Ingegneria del Software

Esercitazione 1

Contatto

Luca Terracciano
PhD Student @ POLIMI

Mail: **luca.terracciano@polimi.it**

Esercitazioni

*Inizio ore **9.15**, fine ore **12.30***

(pausa 10.45-11.00)

- Lettura e spiegazione delle specifiche

- ~10 minuti in cui ognuno di voi cercherà una propria soluzione
  - Su computer (meglio) o su carta

- Risoluzione degli esercizi, *live coding*

Esercitazione di oggi

*Inizio ore **9.15**, fine ore **13.30***

- Incontro con Acciaierie Arvedi (11.15-12.15)
- Lezione fino alle 11
- Riprendiamo a fine incontro

IDE

Integrated Development Environment:
un tool che ci permette di scrivere codice, compilarlo, eseguirlo e testarlo

IntelliJ IDEA
https://www.jetbrains.com/idea/download

Eclipse
https://www.eclipse.org/downloads

Codice

Il testo e il codice delle esercitazioni verranno caricati sulla repository Git del corso

https://github.com/lterrac/ingsw

Esercizi

Polygons

Definire la classe Point e la classe Polygon

Specifiche:

- Un punto è identificato dalle sue coordinate nel piano
- Un poligono è una successione di punti
- Deve essere possibile ottenere il perimetro di un poligono

Set

Definire la classe Set:
una collezione non ordinata di interi senza duplicati

Specifiche:

- Un Set può avere grandezza finita e arbitraria (default 100)
- Deve essere possibile controllare se un elemento è presente nel Set
- Deve essere possibile aggiungere e rimuovere elementi
- Deve essere possibile ottenere una rappresentazione testuale dell'insieme con la sintassi "{elem1, elem2, …}"

Stack

Definire una classe Stack di interi:
una struttura dati LIFO

Specifiche:

- Uno Stack ha una dimensione finita e arbitraria, default = 10
- Metodo *pop:* ottenere l'ultimo elemento aggiunto e rimuoverlo
- Metodo *push*: aggiungere un elemento

Gara Canora

Definire la classe Singer e Competition

Specifiche Singer:

- Un cantante ha un nome, cognome, luogo e data di nascita e un titolo di canzone
- Deve essere possibile comparare alfabeticamente i cantanti (per cognome e in caso di omonimia si usa il nome)
- Due cantanti sono uguali se tutti i loro dati coincidono

Gara Canora

Definire la classe Singer e Competition

Specifiche Competition:

- Una competizione è composta da una classifica di 10 cantanti
- Una competizione ha inoltre un nome, un luogo, un presentatore, un data di inizio e fine (tutti elementi stringa)
- Metodo ***addSinger(Singer singer)***: aggiunge un cantante alla classifica (in coda se non vuota)
- Metodo ***addSinger(Singer singer, int atPosition)***: aggiunge un cantante alla classifica alla posizione specificata, gli altri cantanti scalano (e di conseguenza uno può uscire dalla classifica)
- Metodo ***deleteSinger(Singer singer)***: elimina il cantante dalla competizione se esiste
- Metodo **deleteSinger(String surname, String name)**: elimina un cantante dalla competizione con nome e cognome specificato
- Metodo ***generateRank()***: ritorna la rappresentazione testuale della classifica
- Metodo ***generateOrderedList()***: ritorna una lista testuale alfabetica dei cantanti in classifica

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 == s2

s3 == s4

s1.equals(s2)

s3.equals(s4)

s1.equals(s3)

…

Valutazione Parametri

Illustrare e motivare il valore delle variabili i, counter e counter2 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.increment();
    int i = 3;
    counter.incrementAndSet(i);
    Counter counter2 = new Counter();
    counter2.incrementAndSet(i);
    counter.incrementAndSet(counter2);
}
```

counter = 0

counter = 1

i = 3

i = 3, counter = 4

i = 3, counter = 4, counter2 = 0

i = 3, counter = 4, counter2 = 4

i = 3, counter = 5, counter2 = 5

Linguaggio Procedurale vs OOP

Consideriamo una ipotetica implementazione in C, o in un qualsiasi linguaggio procedurale, di un tipo di dato qualsiasi, ad esempio la seguente per i punti nello spazio. Quale differenza sostanziale esiste con un linguaggio ad oggetti come Java?

```
/* un nuovo tipo per la struttura dati */
typedef struct { float x,y; } *planepoint;


/* le operazioni che manipolano le variabili di tipo planepoint */
void init(planepoint *this, float x, float y) { this->x =x; this->y = y; }

void translate(planepoint *this, float deltax, float deltay) ...

float distance(planepoint *this, planepoint *another) ...
```

Linguaggio Procedurale vs OOP

In un linguaggio procedurale le operazioni agiscono su un tipo hanno almeno un parametro di tale tipo. In Java le dichiarazioni delle **funzioni** *vengano inserite nella dichiarazione del* **tipo** *e per tanto si chiamano* **metodi**.

```
/* dichiarazione del tipo E delle operazioni che manipolano le sue   variabili */
class PlanePoint {
  /* i dati di un oggetto di tipo PlanePoint */
  float x,y;


  /* le operazioni che lo manipolano */
  PlanePoint(float x, float y) { this.x = x; this.y = y; }
  void translate(float deltax, float deltay) ...
  float distance(PlanePoint another) ...
}
```

Linguaggio Procedurale vs OOP

In altre parole:

- Il parametro this viene automaticamente passato da Java alle tre operazioni, e quindi non va dichiarato (non c'e' bisogno!)

- L'operazione che inizializza un nuovo oggetto di un tipo, il costruttore, viene automaticamente invocata da Java non appena viene creato l'oggetto manentendolo sempre in uno stato consistente.

Complex (Parte 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 (Parte 2)

Definire una classe per la gestione di numeri complessi

Specifiche:

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

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

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

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

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

```
class complesso {
    private double reale, immaginaria;

    public complesso(double i,double j)  {
        reale=i;
        immaginaria=j;
    }


   public double reale()  {

        return reale;

    }


    public double immaginaria()  {

        return immaginaria;

    }
```

```
public double modulo()  {
    return Math.sqrt(this.reale*this.reale+
                this.immaginaria*this.immaginaria);
}

public double fase()    {
    if(this.reale==0)
        return Math.PI/2;//se reale=0 fase=90 gradi
    else
        return
         Math.atan(this.immaginaria/this.reale);
            //se non conoscete atan non importa ;
}
```

```
public complesso somma(complesso altro)  {
   return new complesso(this.reale+altro.reale,
                  this.immaginaria+altro.immaginaria);
}

public complesso differenza(complesso altro)  {
   return new complesso(this.reale-altro.reale,
                  this.immaginaria-altro.immaginaria);
}

public complesso prodotto(complesso altro)  {
   return new complesso(this.reale*altro.reale-
                  this.immaginaria*altro.immaginaria,
                  this.reale*altro.immaginaria+
                  this.immaginaria*altro.reale);
}
```

double e uguaglianza

- In genere, per problemi con l'approssimazione nei numeri a virgola mobile, è meglio introdurre anche una uguaglianza approssimata.

```
public boolean approximatelyEquals(Complex other, double precision) {
        // controlla che la differenza in valore assoluto
        // tra i due numeri complessi sia minore di precision
        return (this.diff(other).abs() < precision);
}
```

```
    public String toString() {
       return ""+reale+"+"+immaginaria+"i";
    }
    public static void main(String args[])  {
       complesso a=new complesso(5,6);
       complesso b=new complesso(4,0);
       complesso c=a.somma(b);
       complesso d=a.differenza(b);
       complesso e=a.prodotto(b);
       double m=a.modulo();
       double f=a.fase();
       System.out.println(a);
       System.out.println(b);
       System.out.println(c);
       System.out.println(d);
       System.out.println(e);
       System.out.println(m);
       System.out.println(f);
    }
}
```

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);
}
```

```
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) {
```