27-May-2022 OOPS Concepts:
********** **********

#. class,objects
#. Abstraction,
#. Encapsulation,
#. Inheritance,
#. Polymorphism-Overloading & Overriding
#. Interface.

ABSTRACTION:-
****

ABSTRACTION IS THE PROCESS OF HIDING IMPLEMENTATION WHILE SHOWING THE FUNCTIONALITITES.

When the class is defined as abstrct keyword ,it should have abstract method.

Example:

```
abstract class  MukeshS
{
   public abstract void serial();
}
```

When the method declared as a abstract that should override at another class and that class should extends the abstract class.

```
package com.basicjava;
abstract class  MukeshS
{
   public abstract void serial();
}
public class AbstractExample extends MukeshS {

   //@Override
   public void serial()
   {
      System.out.println("Mukesh watching Naagini 6");
   }
```

```
    public void intelligent()
    {
       System.out.println("I am intelligent,Humble,Java Developer ");
    }
    public static void main(String[] args)
    {
       AbstractExample ae=new AbstractExample();
            ae.serial();
            ae.intelligent();
    }

}
--------------------------------------------------------------------------------
          ENCAPSULATION:
           ------------------------
FormalDefinition:
******************
   Encapsulation is the process of wrapping of data and methods into single unit.
Example
**********
  #. Capsule which is consider as a Class...which consists of several variables(must have private) and
methods

  Working:
  **********
  #.  when the variable is decalred as private,if we want to access the variable from the other class
we need to provide GETTER AND SETTER METHOD.
  #. However we are accessing variable as a one method that to be an PUBLIC.

 THE ENCAPSULATION IS MAINLY USED FOR ACCESSING THE PRIVATE VARIABLE
 IN ONE CLASS TO BE ACCESSED BY ANOTHER CLASS BY USING GETTER AND SETTER METHODS

 forExample:
 ***************
 package com.basicjava;
```

```
class MukeshEncapsulation
{
   public  static String name="Mukesh";
   public String Dob="05-12-200";
   private int age=21;
   private int salary=30000;
   public int getAge()
   {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public int getSalary() {
      return salary;
   }
   public void setSalary(int salary) {
      this.salary = salary;
   }
}
public class EncapsulationTest {

   public static void main(String[] args)
   {
      MukeshEncapsulation m=new MukeshEncapsulation();
      System.out.println("My name is : " +m.name);
      System.out.println("My DateOfBirth is : " +m.Dob);
      System.out.println("My Age is : " +m.getAge());
      System.out.println("My Salary is : " +m.getSalary());
   }

}
Output:
------------
My name is : Mukesh
```

```
My DateOfBirth is : 05-12-200
My Age is : 21
My Salary is : 30000
-----------------------------------------------------------------------------------------------
            INHERITANCE:
            ******************
   Inheritance is a relationShip between parent and child class
parent class
**************
  can have some methods,the child class can access the parent class
method using child class instance
After accessing the methods of parent class the childClass ca n have the methods and also
child class can have its own methods.

Example:
***********
package com.basicjava;
class OOPS
{
   void features()
   {
      System.out.println("OOPS features Abstraction,Encapsulation,ploymorphism");
   }
   void problemSolving()
   {
      System.out.println("OOPS can also helps to solve Real world problem");
   }
}
public class Java8 extends OOPS
{
   void CollectionFeaturs()
   {
      System.out.println("I am having arrayList,set,Tress,stack,queue");
   }
   void Java8Features()
```

```
141     {
142         System.out.println("I am having lambda(),methodReferance,Stream,TimeAndDate API");
143     }
144
145     public static void main(String[] args)
146     {
147       java8 jv= new java8();
148       jv.features();
149       jv.problemSolving();
150       jv.CollectionFeaturs();
151       jv.Java8Features();
152     }
153
154 }
155  #.If we want to access the parent class methods we need to access by the way of CHILD CLASS'S
     INSTANCE
156  by providing EXTENDS KEYWORD TO THE CHILD CLASS.
157
     ------------------------------------------------------------------------------------------------------------------------
     ------------------
158                  POLYMORPHISM:-
159                  **********************
160   types of polymorphism:
161   **************************
162   Overloading and Overriding
163   *************       *************
164  OVERLOADING;
165  ------------------------
166 #.   SAME METHOD NAME BUT DIFFERENT ARGUMENTS.
167 #.  which occures at single class
168 #.  Its is an example of compile time polymorphism
169 #.  HEAR WE NEED TO CONCENTRATE ON MAILY ARGUMENTS.
170 #.  BASED ON THE RESPECTIVE ARGUMENTS THE CORRESSPONDING METHOD WEILL BE
     EXECUTED.
171 forExample:-
```

```
--------------------
package com.basicjava;
class Calculator
{
   void sum(int a,int b)
   {
      int c=a+b;
      System.out.println("The sum of number is : " +c);
   }
   void sum(float a,float b)
   {
      float c=a-b;
      System.out.println("The sum of number is : " +c );

   }
   void sum(double a,float b,int c)
   {

      System.out.println("The sum of number is : " +(a+b+c));
   }
}
public class OverLodingExample
{
   public static void main(String[] args)
   {
      Calculator c=new Calculator();

      c.sum(25f,23f);
      c.sum(56,22);
      c.sum(32.5,23f,12);

   }

}
output:
```

```
----------
The sum of number is : 2.0
The sum of number is : 78
The sum of number is : 67.5
-------------------------------------------------------------------------------------
        OVERRIDING:
        ***************
#. Overriding is the example of RUNTIME POLYMORPHISM.
#. Hear the WE ARE have the SAME METHODNAME AND SAME ARGUMENTS
   EXECUTION:
   ------------------
#. Based on the specific class instance the respective class's method will execute
#. Mainly concentrate on INSTANCE OF CLASS.
#. HEAR THE DIFFERNT CLASSES CAN HAVE THE SAME METHOD NAME AND ARGUMENTS
EXAMPLE:
---------------
package com.basicjava;
class Mukesh
{
   void reading()
   {
      System.out.println("Mukesh reading Overriding concept now");
   }
   void learning()
   {
      System.out.println("Mukesh learning JAVA");
   }
}
class Mani
{
   void reading()
   {
      System.out.println("Mani reading Ponniyan Selvan Navel");
   }
   void learning()
```

```
    {
        System.out.println("Mani learning JAVA and PYTHON");
    }
}
public class OverridingExample {

    public static void main(String[] args)
    {
        Mukesh mu=new Mukesh();
        mu.reading();
        mu.learning();
        Mani ma=new Mani();
        ma.reading();
        ma.learning();

    }

}
--------------------------------------------------------------------------------
```