Concept Explanation

Abstract class and Interface..

Abstraction in Java

- Abstraction is a process of hiding the implementation details and showing only functionality to the user

There are two ways to achieve abstraction in java

- Abstract class (0 to 100%)
- Interface (100%): Since Java 8, we have default methods

Abstract Class

- A class that is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Important points of Abstract class

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated (object creation).
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method

Example of abstract Class

```
   abstract class Bike{
      Bike(){
            System.out.println("bike is created");
      }
      abstract void run();
      void changeGear(){
            System.out.println("gear changed");
             }
}
 class Honda extends Bike{
        void run(){
            System.out.println("running safely..");
        }
   }
 class Main{
      public static void main(String args[]){
           Bike obj = new Honda();
         obj.run();
         obj.changeGear();
      }
 }
Output:
 bike is created
  running safely..
gear changed
```

Interface In Java

- An Interface in Java is defined as an abstract type used to specify the behaviour of a class. An interface in Java is a blueprint of a class. A Java interface contains static constants and abstract methods

Uses Of an Interface

- It is used to achieve total abstraction
- Since java does not support multiple inheritances in the case of class, by using an interface it can achieve multiple inheritances
- It is also used to achieve loose coupling

Example of Interface

```
public interface Printer{
          int number=10;  // public static final int number=10;
         void print();  // public abstract void print();
}
public class ConsolePrinter implements Printer{
         public void print(){
                  System.out.println("Printing on the console.");
         }
 }
public class FilePrinter implements Printer{
 public void print(){   System.out.println("Printing on the File.");  } }

public class Main{
         public static void main(String args[]){

         ConsolePrinter cp = new ConsolePrinter();
          Printer p1 = new ConsolePrinter();
         Printer p2 = new FilePrinter();
         cp.print();  p1.print();  p2.print();
 } }
```

Output:--

```
Printing on the console.
Printing on the console.
Printing on the File
```