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

Sapienza Università di Roma

13 Luglio 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	0	4	3	5	8	12	32
Punteggio Ottenuto							

1. [0 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 un oggetto in Java utilizzando il costruttore di una classe?
   ⊗ `MyClass obj = new MyClass();` ◯ `MyClass obj = createObject();` ◯ `MyClass obj = MyClass.create();`
   (b) Qual è il risultato dell'esecuzione del seguente codice Java?
   int x = 5; int y = x++; System.out.println(x + ", " + y);
   ◯ 5, 5 ◯ 5, 6 ⊗ 6, 5
   (c) Qual è l'output del seguente codice Java?
   String str = "Hello"; str.concat(" World"); System.out.println(str);
   ◯ Hello World ◯ World Hello ⊗ Hello
   (d) Qual è il modificatore di accesso di default per i membri di una classe in Java? ◯ public ◯ private ◯ protected ⊗ nessuno dei precedenti

(e) Qual è la differenza tra un'interfaccia e una classe astratta in Java? ◯ Un'interfaccia può implementare metodi, mentre una classe astratta no. ◯ Una classe astratta può essere istanziata, mentre un'interfaccia no. ☒ Una classe può implementare più interfacce, ma può estendere solo una classe astratta.

2. [4 punti] Descrivi il concetto di Ereditarietà e fornisci un esempio di utilizzo minimale scritto in Java.

3. [3 punti] Qual è la differenza tra il passaggio di parametri per valore e il passaggio di parametri per riferimento in Java? Fornisci un esempio minimale, scritto in Java, per ognuno dei due casi.

4. [5 punti] Descrivi brevemente il *Single Responsibility Principle (SRP)* appartenente ai principi SOLID. A seguire, fornisci un esempio minimale di tale principio, scritto in Java.

5. [8 punti] Si desidera creare un sistema per gestire una libreria online con le seguenti classi e interfacce:

   - **Pubblicazione**: un'interfaccia che definisce il metodo calcolaPrezzo() per calcolare il prezzo di una pubblicazione
   - **Libro**: una classe che rappresenta un libro con gli attributi titolo, autore e prezzo. La classe deve implementare l'interfaccia Pubblicazione
   - **Rivista**: una classe che rappresenta una rivista con gli attributi titolo, editore e numero. La classe deve implementare l'interfaccia Pubblicazione
   - **LibreriaOnline**: una classe che gestisce una collezione di pubblicazioni. Deve avere un metodo aggiungiPubblicazione(Pubblicazione pubblicazione) per aggiungere una pubblicazione alla libreria e un metodo calcolaPrezzoTotale() per calcolare il prezzo totale di tutte le pubblicazioni nella libreria.

   Scrivi un programma in Java che implementi quanto richiesto e crea un programma di collaudo per verificarne il funzionamento.

6. [12 punti] Scrivere un programma in Java il quale, dato uno stack, rimuova i duplicati al suo interno.

   Nota Bene: è sconsigliata la lavorazione diretta sullo stack di partenza.

1)

a) `MyClass obj = new MyClass;`

b) 6,5

c) Hello

d) NESSUNO DEI PRECEDENTI

e) UNA CLASSE PUÓ IMPLEMENTARE PIÚ INTERFACCE, MA PUÓ ESTENDERE SOLO UNA CLASSE ASTRATTA.

2) L'EREDITARIETA' É UN CONCETTO MOLTO IMPORTANTE ALL'INTERNO DELLA PROGRAMMAZIONE AD OGGETTI. EL AVVIENE TRA DUE O PIÚ CLASSI. LA PRIMA É CHIAMATA CLASSE PRIMITIVA (O SUPERCLASSE), LE SECONDE INVECE SONO DETTE CLASSI DERIVATE (O SOTTOCLASSI). IL MECCANISMO DELL'EREDITARIETA' CONSENTE ALLE SOTTOCLASSI DI EREDITARE GLI ATTRIBUTI E I METODI DELLA SUPERCLASSE, QUINDI DI POTERLI UTILIZZARE ANCHE SE NON DICHIARATI ALL'INTERNO DI ESSA (ATTRAVERSO LA PAROLA CHIAVE "SUPER") OPPURE MODIFICARLI E SOVRASCRIVERLI (ATTRAVERSO L'OVERRIDE). I MODIFICATORI DI VISIBILITA' POSSONO ESSERE DI 4 TIPI: PUBLIC, PROTECTED, DEFAULT E PRIVATE. I CAMPI CONTRASSEGNATI CON "PRIVATE" RIMARRANNO NELLA CLASSE BASE, MENTRE CON GLI ALTRI MODIFICATORI VERRANNO UTILIZZATI (N.B. DEFAULT SOLO SE SI TROVANO NELLO STESSO PKG. DEFAULT = "PACKAGE PROTECTED")

```
public class Vehicle{
  private String plate;
  public Vehicle(String stringa){
    this.plate = stringa;
  }
  public void clacson(){
    System.out.println("generic clacson");
  }
}
```

```
public class Car extends Vehicle{
  public Car(String plate, int wheels){
    super(plate);
    this.wheels = wheels;
  }
  public void clacson(){
  @Override
  System.out.println("car clacson");
  }
}
```

"NO, APPLICAZIONI", C.I.T.

3) PASSAGGIO PER RIFERIMENTO E PER VALORE SONO DUE TIPI DI PASSAGGIO DI DATI: LA DIFFERENZA PRINCIPALE TRA I 2 É CHE IL PASSAGGIO PER VALORE VIENE APPLICATO AI TIPI PRIMITIVI (int, byte, boolean, char, etc.), MENTRE IL PASSAGGIO PER RIFERIMENTO VIENE APPLICATO AI TIPI NON PRIMITIVI (String, ArrayList, HashMap, etc.). INOLTRE, SE UN DATO VIENE PASSATO AD UNA FUNZIONE, SE VIENE PASSATO PER VALORE NON PUÓ ESSERE MODIFICATO, MENTRE SE VIENE PASSATO PER RIFERIMENTO SI.

```
public class Operazione{
  public static void sommaTrenta(int number){
    number += 30;
  }
}
```

```
public class Main{
  public static void main(String[] args){
    int x1;
    int x2;
    int somma;
    Scanner reader = new Scanner(System.in);
    x1 = Scanner.nextInt();
    x2 = Scanner.nextInt();
    somma = x1 + x2;
    Operazione.sommaTrenta(somma);
    System.out.println(somma);
  }
}
```

IN QUESTO ESEMPIO IL DATO "SOMMA" VIENE PASSATO PER VALORE. QUINDI, A MENO CHE NON VENGA SALVATO IN UNA NUOVA VARIABILE, IL SUO VALORE NON PUÓ ESSERE MODIFICATO.

```
public class Operazione{
  public static void removeFirst(ArrayList<Integer> numbers){
    numbers.remove(0);
  }
}
```

```
public class Main{
  public static void main(String[] args){
    ArrayList<Integer> numbers = new ArrayList<>(Arrays.asList(1,2,3));
    Operazione.removeFirst(numbers);
    System.out.println(String.join(", ", numbers));
  }
}
```

IN QUESTO ESEMPIO, IL DATO ARRAYLIST "NUMBERS" VIENE PASSATO PER RIFERIMENTO, QUINDI VIENE MODIFICATO DALLA FUNZIONE A CUI LO PASSO. INFATTI, SE LO PRINTO NON AVRA' PIÚ IL PRIMO ELEMENTO.

4) Il Single Responsability Principle (Principio di Responsabilità Singola) è uno dei cinque principi SOLID: un insieme di linee guida e principi di progettazione che mirano a rendere il codice flessibile. Questo principio afferma che le classi e i metodi devono essere incaricata di un'unica responsabilità (di un unico "compito"). Di seguito un esempio:

```
public class Shop{
// dichiaro le variabili
// scrivo il costruttore
  public void calcolaMedia(Prodotto product){
    float avg=product.getPrice()/product.getNum();
    System.out.println("Il prezzo medio è di: "+ avg);
  }
}
```

Ecco, questo è un errore gravissimo!
Qui il metodo calcolaMedia dovrebbe svolgere una sola funzione (calcolare la media), mentre invece ne svolge 2(la printa anche). Questo è un problema, perché avg viene perso dopo il print (e non disponibile per essere riutilizzato). Sarebbe più corretto restituire avg e printarlo nel main.

5)

```
public interface Pubblicazione {
    float calcolaPrezzo();
}
public class Libro implements Pubblicazione{
    private String titolo;
    private String autore;
    private float prezzo;
    public Libro(String titolo, String autore, float prezzo){
        this.titolo = titolo;
        this.autore = autore;
        this.prezzo = prezzo;
    }
    @Override
    public float calcolaPrezzo(){
        return prezzo *= 5;
    }
    ] METODI GETTERS
      E SETTERS

public class Rivista implements Pubblicazione{
    private String titolo;
    private String editore;
    private float prezzo;
    public Rivista(String titolo, String editore, float prezzo){
        this.titolo = titolo;
        this.editore = editore;
        this.prezzo = prezzo;
    }
    @Override
    public float calcolaPrezzo(){
        return prezzo *= 7;
    }
    ] METODI GETTERS
      E SETTERS
}

public class LibreriaOnline{
    private ArrayList <Pubblicazione> pubblicazioni;
    public LibreriaOnline (ArrayList <Pubblicazione> pubblicazioni ){
        this.pubblicazioni = pubblicazioni;
    }
    public void aggiungiPubblicazione(Pubblicazione pubblicazione){
        pubblicazioni.add(pubblicazione);
    }
    public float calcolaPrezzoTotale( ){
        float somma = 0;
        for ( Pubblicazione pubblicazione : pubblicazioni){
            somma += pubblicazione.calcolaPrezzo();
        }
        return somma;
    }
}
```

```
public class Main{
    public static void main(String[] args){
        Libro libro = new Libro (" Musica ","Ludwig",16.00f);
        Rivista rivista = new Rivista ("Calcio", "Lukaku", 15,00f);
        ArrayList <Pubblicazione> pubblicazioni = new ArrayList <> ( );
        LibreriaOnline libreria = new LibreriaOnline(pubblicazioni);
        libreria.aggiungiPubblicazione (libro);
        libreria.aggiungiPubblicazione (rivista);
        System.out.println (libreria.calcolaPrezzoTotale());
    }
}
```

c)

```
public class Main{
  public static void main(String[] args){
    Stack<Integer> stack = new Stack<>();
    Random rdm = new Random();
    for(int i=0; i<10; i++){
      stack.push(rdm.nextInt(5));
    }
    HashSet<Integer> set = new HashSet<>();
    for(Integer num : stack){
      set.add(num);
    }
    Stack<Integer> stack2 = new Stack<>();
    for(Integer elemento : set){
      stack2.push(elemento);
    }
    System.out.println(stack);
    System.out.println(stack2);
  }
}
```

EJEMPLO RANDOMICAMENTE LO STACK