Ingegneria del Software

Esercitazione 2

String Literals

Illustrare l'effetto delle istruzioni in rosso sullo Heap.
Che uguaglianza c'è tra le quattro String?

```
class Example1 {
  public static void main(String args[]) {
    String s1 = "abc";
    String s2 = "abc";
    String s3 = new String("abc");
    String s4 = s3;
  }
}
```

String Literals

Risposta:

`s1` *e* `s2` *puntano allo stesso oggetto stringa nello heap*

`s3` *punta ad un oggetto diverso nello heap, al quale punta anche* `s4`

```
                    s1.equals(s2)
s1 == s2            s3.equals(s4)
s3 == s4            s1.equals(s3)
                    …
```

Strings

Cosa stampa questo programma?

```
public class StringDemo {
 public static void main(String[] args){
  String s1 = “Guess who”;
  String s2 = s1;
  s1 = s1 + “ is back”;
  System.out.println(s1);
  System.out.println(s2);
 }
}
```

Strings are Immutable

Risposta:

> Guess who is back?

> Guess who

- `s2` punta a `s1`
- `s1` concatenato con un'altra stringa genera un nuovo oggetto
- `s2` continua a puntare all'oggetto precedente

Valutazione Parametri

Illustrare e motivare il valore delle variabili i, counter e counter 2 ad ogni riga del main

```
class Counter {

  int counter = 0;

  public void increment(){
      counter++;
  }

  public void incrementAndSet(int i){
      i++;
      counter=i;
  }

  public void incrementAndSet(Counter c){
       c.counter++;
       counter = c.counter
  }
}
```

```
public static void main(String args[]) {
    Counter counter = new Counter();
    counter.increment();
    int i = 3;
    counter.incrementAndSet(i);
    Counter counter2 = new Counter();
    counter2.incrementAndSet(i);
    counter.incrementAndSet(counter2);
  }
```

Valutazione Parametri

Tipi primitivi: passati per valore
Oggetti: passati per referenza

Risposta:

```
public static void main(String args[]) {
    Counter counter = new Counter();          counter = 0
    counter.increment();                      counter = 1
    int i = 3;                                i = 3
    counter.incrementAndSet(i);               i = 3, counter = 4
    Counter counter2 = new Counter();         i = 3, counter = 4, counter2 = 0
    counter2.incrementAndSet(i);              i = 3, counter = 4, counter2 = 4
    counter.incrementAndSet(counter2);        i = 3, counter = 5, counter2 = 5
}
```

Complex (1)

Definire una classe per la gestione di numeri complessi

Specifiche:

- Un numero complesso è identificato da due numeri *real* e *imaginary* di tipo double
- Dato un numero complesso $z = x + yi$, definire le operazioni di *modulo* e *fase*

$$r = |z| = \sqrt{x^2 + y^2}.$$

$$\varphi = \arg(z) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \arctan(\frac{y}{x}) + \pi & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan(\frac{y}{x}) - \pi & \text{if } x < 0 \text{ and } y < 0 \\ \frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \\ \text{indeterminate} & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

Complex (2)

Definire una classe per la gestione di numeri complessi

Specifiche:

- Dati due numeri complessi definire le operazioni di somma, sottrazione, moltiplicazione e uguaglianza.

$$(a + bi) + (c + di) = (a + c) + (b + d)i.$$

$$(a + bi) - (c + di) = (a - c) + (b - d)i.$$

$$(a + bi)(c + di) = (ac - bd) + (bc + ad)i.$$

$$z_1 = z_2 \;\leftrightarrow\; (\mathrm{Re}(z_1) = \mathrm{Re}(z_2) \wedge \mathrm{Im}(z_1) = \mathrm{Im}(z_2)).$$

Persons and Students

Definire le classi Person, Student e Grade

Specifiche:

- Una persona ha un nome, cognome e una data *(java.util.Date)*
- Uno studente è una persona con un id e una lista di voti
- Un voto contiene punteggio e crediti
- Lo studente espone due funzionalità:
  - Calcolo media pesata
  - Controllo se è possibile che si laurei (crediti totali >= 180)

Access Modifier

Completare il codice con gli opportuni modificatori di visibilità in modo tale che l'accesso alle variabili e ai metodi sia il più ristretto possibile ma che non crei errori di compilazione

Memento: Access Modifier

`public`	visibile da qualsiasi parte del programma
`private`	visibile solo dall'interno della classe stessa
`protected`	visibile solo dalle classi dello stesso package e dalle sottoclassi
`default`	visibile dallo stesso package e dalle sottoclassi se sono nello stesso pacchetto.

```
package a;
... class First {
  ... int x;
  ... int y;
  ... void h() { y = -1; }
}
... class Second extends First {
  ... void f(int x) { this.x = x; h(); }
}

package b;
imports a.*;
class Third {
  public static void main(String[] s) {
    Second z = new Second();
    z.f(3);
}

class Fourth extends First {   void g() { h(); }  }
```

```
package a;
public class First {
  int x;  // default
  private int y;
  protected void h() { y = -1; }
}
public class Second extends First {
  public void f(int x) { this.x = x; h(); }
}

package b;
imports a.*;
class Third {
  public static void main(String[] s) {
    Second z = new Second();
    z.f(3);
}
class Fourth extends First {   void g(void) { h(); }  }
```