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

1: OOP sta per Object Oriented Programming e un metodo di programmazione che permette di definire oggetti che possiamo usare per far interagire un programma tra questi oggetti.
Si ha un linguaggio di programmazione OOP quando è presente:
-Incapsulamento: ovvero la separazioni delle classi in modo che esse siano non dipendenti una dall'altra;
-Ereditarietà: è una caratteristica delle classi in quanto esse possono assumere caratteristiche uguali a quelle di altre classi;
-Polimorfismo: permette di poter eseguire un programma su più interfacce e quindi di conseguenza ne permette il facile riutilizzo;
L'OOP è organizzato con diversi elementi, tra questi i primi sono le classi, a seguire gli oggetti che ha sua volta può essere composto da metodi

2- Definisci Java e i suoi elementi fondamentali:

2: Java è un linguaggio di programmazione OOP based, tra gli elementi principali che compongono java abbiamo String che è l'unico tipo primitivo ad avere l'iniziale maiuscola in quanto è stato introdotto dopo ma per la sua enorme utilità appunto è stato considerato come tipo primitivo. Abbiamo poi gli scanner che ci permettono di inserire in input dei valori per le nostre variabili. Altre funzioni importanti sono gli arrayList che permettono di ricreare un "array" e volendo si possono caricare i valori nell'ArrayList in un array. In java è presente il System.out.print che permette di stampare a video variabili o stringhe.

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

3: Tra i vari cicli di iterazione abbiamo visto le condizioni if, if-else e i loop for, while e do-while.

if:
```
if (condizione)
{
Codice;
}
```
L'if è una condizione che esegue il codice al suo interno se appunto la sua condizione è vera, altrimenti si salta al pezzo di codice che segue

if-else:
```
if(condizione)
{
codice;
}
else
{
codice
}
```
La condizione if-else, come per la condizione if esegue il codice se la condizione dentro la if è vera, altrimenti se questa condizione è falsa si salta al codice contenuto nell'else

for:	for(int i; i == n; i++)
	{
	codice
	}
		Il loop for è un ciclo di iterazione che ripete un pezzo di codice al suo interno fino a quando la condizione posta al centro “i == n” ad esempio diventa vera, in questo caso si esce fuori dal ciclo

while:	while(condizione)
	{
	codice
	}
		Il loop while esegue un pezzo di codice al suo interno fino a quando la condizione rimane vera, c’è bisogno di una operazione che cambi la condizione a falso in modo tale da poter uscire, altrimenti c’è il rischio di incorrere in un loop

infinito (non infinito ma fino a quanto la nostra memoria non termina)

do-while:	do
	{
	codice;
	}
	while(condizione);
		Il loop do-while come per il while esegue il codice all’interno del do fino a quando la condizione d’uscita nel while diventa falsa

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

I metodi sono dei blocchi di codice che possono essere scritti dal main ed essere riutilizzati quando questi vengono chiamati nel main, si possono richiamare metodi in altri metodi anche creando così una ricorsività ma questo produce un salvataggio di dati dispendioso per il programma. I parametri di un metodo sono le variabili inserite nelle ( ), queste sono variabili che vengono definiti nel metodo e che vengono utilizzate appunto nel blocco di codice del metodo, diventano attributi quando vengono richiamati nel main e assumono un valori dunque. L’overloading è un metodo di utilizzo di uno stesso metodo ma con parametri diversi.
Esempio overloading:

```
public static void overloading(int id, String desc)
  {
    System.out.println(id + " " + desc);
  }

public static void overloading(int id, String desc, String desc2)
  {
    System.out.println(id + " " + desc + " " + desc2);
  }
```

Esempio metodo:

```
public static int metodo(int x)
{
	return x;
}
```

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

L’incapsulamento funzionale è un metodo principale usato nella programmazione ad oggetti per rendere inaccessibili pezzi di codice a persone che non dovrebbero e che non serve vedere quel pezzo di codice. Per rendere possibile ciò esistono public e private, questi due rendono possibile che un classe renda visibili con public i suoi attributi ad altre classi invece con private questi rimangono nascosti e solo quella classe potrà utilizzarli.

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

6: Il casting è un metodo di conversione di un tipo attributo ad un altro tipo. Ci sono vari tipi di casting: casting implicito eseguito da java stesso, casting esplicito ovvero nel codice si può notare visibilmente che c’è un tentativo di convertire un numero di tipo float ad esempio ad int con un arrotondamento e Widening casting

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?

7: Una tipizzazione forte richiede la dichiarazione forzata di una variabile in modo tale che esso si chiaramente compreso dal linguaggio di programmazione e generando così errori nel caso quel dato non viene compreso, la tipizzazione debole invece al contrario no
n richiede una dichiarazione forzata del tipo di dato ma questo può causare parecchi errori semantici nel codice. I tipi basilari sono: int, double, float, boolean, String, short, long, char, byte

8-Qual è la differenza tra ArrayList e Array?

8: La differenza principale tra ArrayList e Array è che l’array può essere dichiarato in modo in modo che esso assuma un certa “larghezza” e può contenere solo gli elementi che quella “larghezza può contenere, invece un arrayList ha la capacità di espandersi e quindi modificarsi per contenere più elementi che possono essere aggiunti con il proseguire del codice