```

Output:

```
# of instances: 10
```

Variabili “static”: esempio 2

```
class  S {
   static int instanceCount = 0; //variabile “di classe”
   S() {instanceCount++;}
   public void finalize() {instanceCount--;}
}
public class  A {
   public static void main(String a[]) {
        new A();
   }
   A() {
     for (int  i = 0;  i < 10;  ++i) {
         S instance=new S();
     }
     System.out.println("# of instances:"+S.instanceCount);
     System.gc();
     System.out.println("# of instances: "+S.instanceCount);
   }
}
```

Output:

```
# of instances: 10
# of instances: 0
```

Metodi "static": esempio 1

```
class  S {
   static int instanceCount = 0; //variabile "di classe"
   S() {instanceCount++;}
   static void azzeraContatore() {instanceCount=0;}
}
public class  A {
   public static void main(String a[]) {
        new A();
   }
   A() {
      for (int  i = 0;  i < 10;  ++i) {
         if (i%4==0) S.azzeraContatore();
         S instance=new S();
      }
      System.out.println("instanceCount:
"+S.instanceCount);
  }
}
```

Può agire solo su variabili statiche!

Output:
instanceCount: 2

Ruolo:
Metodi che agiscono su variabili statiche

metodi "static": esempio 2

```
Math.sqrt(double x);
System.gc();
System.arrayCopy(...);
System.exit();
Integer.parseInt(String s);
Float.parseFloat(String s);
```

Notare la maiuscola! (per convenzione)

Ruolo: analogo alle librerie del C

```
Che cos'e':
System.out.println() ?
```

Attenzione!

Usare variabili static per condividere informazioni è una CATTIVA PRASSI.

- Le variabili globali violano il principio di encapsulazione dello stato nell'oggetto.
- Quando usate, dovrebbero in genere essere static final
- Il ciclo di vita delle variabili globali è "eterno": vengono istanziate a inizio esecuzione e restano vive fino alla fine.
- Presentano problemi in caso di multithreading

Perchè il main è "static"?

```
public class  A {
    String s="hello";
    public static void main(String a[]) {
        System.out.println(s);
    }
}
```

```
Non static variable s cannot be referenced from static context
```

```
public class  A {
    String s="hello";
    public static void main(String a[]) {
      new A();
    }
    A() {
      System.out.println(s);
    }
}
```

```
hello
```

Parametri di ingresso

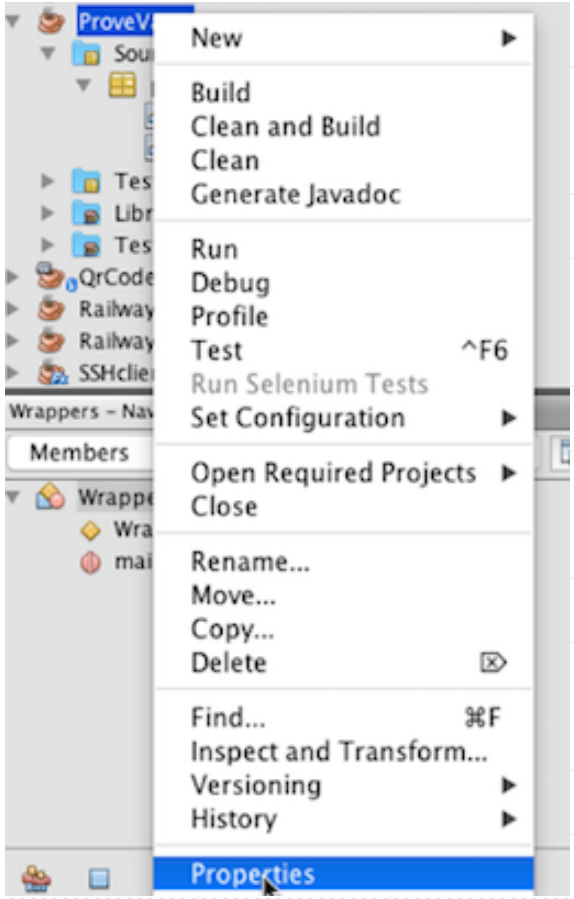
I parametri del main sono inclusi in un vettore di String

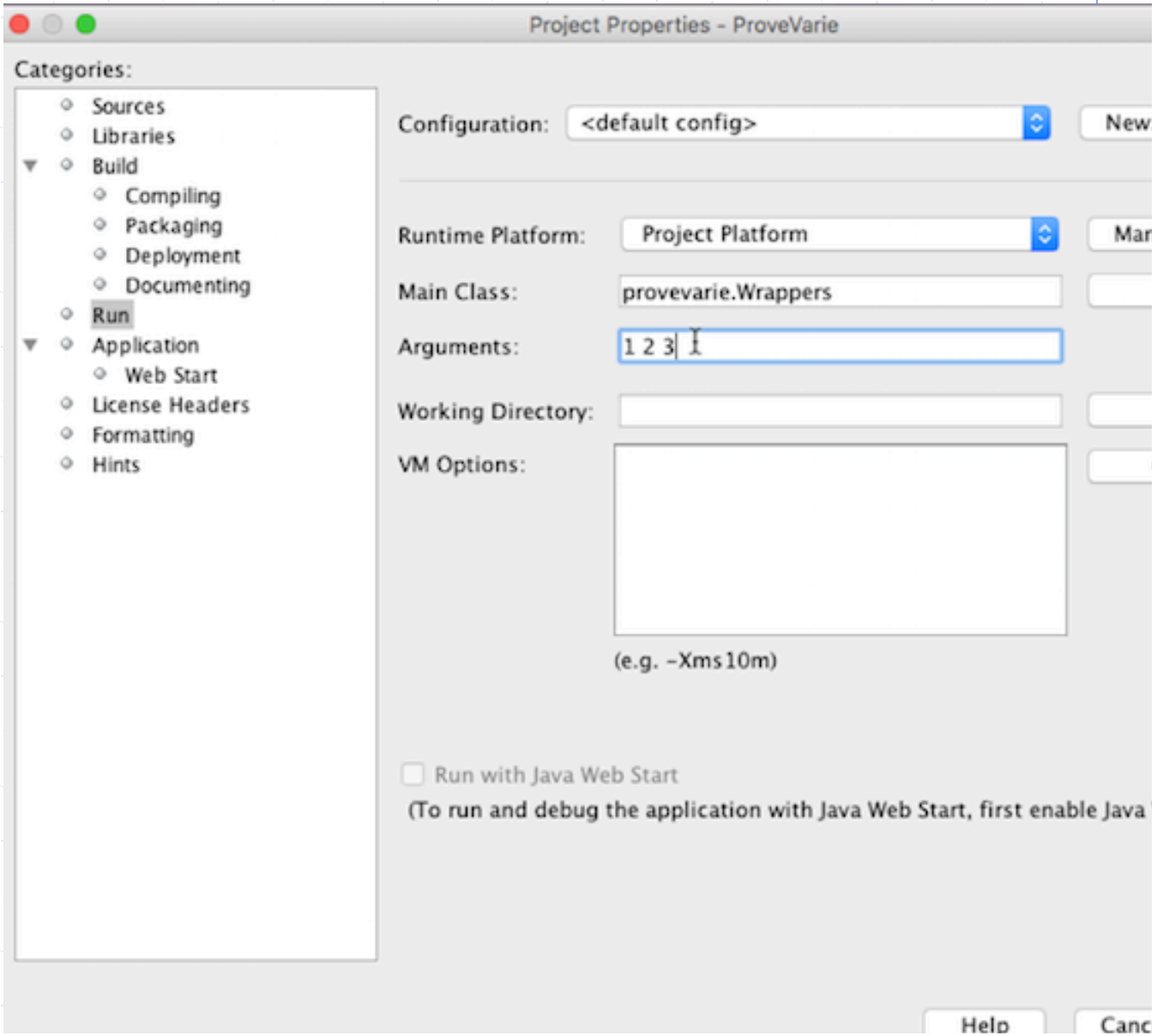
```
/* sum and average command lines */
class SumAverage {
  public static void main (String args[]) {
    int sum = 0;
    float avg = 0;
    for (int i = 0; i < args.length; i++) {
      sum += Integer.parseInt(args[i]);
    }
    System.out.println("Sum is: " + sum);
    System.out.println("Average is: "
        + (float)sum / args.length);
  }
}
```

Parametri di ingresso – altro esempio

```
/* sum and average command lines */
class ElementShower{
   public static void main(String a[]){
         System.out.println(a.length);
         int arg=0;
         for (String s:a) {
             System.out.println("argomento "+
                   (++arg)+" : "+s);
         }
   }
}
```

Parametri di ingresso in Netbeans





Un esempio riassuntivo

Esempio: Tombola!

Tombola!

	17	22		45		66		83
1	19		30	48			70	
		29	37		53		74	86

Tombola

Il croupier(banco) ha a disposizione un tabellone sul quale sono riportati tutti i numeri da 1 a 90, e un sacchetto riempito con pezzi numerati in modo analogo.

Il suo compito consiste nell'estrarre i pezzi in modo casuale, e annunciare agli altri giocatori il numero uscito.

I giocatori dispongono di una o più cartelle precedentemente acquistate, composte da 3 righe, su ciascuna delle quali sono riportati cinque numeri compresi tra 1 e 90.

Ogni volta che il numero estratto è presente su una o più delle sue schede, il giocatore "copre" la casella corrispondente.

Lo scopo ultimo del gioco è quello di realizzare la tombola, ovvero arrivare per primi a coprire tutti i numeri presenti su una delle proprie cartelle.

Estratto da https://it.wikipedia.org/wiki/Tombola

Tombola

Il **croupier(banco)** ha a disposizione un **tabellone** sul quale sono riportati tutti i numeri da 1 a 90, e un **sacchetto** riempito con **pezzi** numerati in modo analogo.

Il suo compito consiste nell'**estrarre** i pezzi in modo casuale, e **annunciare** agli altri giocatori il **numero uscito**.

I **giocatori** dispongono di una o più **cartelle** precedentemente acquistate, composte da 3 **righe**, su ciascuna delle quali sono riportati cinque numeri compresi tra 1 e 90.

Ogni volta che il numero estratto è presente su una o più delle sue schede, il giocatore "**copre**" la casella corrispondente.

Lo scopo ultimo del gioco è quello di realizzare la tombola, ovvero **arrivare per primi a coprire** tutti i numeri presenti su una delle proprie cartelle.

Estratto da https://it.wikipedia.org/wiki/Tombola

Diagramma ad oggetti…