Ingegneria del Software

Esercitazione 1

Contatto

Giovanni Quattrocchi
Ph.D. student, Politecnico di Milano

Mail: **giovanni.quattrocchi@polimi.it**

Esercitazioni

*Inizio ore **14.00**, fine ore **17.15***
(pausa 15.30-15.45)

Per ogni esercizio:

- Lettura e spiegazione delle specifiche
- 10/15 minuti in cui ognuno di voi cercherà una propria soluzione
  - Su computer (meglio) o su carta
- Risoluzione degli esercizi, *live coding*

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 verrano caricati sulla repository Git del corso

https://github.com/gioenn/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

Specifiche:

- Un Set può avere grandezza finita e arbitraria (default 100)
- 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
- Metodo *push*: aggiungere un elemento

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)
                    …
```

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
}
```

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.