Corso di Laurea in Informatica
Prova Scritta di Metodologie di Programmazione - Primo Canale

Università di Roma "La Sapienza"

7 Settembre 2023

Durante l'esame non è consentito l'utilizzo di alcunché. Non è consentito inoltre l'utilizzo della matita o di penne il cui colore sia diverso dal nero o dal blu. Il ritiro dalla prova equivale al mancato superamento dell'esame.

Nome e Cognome:

Matricola:

Domanda	1	2	3	4	5	Totale
Punteggio Totale	10	7	4	7	13	31
Punteggio Ottenuto						

1. 10 punti Per ogni domanda, indicare con una X la risposta desiderata. Si ricorda che ogni domanda ha al più una risposta corretta. L'assegnazione dei punti alle risposte è la seguente: verranno attribuiti 2 punti per ogni risposta esatta, -0.75 punti per ogni risposta errata, 0 punti per ogni risposta omessa. Al fine del superamento della soglia è necessario totalizzare un punteggio di almeno 5 punti.
   (a) Qual è il principio della programmazione orientata agli oggetti che afferma che una classe dovrebbe avere una sola ragione per essere modificata?
   ◯ Principio di incapsulamento ◯ Principio di ereditarietà ◯ Principio di polimorfismo ☒ Principio di single responsibility
   (b) Quale delle seguenti dichiarazioni di una variabile in Java è corretta?
   ◯ int x = "Hello"; ☒ double y = 3.14; ◯ String z = true; ☒ String w = "World"; SONO ENTRAMBE GIUSTE
   (c) In un ciclo "for" in Java, quale delle seguenti parti è opzionale?
   ◯ Inizializzazione ◯ Condizione ◯ Incremento ☒ Tutte le parti sono obbligatorie
   (d) Quale principio SOLID prevede che una classe debba essere chiuse alle modifiche ma aperta alle estensioni?
   ◯ DIP ◯ LSP ◯ SRP ☒ OCP
   (e) Il metodo `public static Class forName(String className)`: ☒ Restituisce l'oggetto Class che rappresenta la classe dal nome className ◯ Non restituisce alcun valore ◯ Restituisce un oggetto Object

2. [7 punti] Qual è la differenza tra overloading e overriding? Fornire un esempio minimale, scritto in Java, che descriva quanto richiesto.

3. [4 punti] Per ogni costrutto iterativo, indicare il numero di volte per il quale viene eseguito il suo corpo. Se non diversamente espresso, si assume che la variabile contatore non venga modificata all'interno del corpo di ciascun costrutto iterativo.
   (a) `for (int i = 0; i < 4; i++){...}`
   (b) `for (int i = 9; i >= 0; i -= 2){...}`
   (c) `for (int i = 16; i > 1; i /= 2){...}`
   (d) `for (int i = 1; i <= 64; i *= 2){...}`
   (e) `for (int i = 0; i <= 20; i += 5){...}`
   (f) `for (int i = 10; i >= 0; i -= 1){...}`

4. [7 punti] Spiegare le principali differenze tra ArrayList e Set in Java.

5. [13 punti] Un supermercato ha accesso ad un database contenente tutte le informazioni sui propri prodotti. Tali informazioni sono rappresentate in formato tabellare, in cui ciascuna riga contiene i seguenti campi separati da uno spazio:

   codiceProdotto1 NomeProdotto1 Quantità1 PrezzoPerUnità1
   codiceProdotto2 NomeProdotto2 Quantità2 PrezzoPerUnità2
   ...

   Scrivere un programma che legga un file di testo con questa struttura, segnalando un opportuno errore in caso di file inesistente. Successivamente, visualizzare:

   - Le informazioni relative al prodotto più costoso;
   - Il prezzo medio per ciascun prodotto, calcolato come il rapporto tra prezzo per unità e la quantità disponibile. Gestire gli eventuali casi speciali
   - I nomi dei prodotti esauriti, ossia con una quantità pari a zero

1)
a) PRINCIPIO DI SINGLE RESPONSABILITY
b) double = 3.14
c) TUTTE LE PARTI SONO OBBLIGATORIE
d) OCP
e) RESTITUISCE L'OGGETTO Class CHE RAPPRESENTA LA CLASSE DAL NOME className

3) a $\frac{4-0}{1} = 4$

2) OVERLOADING E OVERRIDING SONO 2 CONCETTI FONDAMENTALI DI POLIMORFISMO.

PARLIAMO DI OVERLOADING QUANDO VENGONO CREATI PIÚ COSTRUTTORI CON LO STESSO NOME MA CHE ACCETTANO ARGOMENTI DIVERSI. DI SEGUITO UN ESEMPIO:

```
public sommaNumeri(int numero1, int numero2){
    numero1 = this.numero1;
    numero2 = this.numero2;
    somma = numero1 + numero2;
}

public sommaNumeri(double numero1, double numero2){
    numero1 = this.numero1;
    numero2 = this.numero2;
    somma = numero1 + numero2;
}
```

QUESTO CODICE PERMETTE DI ACCETTARE SIA DEGLI INT CHE DEI DOUBLE, SENZA CAUSARE UN ERRORE

PARLIAMO DI OVERRIDING QUANDO ABBIAMO UNA CLASSE FIGLIA CHE, EREDITANDO UN METODO DA UNA CLASSE MADRE, LO "SOVRASCRIVE", INSERENDO UN COMANDO PIÚ SPECIFICO PER QUELLA:

```
public class Animale{
  public void faiVerso(){
  System.out.println("Verso di un animale");
  }
}
public class Cane extends Animale{
  @Override
    public void faiVerso(){
      System.out.println("Bau bau!");
    }
}
```

IN QUESTO CODICE, SE SI CREA UN OGGETTO CANE:
Animale animale = new Cane() E SI INVOCA IL METODO FAIVERSO, IL VERSO NON SARÀ GENERICO MA QUELLO SPECIFICO DELLA CLASSE

3) $a = \frac{4-0}{1} = 4$ $b = \frac{0-8}{-2} + 1 = 4 + 1 = 5$ $c = \log_2 16 = 4$ $d = \log_2 64 + 1 = 7$ $e = \frac{20-0}{5} + 1 = 5$ $f = \frac{0-10}{-1} + 1 = 11$

4) LA PRIMA DIFFERENZA SOSTANZIALE È CHE GLI ARRAYLIST DERIVANO DALL'INTERFACCIA "LIST", MENTRE I SET DALL'OMONIMA INTERFACCIA. UN ELEMENTO PUÓ ESSERE INSERITO PIÚ VOLTE IN UN ARRAYLIST, MENTRE I SET MEMORIZZANO OGNI ELEMENTO UNA SOLA VOLTA. UN ELEMENTO IN UN ARRAYLIST PUÓ ESSERE CERCATO MOLTO VELOCEMENTE TRAMITE IL SUO INDICE, MENTRE I SET NON SONO ORDINATI QUINDI BISOGNA ITERARE SU TUTTO QUANTO UTILIZZANDO IL METODO CONTAINS().

5)
```
public class Product{
  private String name;
  private String categ;
  private float price;
  public Product(String name, String categ, float price){
  this.name=name;
  this.categ=categ;
  this.price=price;
  }
  public String getName(){
   return name;
  }
  public String getCateg(){
   return categ;
  }
  public float getPrice(){
   return price;
  }
```

```
public class Shop{
  private ArrayList <Product> products;
    public Shop(ArrayList <Product> products){
     this.products=products;
    }
    public String listS(String prod){
      String string="";
      for(Product element: products){
        if(element.getCateg().equals(prod){
          string+=element.getName()+" "+element.getPrice()+"\n";
        }
      }
      return string;
    }
    public float midPP(String prod){
       float avg=0;
       int counter=0;
        for(Product element: products){
           if element.getCateg().equals(prod){
             counter++;
             avg+=element.getPrice();
           }
        }
        return (avg/=counter);
    }
    public String printMax(){
      float max=0;
      for(Product element: products){
         if(element.getPrice()>max){
           max=element.getName();
         }
      }
      return String.valueOf(max);
    }
```

```
puldic class Main{
  public static void main(String[] args){
    Scanner reader;
    try{
        reader=new Scanner(new File("products.txt"));
    }catch(FileNotFoundException exc){
      System.out.println("file not found");
      return;
    }
    ArrayList <Product> products=new ArrayList<>();
    String[] line
    while(reader.hasNextLine()){
      line=reader.nextLine.split();
      products.add(new Product(line[0],line[1], Float.parseFloat(line[2]));
    }
     Shop shop=new Shop(products);
     System.out.println(shop.listS("Shoes"));
     System.out.println(shop.midPP("Paints"));
     System.out.println(shop.searchMax());

  }
}
```