Testo Antonio Delrio 24/02/2023

1- Cos'è OOP e come la definiamo a diversi livelli.

È un paradigma di programmazione che dà modo di definire oggetti in grado di scambiarsi messaggi, dove si possono definire relazioni di interdipendenza, secondo 3 principi:

- Uso
- Manutenzione
- Specializzazione

2- Definisci Java e i suoi elementi fondamentali

Java è un linguaggio di programmazione basato sul paradigma OOP. Esso è composto da Classi, che sono dei modelli astratti che servono per creare oggetti, secondo sempre 3 principi:

- Incapsulamento

- Ereditarietà

- Polimorfismo

Elementi fondamentali di java sono quindi le Classi, gli attributi e i metodi

Possiamo avere degli attributi e metodi pubblici e privati. Pubblici vuol dire che sono utilizzabili anche fuori dalla classe di definizione, mentre privati solo all’interno

3- Descrivi, spiega e prova i loop e le condizioni che abbiamo visto

In Java abbiamo diversi tipi di condizioni e di loop:

- if, condizioni che servono per esecuzione di codice basato su una condizione, ad esempio “se questo tipo booleano è vero allora fai questo”. Agli if si possono aggiungere gli else, utilizzabili solo dopo un if, che servono per dare un’alternativa all’if, ad esempio “se questo booleano è vero allora fai questo, altrimenti (else) fai quest’altro”. Possiamo concatenare più if else in modo da far susseguire più condizioni diverse.
- Switch, condizioni che svolgono lo stesso lavoro degli if, ma a differenza di questi uno switch fa scegliere su più possibilità e sulla base di quella uguale all’elemento preso in considerazione, restituisce il blocco al suo intero.
- While, permette di svolgere un loop controllato da condizione prima dell’inizio del blocco di codice. Il loop verrà eseguito fin quando la condizione non diventerà falsa.
- Do While, uguale al while, con la differenza che il primo blocco di codice verrà eseguito senza controllare la condizione.
- For, loop che, tramite un controllo su un indice, svolge il loop fino a quando l’indice definito al suo interno smette di rispettare la condizione. Ogni volta che il loop sta per riniziare viene incrementato l’indice.

CODICE:

```
    int a = 10;
        int b = 3;
        int c = 2;
        if(a>10){
            System.out.println("A è maggiore di 10");
        } else if(a<10){
            System.out.println("A è minore di 10");
        }else{
            System.out.println("A è uguale a 10");
        }
        while(b != 5){
            System.out.println("WHILE");
            b++;
        }
        do{
            System.out.println("DO WHILE");
            c++;
        } while(c != 3);
        for(int i = 0; i < 4; i++){
            System.out.println("FOR");
        }
```

4- Cosa sono i metodi cosa sono i loro parametri e quando diventano attributi, e cos'è overloading? provali e spiegali

I metodi sono delle parti di codice definite in una parte diversa dal main, che servono per il raggruppamento del codice, atto a svolgere una determinata risoluzione di un problema. Servono per vari motivi tra cui il riutilizzo del codice e la pulizia del codice.

I metodi possono non ricevere valori oppure riceverli quando vengono invocati. I valori richiesti dai metodi sono detti parametri e diventano attributi una volta che il parametro viene passato.

Esistono metodi statici e metodi di istanza. I metodi statici sono quei metodi che sono utilizzabili solamente all'interno della classe in cui vengono definiti, mentre quelli di istanza possiamo utilizzarli dove vogliamo.

L'overloading (sovraccarico) è una caratteristica di java dove noi possiamo definire diversi metodi con lo stesso nome, che vengono distinti da Java sulla base del numero di parametri.

CODICE:

```
public static void main(String[] args){
        int a = 10;
        int b = 10;
        int c =20;
        overloading(a, b, c);
    }
    public void noStat(int a){
        System.out.println(a);
    }
    public static void overloading(int a, int
b){      //metodo con parametri overloading
        System.out.println(a + b);
    }
    public static void overloading(int a, int b, int
c){   //metodo con parametri overloading
        System.out.println(a + b + c);
    }
    static void ciao(){ //metodo con parametri
        System.out.println("Ciao");
    }
```

5- Cos'è l'incapsulamento funzionale e quando lo usiamo implicitamente?

L’incapsulamento funzionale, o isolamento funzionale dice che ogni blocco di codice singolo che noi creiamo deve essere indipendente dalla sua funzione, e quindi deve poter funzionare in qualsiasi caso.

6- Cos'è il casting e di quali tipi ne abbiamo traccia in java e nell'OOP, provalo e spiegalo

Il casting è una conversione che può essere implicita o esplicita, ma che quando viene definito implicitamente non è del tutto certo che funzioni senza problemi. Per questo in Java sarebbe meglio che avvenga esplicitamente. Ad esempio se noi passiamo un valore intero a un valore double senza specificarlo nel codice, Java lo convertirà implicitamente. Mentre se vogliamo possiamo esplicitamente indicare la conversione scrivendo tra parentesi il tipo nella quale vogliamo convertirlo.

CODICE:

```
    int r = 12;
        int d = (int) 10.85; // conversione esplicita
        double f = r; // conversione implicita
        System.out.println(r);
        System.out.println(d);
        System.out.println(f);
```

7- Cos'è la tipizzazione forte e debole, quali sono i tipi basilari e non e quali sono i modi in cui interagisce con i metodi?

La tipizzazione forte è una tipizzazione nella quale noi dobbiamo sempre specificare il tipo, mentre in quella debole non ne abbiamo bisogno. In Java, ad esempio, non possiamo dire il nome di una variabile senza indicare se intera o double ecc. Cosa che invece avviene ad esempio in Python.

8. Qual’è la differenza tra array e arrayList

Un array è tipo nativo di java che definisce un’aggregazione di più valori dello stesso tipo.

Un arrayList è un oggetto della libreria ArrayList che, come l’array definisce sempre un aggregazione di più valori dello stesso tipo, ma con metodi diversi e diverse funzioni. Ad esempio con un arrayList possiamo aggiungere elementi senza aver definito da prima il numero di elementi massimi e senza andare incontro a overflow. Con l’array questo è possibile sono convertendo l`array in arratyList, aggiungendo l’elemento e poi riconvertendo l’arrayList in array.