Linguaggi di Programmazione

a.a. 14/15

docente: Gabriele Fici

gabriele.fici@unipa.it

3 - Le Classi (parte I)

3 - Le Classi

- Le classi sono modelli per creare e manipolare oggetti.

  es. Ho una classe "Serbatoio" che mi permette di creare un oggetto di tipo serbatoio

- Un oggetto ha:
  - un'identità (quindi un suo codice identificativo e uno spazio in memoria)
  - uno stato (definito da *attributi*, che sono variabili conservate nell'oggetto)
    es. `int livello`
  - un comportamento (definito da operazioni, dette *metodi*, eseguiti dall'oggetto per cambiarne lo stato)
    es. `void rifornimento (int j)`

3 - Le Classi

- Ogni oggetto è istanza di una classe, che ne determina il tipo
- Una classe deve descrivere per i propri oggetti:
  - gli stati possibili (attraverso gli *attributi*)
  - i comportamenti possibili (attraverso i *metodi*)
- La sintassi per dichiarare i metodi è:

```
<tipoRitornato> <nomeMetodo>(<dichiarParametri>){
    <corpo>
    }
```

Esempio:

```
void rifornimento (int j) {
  livello += j;
}
```

3 - Le Classi

- Un nuovo oggetto della classe si costruisce attraverso il *metodo costruttore*, che è un metodo particolare avente lo stesso nome della classe, e non ha valore di ritorno
- Il costruttore di default (senza parametri) viene fornito dal compilatore, a meno che non si definiscano altri costruttori

Esempio:

```
public class Serbatoio {
...
   Serbatoio (){          // Metodo costruttore
      livello = 10;
   }
...
}
```

3 - Le Classi

Esempio di classe (file `Serbatoio.java`):

```
public class Serbatoio {
   int livello;                  // attributo

 Serbatoio (){                   // metodo costruttore: ha lo
   livello = 10;                 // stesso nome della classe
 }

 void rifornimento (int j) {          // metodo
   livello += j;
 }

 int getLivello () {                  // metodo
   return livello;
 }
}
```

3 - Le Classi

Convenzioni:

- I nomi delle classi iniziano per lettera Maiuscola

  es. `class Serbatoio`

- I nomi delle variabili, degli attributi e dei metodi (tranne il metodo costruttore) iniziano per lettera minuscola

  es. `int livello` `int rifornimento()`

- I nomi delle costanti sono in MAIUSCOLO, con eventuale carattere underscore

  es. `final int MAX_VALUE`

- Le parole composte si scrivono con iniziali maiuscole (notazione a cammello, in inglese *CamelCase*)

  es. `getLivello()` `ContoBancario`

3 - Le Classi

- Le variabili che compaiono nei metodi si chiamano *variabili locali* (tutte le altre variabili, ad es. gli attributi delle classi, si chiamano *variabili di istanza*)
- Le variabili locali scompaiono quando il metodo termina la propria esecuzione
- Le variabili locali devono sempre essere inizializzate esplicitamente (mentre le variabili di istanza vengono inizializzate a un valore predefinito: `null` per i riferimenti agli oggetti, `0` per i tipi numerici, e `false` per il tipo boolean)

Esempio:

```
public int faiQualcosa (){      // metodo
    int i = 10;     // i e' una variabile locale
    ...
    return i;
 }  // qui i non esiste più
```

3 - Le Classi

- In Java non si gestiscono direttamente i puntatori
- Ogni oggetto viene manipolato automaticamente attraverso un *riferimento*
- Un nuovo oggetto si crea usando l'operatore `new`, che restituisce un riferimento all'oggetto appena creato, invocando un costruttore
- Al momento della dichiarazione, il riferimento all'oggetto ha valore `null`

Esempio:

```
Serbatoio s;
s = new Serbatoio();
/* oppure direttamente:
 Serbatoio s = new Serbatoio();
 */
```

3 - Le Classi

- Gli oggetti vengono sempre acceduti tramite i loro riferimenti
- Se manca il riferimento, l'oggetto non è più accessibile.

Esempio:

```
Serbatoio s; // dichiarazione del riferimento
s = new Serbatoio(); //s punta all'oggetto creato
s = s1;  // l'oggetto creato non ha più riferimenti
```

3 - Le Classi

- Un metodo speciale è il metodo `main`, che rappresenta il punto di accesso al programma, e va dichiarato con una sintassi specifica (che sarà chiara in seguito):

```
public class Serbatoio {
 int livello;                          // attributi

 Serbatoio (){...}                     // metodi
 void rifornimento (int j) {...}
 void consumo (int j) {...}

 public static void main(String[] args){ // metodo main
   Serbatoio s = new Serbatoio();
   System.out.println("Il livello e' " + s.livello);
 }
}
```

3 - Le Classi

- Diversamente da altri linguaggi, non occorre deallocare manualmente la memoria
- Il "*Garbage Collector*" della JVM si occupa di liberare dalla memoria gli oggetti che non hanno più un riferimento
- La JVM gestisce un contatore dei riferimenti per ogni oggetto, quando il contatore è a zero l'oggetto viene deallocato automaticamente

Esempio:

```
Serbatoio s, s1;
s = new Serbatoio();
s1 = s;     // s1 e s puntano allo stesso oggetto
s1 = null;
s = null;  /* l'oggetto di tipo Serbatoio non è
più accessibile e sarà cancellato */
```

3 - Le Classi

- L'accesso agli attributi e ai metodi di un oggetto si effettua mediante la "notazione punto" (*dot notation*)

Esempio:

```
int j;
s = new Serbatoio();
j = s.livello;          //accesso all'attributo
s.rifornimento(10);     //accesso al metodo
```

3 - Le Classi

- I parametri dei metodi possono essere tipi fondamentali oppure oggetti
- Gli oggetti vengono passati sempre attraverso i loro riferimenti
- I tipi fondamentali vengono passati sempre per valore, cioè viene passato al metodo una copia della variabile, la quale, essendo locale, cesserà di esistere alla chiusura del metodo

Esempio:

```
public class MiaClasse {
  int attributo = 10;
  public void incrementa (int n) { n++; }
  public static void main (String[] args) {
    MiaClasse oggetto = new MiaClasse();
    oggetto.incrementa(oggetto.attributo);
    System.out.println(oggetto.attributo); // stampa 10!
  }
}
```

3 - Le Classi

- Un file `.java` può contenere più classi, ma al più una classe può essere `public`
- Se un file `.java` contiene una classe `public`, questa deve avere lo stesso nome del file
- Se più classi in uno stesso file contengono un metodo `main`, verrà eseguito quello della classe che ha lo stesso nome del file
- Se un programma contiene tante classi, può essere conveniente aggiungere una classe `Main` contenente il solo metodo `main`
- Se mettiamo diversi file `.java` nella stessa cartella, ogni classe potrà accedere agli attributi e ai metodi delle altre classi, a meno che questi non siano `private`

3 - Le Classi

Annotazioni per Javadoc

- I commenti per la Javadoc si fanno con `/** commento */`
- Si possono usare i tag HTML all'interno dei commenti
  es.

```
/** Questa classe vi permette di:
* <UL><LI>Inserire un libro con il metodo
<PRE>inserisci()</PRE>
* <LI>Rimuovere un libro
* <LI>Cercare un libro
* </UL>
* <B>Attenzione!</B> I libri sono da creare a
parte.
*/
```

3 - Le Classi

Javadoc

- Si genera la documentazione digitando:
  `javadoc [options] [packagenames] [sourcefiles] [classnames] [@files]`

- Tra le opzioni:
  `-d <directory>` Directory di destinazione dell'output
  `-public` Mostra solo membri e classi public
  `-protected` Mostra membri e classi public o protected (default)
  `-package` Mostra membri e classi public, protected e package
  `-private` Mostra tutti i membri e le classi

- es. `javadoc -d docs *.java`

3 - Le Classi

- I tags per la Javadoc sono:

  `@author <nome>` Inserisce il nome dell'autore
  `@see <nome classe>` Crea un link ad un'altra classe
  `@see <packages>.<nome classe>` Come prima
  `@see <nome classe>#<nome metodo>` Crea un link ad un metodo di un'altra classe
  `@version <info>` Inserisce informazioni sulla versione
  `@since <versione>` Dice a partire da che versione esiste questa classe/metodo/attributo
  `@param <nome parametro> <descrizione>` Permette di descrivere un parametro
  `@return <descrizione>` Descrive il valore di ritorno
  `@throws <eccezione> <descrizione>` Descrive un'eccezione di un metodo
  `@deprecated <descrizione>` Descrive il motivo della deprecation

3 - Le Classi

- Esempio:

```
/** Classe Serbatoio, modella un semplice serbatoio
 * @author      Gabriele Fici <gabriele.fici@unipa.it>
 * @version     0.2
 */
public class Serbatoio {
  private int livello;           // attributo livello
  private int capacita = 100;    // attributo capacita'

    /** metodo costruttore,
     *  setta il livello a 10 litri */
    public Serbatoio (){ livello = 10; }

    /** metodo rifornimanto
     *  @param l incrementa il livello di l litri */
    public void rifornimento (int l){ livello += l; }

    /** metodo getLivello
     *  @return livello del serbatoio */
    public int getLivello (){ return livello; }
...
```