Installazioni preliminari

- È uscita la versione 6u23 dell'SDK.
- Disponibile a:
  http://www.java.com/en/download/index.jsp

- Per installare Eclipse invece andare su:
  http://www.eclipse.org

Stile

- Sono elementi importanti
  - uso dei commenti
  - indentazione
  - ordine delle istruzioni
  - scelta dei nomi per le variabili
  - uso di alcuni costrutti più leggibili

Stile

- In particolare, cercare di:
  - usare identificatori significativi, formattare bene il codice e mettere commenti dove necessario
  - rendere possibile il riuso del codice
  - evitare di dichiarare troppe variabili e usare il più possibile i metodi
  - cercare di identificare le operazioni più comuni

Stile

- Java è case-sensitive
- I nomi sono composti da una sequenza di caratteri alfanumerici che deve iniziare con un carattere alfabetico
- Classi: **`Studente, BankAccount`**
- metodi: **`deposit(), getNome()`**
- variabili: **`balance, natoIl`**
- costanti: **`SKINNER, WITHDRAW_FEE`**

Stile

- Indentazione: quando si usano molte parentesi graffe innestate è importante avere indentato correttamente il codice per trovare “cosa si apre” e “cosa si chiude”

```
class Stile1 {
  //campi
  //costruttori
  //metodi
}
```

```
class Stile2
{ //campi
  //costruttori
  //metodi
}
```

Commenti

- I commenti hanno diverse finalità:
  - commenti documentazione, descrivono lo scopo di una classe, di un metodo, di un costruttore
  - commenti implementazione, descrivono dei dettagli realizzativi
  - commenti asserzione, descrivono proprietà che si verificano durante l'esecuzione del codice

Esempio: classe Punto

```
/** Un oggetto Punto rappresenta un punto
  in uno spazio bidimensionale */
class Punto {
  //campi
  //costruttori
  //metodi
}
```

- Per riempire questa classe dobbiamo avere presente come vogliamo progettarne lo **stato** e il **comportamento**.

Esempio: classe Punto

```
/** Un oggetto Punto rappresenta un punto
  in uno spazio bidimensionale */
class Punto {
  //campi
  private double x; //coordinata X
  private double y; //coordinata Y
  //costruttori
  public Punto(double x, double y)
  { this.x = x; this.y = y; }
  public Punto()
  { this(0.0,0.0); }
...
```

Esempio: classe Punto

- Vogliamo implementare i seguenti metodi:
  - metodi **`set`** e **`get`** per entrambe le coordinate
  - metodo **`trasla(offsetX,offsetY)`**
  - vogliamo anche (ri)scrivere i metodi **`toString()`**, **`equals()`** , **`compareTo()`**
- Vogliamo scrivere una classe che usa **`Punto`**. Il **`main`** di questa classe può:
  - definire un punto “origine” e altri scelti da noi
  - traslare i punti di valori arbitrari
  - stampare il risultato

Esempio: classe Punto

- Miglioriamo la classe Punto:
  - cambiamo il costruttore senza parametri in modo che crei un punto a caso, anziché l'origine
  - metodo **`distanza`** tra due punti, che ritorna la distanza euclidea
- Scriviamo un'altra classe che usa **`Punto`**. Il **`main`** di questa classe:
  - crea punti arbitrari e/o casuali
  - li stampa a video e ne calcola la distanza
  - chiede all'utente di traslarli e ricalcola

Esempio: classe Punto3D

- Ora estendiamo la classe **`Punto`** sfruttando l'ereditarietà:
  - creiamo una classe **`Punto3D`** che abbia anche una coordinata z
- Vanno anche introdotti alcuni metodi:
  - ovviamente mancheranno metodi **`getZ`** e **`setZ`**
  - metodo proiezioneXY per proiettare sul piano z=0
  - che dire del metodo **`distanza`**? e di **`trasla`**?

Esempio: classe Punto3D

- Scriviamo una classe che usa **`Punto3D`**.
  Il **`main`** di questa classe:
  - crea 3 punti arbitrari
  - ne calcola la distanza reciproca
  - calcola l'area del triangolo da essi individuato mediante la formula di Erone

$$Area = \sqrt{p(p-a)(p-b)(p-c)}$$

dove *p* è il semiperimetro

Javadoc

- Uno dei pregi di Java è quello di integrare la documentazione con il codice stesso
- Formato dei commenti:
  - **`/* commenti */`**
  - **`// commenti`**
  - **`/** commenti documentazione */`**
- Questi ultimi generano automaticamente la documentazione in formato HTML utilizzando un apposito strumento **`javadoc`**

Javadoc

- Ad esempio se abbiamo definito una classe **`Dado`** nel file **`Dado.java`** possiamo generare la documentazione con

  `javadoc Dado.java`

- Vengono generati i file di documentazione della classe a partire da un index.html che possono essere visualizzati con un browser

Javadoc

- All'interno della documentazione si possono usare tag HTML nel modo: **`<code> </code>`**
- Inoltre si possono usare specifici tag per comandi di **`javadoc`**. Questi comandi iniziano con il carattere `@`
- Block tags, presenti solo nella sezione dopo la descrizione principale: **`@tag`**
- Inline tags, presenti ovunque: **`{@tag}`**

@author

- Specifica il nome dell'autore.
- È possibile specificarne più d'uno
- Viene incluso nella documentazione solo se **`javadoc`** viene eseguito con l'opzione **`-author`**

Notazione grafica UML

- Rappresentazione in UML per l'ereditarietà.

- Attenzione: punta alla superclasse!

- Può essere utile usare UML per rappresentare graficamente l'interazione tra le classi.