Corso di Laurea in Informatica

Prova Scritta di Metodologie di Programmazione - Primo Canale

Sapienza Università di Roma

13 Giugno 2023

SOGLIA:

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? ☒ 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 ○ 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 ☒ 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 ☒ 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)`: ○ 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.

UN ECCEZIONE È DETTA "CONTROLLATA" QUANDO DIPENDE DA FENOMENI ESTERNI CHE IL PROGRAMMATORE NON PUO' CONTROLLARE, CHE DEVE QUINDI "GESTIRE" CON I COSTRUTTI throws OPPURE try/catch. UN ESEMPIO DI ECCEZIONE CONTROLLATA È LA FileNotFoundException.

```
File f = new File("test.txt");
Scanner in;
try{ in = new Scanner(f);} // CI SI ASPETTA UN ECCEZIONE
catch(FileNotFoundException e){ return;}
```

UN ECCEZIONE "NON CONTROLLATA" È DOVUTA DALL'ERRORE DEL PROGRAMMATORE, CHE HA SCRITTO UN CODICE CAPACE DI GENERARE UN ECCEZIONE SENZA RENDERSENE CONTO. DI QUESTA CATEGORIA FANNO PARTE LE RuntimeException.

```
int[ ] list = new int[4];
For(int i = 0; i ≤ 4; i++){
    System.out.println(list[i]); // GENERERA' UN ArrayIndexOutOfBounds
}
```

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 de corpo di ciascun costrutto iterativo.

(a) `for (int i = 1; i <= 5; i++){...}` 5

(b) `for (int i = 10; i >= 0; i -= 2){...}` 6

(c) `for (int i = 100; i > 0; i /= 2){...}` 7

(d) `for (int i = 2; i < 1000; i *= 2){...}` 9

(e) `for (int i = 0; i <= 100; i += 10){...}` 11

(f) `for (int i = 50; i >= -10; i -= 5){...}` 13

(c): 100 → 50 → 25 → 12 → 6 → 3 → 1

(d): 2 → 4 → 8 → 16 → 32 → 64 → 128 → 256 → 512

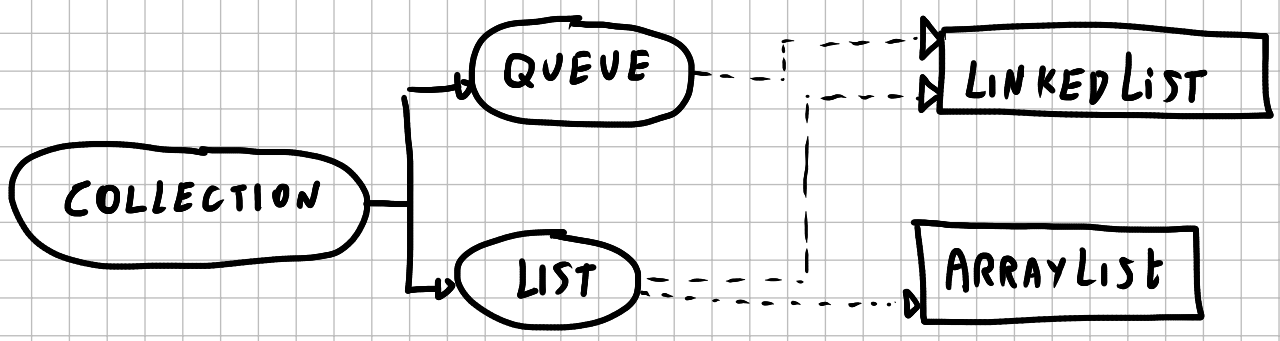
(e) 0 → 10 → 20 → 30 → 40 → 50 → 60 → 70 → 80 → 90 → 100

(f) 50 → 45 → 40 → 35 → 30 → 25 → 20 → 15 → 10 → 5 → 0 → -5 → -10

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.

ENTRAMBE LE CLASSI IMPLEMENTANO L'INTERFACCIA List, CHE A SUA VOLTA DERIVA DA Collection, DESCRIVONO UN INSIEME DINAMICO DI DATI CHE AMMETTE DUPLICATI, FANNO ENTRAMBE USO DI GENERICS.

- LinkedList È IMPLEMENTATA COME UNA LISTA DOPPIAMENTE PUNTATA, OGNI NODO HA IL RIFERIMENTO AL SUO SUCCESSORE E AL SUO PREDECESSORE. GARANTISCE QUINDI UNA MAGGIORE EFFICENZA NELLE OPERAZIONI DI INSERIMENTO E RIMOZIONE ED ACCEDE AGLI ELEMENTI SEQUENZIALMENTE. PUO' ESSERE USATA PER ISTANZIARE UNA Queue.
- ArrayList È IMPLEMENTATA CON UNA MATRICE DINAMICA, NON PUO' ESSERE USATA PER SIMULARE UNA CODA MA VANTA DI ACCESSO CASUALE AI SUOI ELEMENTI, RENDENDO LA NOTEVOLMENTE PIU' EFFICENTE SE CI È NECESSARIO ACCEDERE A DIVERSI ELEMENTI IN MODO RANDOMICO.



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.

```
public class BankAccount{
    private double money;
    public BankAccount(double m){ money = m;}
    public void deposit(double amount){ money += amount;}
    public boolean withdraw(double amount){
            if(money < amount){ return false;}
            money -= amount;
            return true;
    }
    public void applyInterest(double interestRate){ money += (money*(interestRate/100.0));}
    public double getMoney(){return money;}
    public void printInfo(){System.out.println("Saldo attuale " + money + " €.");}
}
public class Test{
    public static void main(String[] args){
        BankAccount b = new BankAccount(500.0);
        b.printInfo();
        if(b.withdraw(200.0)){System.out.println("prelievo eseguito.");
        else{System.out.println("saldo insufficente.");
        b.printInfo();
        b.deposit(100.0);
        b.printInfo();
        if(b.getMoney()>1000.0){b.applyInterest(5.0);}
        b.printInfo();
    }
}
```

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.

```
public class Product{
    public String name;
    public String cat;
    public float price;
    public Product(String[] data){
            name = data[0];
            cat = data[1];
            price = Float.parseFloat(data[2]);
    }
}
```

```
import java.io.*; import java.util.*;
public class Store{
        private ArrayList<Product> list = new ArrayList<>();
        private Product exp=new Product("x,x,-1.0".split(","));
        public Store(Scanner in){ // SI PASSA LO SCANNER COL FILE
                while(in.hasNextLine()){
                        Product tmp = new Product(in.nextLine().split(","));
                        list.add(tmp);
                        if(tmp.price > exp.price){ exp =tmp;}
                }
        }
        public void printMostExpensive(){System.out.println("prodotto piu' costoso: " + exp.name);}
        public float avgPrice(String cat){
                float n = 0;
                float sum = 0;
                for(Product p : list){
                        if(p.cat.equals(cat)){
                                n++;
                                sum += p.price;
                        }
                }
                return sum/n;
        }
        public void listItem(String cat){
                System.out.println("lista olei prodotti di categoria " + cat + ":");
                for(Product p:list){if(p.cat.equals(cat)){System.out.println(p.name);}}
        }
}
import java.io.*;
public class Test{
        public static void main(String[] args){
                Scanner in;
                try{in = new Scanner(new File("list.txt"));}
                catch(FileNotFoundException e){
                        e.printStackTrace();
                        return;
                }
                Store s= new Store(in);
                s.listItem("Shoes");
                System.out.println("prezzo medio Pants:" + String.valueOf(s.avgPrice("Pants")));
                s.printMostExpensive();
        }
}
```