2. OOPs Concepts:

Class

Objects

Abstraction

Encapsulation

Inheritance

Polymorphism

3. What is class?

A class — in the context of Java — is a template used to create objects and to define object data types and methods.

Example:

```
public class Website {

  //fields (or instance variable)

  String webName;

  int webAge;

  // constructor

  Website(String name, int age){

    this.webName = name;

    this.webAge = age;

  }

  public static void main(String args[]){

    //Creating objects

    Website obj1 = new Website("beginnersbook", 11);

    Website obj2 = new Website("google", 28);
    //Accessing object data through reference
    System.out.println(obj1.webName+" "+obj1.webAge);
    System.out.println(obj2.webName+" "+obj2.webAge);
  }
}
```

Output:

```
beginnersbook 11
google 28
```

4. What is Object?

An object is an instance of a Java class, meaning it is a copy of a specific class

Java objects have three primary characteristics: identity, state, and behavior.

Examples of states and behaviours

Example 1:

Class: House

State: address, color, area

Behaviour: Open door, close door

```
class House {
  String address;
  String color;
  double area;
  void openDoor() {
    //Write code here
  }
  void closeDoor() {
      //Write code here
   }
 ...
 ...
}
```

5. what is inheritance in java?

Inheritance in Java is a concept that acquires the properties from one class to other classes;

for example, the relationship between father and son.

Inheritance in Java is a process of acquiring all the behaviours of a parent object.

Example Program:

```
class Employee{
 float salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

Output:

```
 Programmer salary is:40000.0
 Bonus of programmer is:10000
```

6. types of inheritance?

* Single Inheritance.

* Multiple Inheritance.

* Multilevel Inheritance.

* Hierarchical Inheritance.

* Hybrid Inheritance.

7. Single Inheritance:

refers to a child and parent class relationship where a class extends the another class.

Example Program:

```
public class Single {
        int c=6+5;


        public static void main(String[] args) {
                Single1 ob = new Single1();
                System.out.println("My single class answer: "+ob.c);
                System.out.println("My single1 class answer: "+ob.d);
        }
}
public class Single1 extends Single{
int d=8+9;
}
```

output:

```
 Programmer salary is:40000.0
 Bonus of programmer is:10000
```

8. Multilevel inheritance:

refers to a child and parent class relationship

where a class extends the child class. For example class A extends class B and class B extends class C.

Example Program:

```
public class Multilevel {

        public void print_my() {

                System.out.print("My ");

        }

        public static void main(String[] args) {

                Multilevel2 ob = new Multilevel2();

                ob.print_my();

                ob.print_name();

                ob.print_ramesh();

        }

}

public class Multilevel1 extends Multilevel{

        public void print_name() {

                System.out.print("Name is ");

        }

}

public class Multilevel2 extends Multilevel1 {

public void print_ramesh() {

System.out.print("Ramesh");}

}
```

Output:

```
My Name is Ramesh
```

9. Hierarchical inheritance:

refers to a child and parent class relationship where more than one classes extends the same class.

For example, class B extends class A and class C extends class A.

Example Program:

```
public class Hierarchical {

        public void classA() {

                System.out.println("ClassA");

        }

        public static void main(String[] args) {

                Hierarchical2 ob = new Hierarchical2();

                Hierarchical1 ob1 = new Hierarchical1();

                Hierarchical3 ob2 = new Hierarchical3();

                ob.classA();

                ob1.classB();

                ob.classC();

                ob2.classD();

}

}

public class Hierarchical1 extends Hierarchical {

public void classB() {
        System.out.println("ClassB");
}
}
class Hierarchical2 extends Hierarchical{
        public void classC() {
                System.out.println("ClassC");
        }
}
class Hierarchical3 extends Hierarchical{
        public void classD() {
                System.out.println("ClassD");
        }
}
```

Output:

```
ClassA
ClassB
ClassC
ClassD
```

10. Multiple Inheritance:

refers to the concept of one class extending more than one classes, which means a child class has two parent classes.

11. What is Encapsulation?

Encapsulation in Java is a process of wrapping code and data together into a single unit,

for example, a capsule which is mixed of several medicines.

Example Program:

```
class EmployeeCount
{
  private int numOfEmployees = 0;
  public void setNoOfEmployees (int count)
  {
     numOfEmployees = count;
  }
  public double getNoOfEmployees ()
  {
     return numOfEmployees;
  }
}
public class EncapsulationExample
{
  public static void main(String args[])
  {
    EmployeeCount obj = new EmployeeCount ();
    obj.setNoOfEmployees(5613);
    System.out.println("No Of Employees: "+(int)obj.getNoOfEmployees());
   }
}
```

Output:

```
No Of Employees: 5613
```

12. What is Polymorphism?

Polymorphism is a object oriented programming feature that allows us to perform a single action in different ways.

Example Program:

```
ublic abstract class Animal{

   ...

   public abstract void animalSound();

}


public class Lion extends Animal{

...

    @Override

    public void animalSound(){

        System.out.println("Roar");

    }

}
```

and

```
public class Dog extends Animal{

...

    @Override

    public void animalSound(){

        System.out.println("Woof");

    }

}
```

13. Types of Polymorphism

1) Static Polymorphism-Method Overloading

2) Dynamic Polymorphism-Method Overriding

14. Method Overloading:

This allows us to have more than one methods with same name in a class that differs in signature.

Example Program:

```
class DisplayOverloading

{

   public void disp(char c)

   {

        System.out.println(c);

   }

   public void disp(char c, int num)

   {

        System.out.println(c + " "+num);

   }

}

public class ExampleOverloading

{

  public static void main(String args[])

  {

      DisplayOverloading obj = new DisplayOverloading();

      obj.disp('a');

      obj.disp('a',10);
  }
}
```

Output:

```
a
a 10
```

15. Method Overriding:

In Java, method overriding occurs when a subclass (child class) has the same method as the parent class.

Example Program:

```
lass Animal{
  public void animalSound(){
        System.out.println("Default Sound");
  }
}
public class Dog extends Animal{

  public void animalSound(){
        System.out.println("Woof");
  }
  public static void main(String args[]){
        Animal obj = new Dog();
        obj.animalSound();
  }
}
```

Output:

```
Woof
```

16. Abstraction:

Abstraction is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user.

Example:

Relavant data is laptop outside like a keyboard,monitor,usb

hiding data is laptop inside like a Ram,Battery,Harddisk

Example Program:

```
abstract class Animal{

  //abstract method

  public abstract void animalSound();

}

public class Dog extends Animal{


  public void animalSound(){

        System.out.println("Woof");

  }

  public static void main(String args[]){

        Animal obj = new Dog();

        obj.animalSound();

  }

}
```

Output:

```
Woof
```

17. Section Sort:

Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list.

Selection logic:

```
public class Selection {
        public static void selectionsort(int[] arr) {
                for(int i=0;i<arr.length;i++) {
                        int minIndex=i;
                        for(int j=i+1;j<arr.length;j++) {
                                if(arr[j]<arr[minIndex]) {
                                        minIndex=j;
                                }
                        }
                        int temp=arr[i];
                        arr[i]=arr[minIndex];
                        arr[minIndex]=temp;
                }
        }
}
public class Selectionmain {
        public static void main(String[] args) {
   int[] arr = {10,5,26,3,15,22};
   Selection.selectionsort(arr);
System.out.println(Arrays.toString(arr));
        }
}
```

Output:

```
[3, 5, 10, 15, 22, 26]
```

18. Bubble Sort:

Bubble sort is a type of sorting algorithm you can use to arrange a set of values in ascending order.

If you want, you can also implement bubble sort to sort the values in descending order.

Bubble Logic:

```
public class Bubble {
        public static void bubblesort(int[] arr) {
                for(int i=1;i<arr.length;i++) {
                        for(int j=0;j<arr.length-1;j++) {
                                if(arr[j]>arr[j+1]) {
                        int temp=arr[j];
                        arr[j]=arr[j+1];
                        arr[j+1]=temp;
                }
        }

}
        }
}
public class BubbleMain {
        public static void main(String[] args) {
   int[] arr = {10,5,2,20,50,77,1};
```

```
    Bubble.bubblesort(arr);

    System.out.println(Arrays.toString(arr));

        }

}
```

output:

```
[1, 2, 5, 10, 20, 50, 77]
```

19. Insertion sort:

Insertion sort is a simple sorting algorithm that allows for efficient,

in-place sorting of the array, one element at a time.

Insertion Logic:

```
public class Insertion {

        public static void insertionsort(int[] arr) {

                for(int i=1;i<arr.length;i++) {

                        for(int j=i;j>0;j--) {

                                if(arr[j]<arr[j-1]) {

                        int temp=arr[j];

                        arr[j]=arr[j-1];

                        arr[j-1]=temp;

                }

                                else

                                        break;

        }


}
```

```
            }
}

public class InsertionMain {

            public static void main(String[] args) {
    int[] arr = {50,6,80,51,3,1,90};
    Insertion.insertionsort(arr);
    System.out.println(Arrays.toString(arr));
            }

}
```

Output:

```
[1, 3, 6, 50, 51, 80, 90]
```

20. Merge Sort?

The merge sort algorithm is based on the principle of divide and conquer

algorithm where a problem is divided into multiple sub-problems.

21. Quick Sort?

QuickSort is a sorting algorithm based on the Divide and Conquer

algorithm that picks an element as a pivot and partitions the given

array around the picked pivot by placing the pivot in its correct position in the sorted array.