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

Sapienza Università di Roma

13 Giugno 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	6	Totale
Punteggio Totale	10	4	3	5	8	12	32
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) Quale dei seguenti è un esempio corretto di creazione di una sottoclasse in Java? (X) class MyFrame extends Jframe ( ) private interface JFrame implements MyFrame ( ) public MyFrame implements Jframe
   (b) Lo statement `"new String[] {"Metodologie", "Di", "Programmazione"};"`: ( ) Non è sintatticamente corretto (X) Crea un array anonimo con tre elementi di tipo String ( ) Crea una array chiamato String con tre elementi
   (c) L'uso dell'istruzione "break" in un blocco "switch": ( ) Permette di continuare l'esecuzione del blocco (X) Interrompe l'esecuzione del blocco ( ) Nessuna delle precedenti
   (d) Le *code convention* in Java prevedono che: ( ) Il nome delle classi cominci con lettera minuscola, mentre quello dei metodi con la maiuscola (X) Il nome delle classi cominci con lettera maiuscola, mentre quello dei metodi con la minuscola ( ) Non esiste alcuna convenzione a riguardo
   (e) Il metodo `public static Class forName(String className)`: (X) Restituisce l'oggetto Class che rappresenta la classe dal nome className ( ) Non restituisce alcun valore ( ) Restituisce un oggetto Object

2. [4 punti] Qual è la differenza tra eccezione controllata e non controllata? Fornire un esempio per ognuno dei due tipi e una porzione di codice, scritta in Java, che mostri il meccanismo di gestione di uno di essi.

3. [3 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 = 1; i <= 5; i++){...}`

   (b) `for (int i = 10; i >= 0; i -= 2){...}`

   (c) `for (int i = 100; i > 0; i /= 2){...}`

   (d) `for (int i = 2; i < 1000; i *= 2){...}`

   (e) `for (int i = 0; i <= 100; i += 10){...}`

   (f) `for (int i = 50; i >= -10; i -= 5){...}`

4. [5 punti] Spiegare le principali differenze tra LinkedList e ArrayList in Java, evidenziando gli scenari in cui è meglio utilizzare ognuna delle due classi.

5. [8 punti] La classe *BankAccount* gestisce le informazioni relative ai conti bancari. Il conto viene creato con un saldo iniziale e consente operazioni di prelievo e deposito. Nel caso in cui il saldo del conto superi una una determinata soglia, il sistema applica un interesse al saldo attuale. Progettare i seguenti metodi:

   ```
   public void deposit(double amount) // effettua un deposito
   public boolean withdraw(double amount) // effettua un prelievo, restituisce true se
   il prelievo ha avuto successo, altrimenti false
   public void applyInterest(double interestRate) // applica l'interesse all'importo presente
   sull'account se il saldo supera una soglia specifica
   ```

   Realizzare un programma di collaudo (test) per verificare la correttezza dell'implementazione. Il programma dovrà creare un conto con un saldo iniziale di \$500, effettuare un prelievo di \$200, un deposito di \$100 e applicare un interesse del 5% all'importo presente sul conto se il saldo supera \$1000.

6. [12 punti] Un negozio di abbigliamento tiene traccia delle informazioni sui propri prodotti attraverso un file di testo. Ciascuna riga del file contiene i seguenti campi separati da una virgola:

   nomeProdotto1,categoria1,prezzo1
   nomeProdotto2,categoria2,prezzo2
   ...

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

   - La lista dei prodotti nella categoria *Shoes*;
   - Il prezzo medio dei prodotti nella categoria *Pants*;
   - Il nome del prodotto più costoso.

   Scrivere un programma di collaudo (test) che utilizzi le classi create in precedenza. Si assume che il formato di ciascuna riga sia sempre corretto.

1)

a) class MyFrame extends JFrame

b) CREA UN ARRAY ANONIMO CON TRE ELEMENTI DI TIPO STRING

c) INTERROMPE L'ESECUZIONE DEL BLOCCO

d) IL NOME DELLE CLASSI COMINCI CON LA LETTERA MAIUSCOLA, MENTRE QUELLO DEI METODI CON LA MINUSCOLA

e) RESTITUISCE L'OGGETTO Class CHE RAPPRESENTA LA CLASSE DAL NOME className

2) LA GESTIONE DELLE ECCEZIONI È UN METODO PER AFFRONTARE E GESTIRE GLI ERRORI CHE POSSONO VERIFICARE DURANTE L'ESECUZIONE DI UN

LE ECCEZIONI CONTROLLATE SONO ECCEZIONI CHE VENGONO INDIVIDUATE DURANTE LA COMPILAZIONE E CHE LO SVILUPPATORE È OBBLIGATO A GESTIRE.

PER GESTIRLA, È POSSIBILE USARE LE ISTRUZIONI TRY-CATCH. NEL TRY SCRIVO IL NORMALE FLUSSO DEL CODICE. NEL CATCH INVECE SCRIVO COSA FARE IN CASO DI ERRORE.

```
try{
    reader= new Scanner(new File("readme.txt");
}catch(FileNotFoundException exc){
 System.out.println("File not found");
 return;
}
```

COSÌ, SE IL FILE NON VIENE TROVATO IL PROGRAMMA SI CHIUDE CON UN MESSAGGIO ANZICHÉ CRAGHARE.

LE ECCEZIONI NON CONTROLLATE DERIVANO DALLA CLASSE RuntimeException O DALLE SUE SOTTOCLASSI, SONO ECCEZIONI CHE NON RICHIEDONO UNA GESTIONE ESPLICITA DA PARTE DEL PROGRAMMATORE, PERCHÉ SI VERIFICANO SOLO CON ALCUNE INTERAZIONI DELL'UTENTE

(IPOTIZZIAMO VENGA INSERITO DALL'UTENTE)

```
public void sqrt(double num){
 squareRoot=Math.sqrt(num);
 System.out.println("square root:" + squareRoot);
}
```

QUESTO CODICE NORMALMENTE FUNZIONA. SE PERÒ L'UTENTE INSERISCE UN NUMERO NEGATIVO, VIENE LANCIATA L' IllegalArgumentException (NON SI PUÓ FARE LA RADICE DI UN NUMERO NEGATIVO). PER CUI È CONSIGLIATO (MA NON OBBLIGATORIO) AL PROGRAMMATORE DI GESTIRE QUESTI CASI ECCEZIONALI.

3)

a = 5-1+1 = 5

b = $\frac{0-10}{-2}$ + 1 = 6

LEMMA DI CINIERI-ALABRETTI

c = $\log_{\frac{1}{2}} 100$ + 1 = 7 (APPROSSIMATO PER DIFETTO; VALORE PER CUI AMBO OGM VOLTA)

100 → 50 → 25 → 12 → 6 → 3 → 1

I COROLLARIO DEL LEMMA

d) $\log_2 1000 - \log_2 \frac{m}{2}$ (APPROSSIMATO PER DIFETTO; PERCHÉ MEZZO DA 2)

e = $\frac{100-0}{10}$ + 1 = 11

f = $\frac{-10-50}{-5}$ + 1 = 13

5)

```
public class BankAccount{
    private double balance;
    public BankAccount(double balance){
     this.balance = balance;
    }
    public void deposit(double amount){
      balance+= amount;
    }
    public boolean withdraw(double amount){
      if (balance >= amount){
          balance -= amount;
          return true;
      }
      return false;
    }
    public void applyInterest(double interestRate){
      balance+=(balance* (interestRate/100));
    }
    public void getBalance{
      balance;
    }
}
```

```
Public class Test{
  public static void main (String[] args){
     BankAccount bankAccount = new BankAccount(500);
      System.out.println(bankAccount.getBalance());
     bankAccount.withdraw(200);
     System.out.println(bankAccount.getBalance());
     bankAccount.deposit(100);
     System.out.println(bankAccount.getBalance());
     if(bankAccount.getBalance()>1000){
       bankAccount.addInterest(5);
     }
     System.out.println(bankAccount.getBalance());
  }
}
```

6)
```
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);
      }
```

```
public 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("Pants"));
    System.out.println(shop.searchMax());
  }
}
```