Week 2:

A) Write Java program on use of inheritance, preventing inheritance using final, abstract classes.

Aim : Java program on use of inheritance

Description :

- Inheritance is defined as a mechanism where the sub or child class inherits the properties and characteristics of the super class or other derived classes.
- It also supports additional features of extracting properties from the child class and using it into other derived classes."

Procedure:

- Define a class (parent class) and write some properties
- Define one more class (child class) write some methods and inherit properties from parent class
- Create an object for child class and access the both classes properties with child class object

Program:

```
class       MathOperations
{
            public void addition(int  x, int   y)
            {
                    System.out.println(" The sum of the given numbers ="+(x+y));
            }

            public void substraction(int  x, int   y)
            {
                    System.out.println(" The difference of the given numbers ="+(x-y));
            }
}
class       MoreOperations  extends  MathOperations
{
            public void multiplication(int  x, int   y)
            {
                    System.out.println(" The sum of the given numbers ="+(x*y));
            }
```

```
        public static void main(String[]  args)
        {
                int a=40,b=20;
                MoreOperations     m = new MoreOperations();
                 m.addition(a,b);
                 m.substraction(a,b);
                 m.multiplication(a,b);
        }
}
```

Output:

```
        The sum of the given numbers =60
        The difference of the given numbers =20
        The product of the given numbers =800
```

b) preventing inheritance using final

Aim: preventing inheritance using final and abstract classes

Description:

- ➢ The class declared as final is called final class. Note that the final class cannot be inherited.
- ➢ There are two uses of the final class i.e. to prevent inheritance and to make classes immutable.

Procedure:

- ➢ Develop one class with final modifier
- ➢ Develop more class and extends from base class

Program:

```
final class Demo1
{
        // variables, methods, and fields
}

class Demo2 extends Demo1       // The following class is illegal
{
        // COMPILE-ERROR! Can't subclass A
}
```

Output:

```
        D:\>javac Demo2.java
        Demo2.java:6: error: cannot inherit from final Demo1
        class Demo2 extends Demo1       // The following class is illegal
```

C) Aim: preventing inheritance using abstract classes

Description:

- An abstract class can also have methods that are neither abstract nor final, just regular methods.
- There may be "final" methods in "abstract" class. But, any "abstract" method in the class can't be declared final. It will give **"illegal combination of modifiers: abstract and final"** error.

public abstract final void show(); // illegal combination of modifiers: abstract and final

Program:

```
abstract   class      Demo1              //ABSTRACT CLASS
{
             public final void show()       // FINAL METHOD
           {
                       System.out.println("Yes");
            }
 }
class Demo2 extends Demo1             //INHERTING ABSTRACT CLASS
{
              public void show()              //OVERRIDING THE FINAL METHOD
            {
                       System.out.println("Success overriding");
            }
            public static void main (String[] args)
           {
                     Demo2   id = new Demo2();       //OBJECT OF SUBCLASS
                      id.show();                    //CALLING FINAL METHOD
            }
}
```

Output:

```
D:\>javac Demo2.java

Demo2.java:19: error: show() in Demo2 cannot override show() in Demo1

   public void show()              //OVERRIDING THE FINAL METHOD

              ^

 overridden method is final

     1   error
```

B) Write Java program on dynamic binding, differentiating method overloading and overriding.

Aim: Java program on dynamic binding, differentiating method overloading and overriding.

Description:

The **key difference** between overloading and overriding in Java is that the **Overloading is the ability to create multiple methods of the same name with different implementations and Overriding is to provide an implementation for a subclass method that already exists in the superclass**.

Procedure:

- we have created two methods, first sum() method performs addition of two numbers and second sum method performs addition of three numbers.
- Create an object for that class and call both sum methods using period (.) operator

Example <u>**By changing number of arguments**</u>

```
class Addition
{
        void sum(int a, int b)    // addition of two numbers
        {
                System.out.println(a+b);
        }
        void sum(int a, int b, int c)    addition of three numbers
        {
                System.out.println(a+b+c);
        }

        public static void main(String args[])
        {
                Addition  obj   =    new    Addition();
                obj.sum(10, 20);
                obj.sum(10, 20, 30);
        }
}
```

Output

```
30
60
```

Method overriding :

Description:

- If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.
- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.

Rules of overriding:

- The method must have the same name as in the parent class
- The method must have the same parameter as in the parent class.
- There must be an IS-A relationship (inheritance).

Procedure:

Create a class (base class) and define a method
Create one more class (derived) and extend using base class
Redefine method in derived class as per our own requirement
Define a main method , and create an object for derived class and class the method

Program:

```
 class      Demo1
{
            public  void show(int a, int b , int c)
          {
                      System.out.println("sum ="+(a+b+c));
           }
 }

class Demo2 extends Demo1
{
              public void show(int a,int b,int c)              //OVERRIDING THE SUM METHOD
           {
                   int d=40;
                     System.out.println("sum ="+(a+b+c+d));
           }
           public static void main (String[] args)
          {
                   Demo2   id = new Demo2();       //OBJECT OF SUBCLASS
                    id.show(10,20,30);                    //CALLING  METHOD
           }
}
```

Output:

```
sum =100
```

c) Develop a java application to implement currency converter (Dollar to INR, EURO to INR, Yen) using Interfaces

Description:

INR is currency of INDIA
EURO is currency of European union
YEN is currency of Japan

Procedure:

- Declare methods in interface
- Define those methods in class
- Create a menu in do-while loop
- Use choice in switch to use this method

Program:

```
import java.util.*;

interface converter
{
	void dollar_rupee();
	void rupee_dollar();
	void euro_rupee();
	void rupee_euro();
	void yen_rupee();
	void rupee_yen();

}
public class currency implements   converter
{
	double inr,usd;
	double euro,yen;

	Scanner in=new Scanner(System.in);

	public void dollar_rupee()
	{
		System.out.println("Enter dollars to convert into Rupees:");
		usd=in.nextInt();

		inr=usd*67;
		System.out.println("Dollar ="+usd+"equal to INR="+inr);
	}

	public void rupee_dollar()
	{
		System.out.println("Enter Rupee to convert into Dollars:");
		inr=in.nextInt();

		usd=inr/67;
		System.out.println("Rupee ="+inr+"equal to Dollars="+usd);
	}
```

```
public void euro_rupee()
{
        System.out.println("Enter euro to convert into Rupees:");
        euro=in.nextInt();

        inr=euro*79.50;
        System.out.println("Euro ="+euro +"equal to INR="+inr);
}

public void rupee_euro()
{
        System.out.println("Enter Rupees to convert into Euro:");
        inr=in.nextInt();

        euro=(inr/79.50);
        System.out.println("Rupee ="+inr +"equal to Euro="+euro);
}

public void yen_rupee()
{
        System.out.println("Enter yen to convert into Rupees:");
        yen=in.nextInt();

        inr=yen*0.61;
        System.out.println("YEN="+yen +"equal to INR="+inr);
}

public void rupee_yen()
{
        System.out.println("Enter Rupees to convert into Yen:");
        inr=in.nextInt();

        yen=(inr/0.61);
        System.out.println("INR="+inr +"equal to YEN"+yen);
}

public static void main(String args[])
{

                int ch;
                currency     c   =   new currency();


        do
        {
                System.out.println("1.dollar to rupee ");
                System.out.println("2.rupee to dollar ");
                System.out.println("3.Euro to rupee ");
                System.out.println("4..rupee to Euro ");
                System.out.println("5.Yen to rupee ");
```

```
                        System.out.println("6.Rupee to Yen ");

                        Scanner in=new Scanner(System.in);

                        System.out.println("Enter 0 to  quit and 1 to continue ");
                        ch=in.nextInt();

                             switch(ch)
                             {
                             case 1:
                                  {
                                       c.dollar_rupee();
                                       break;
                                  }
                             case 2:
                                  {
                                       c.rupee_dollar();
                                       break;
                                  }
                             case 3:
                                  {
                                       c.euro_rupee();
                                       break;
                                  }
                             case 4:
                                  {
                                       c.rupee_euro();
                                       break;
                                  }
                             case 5:
                                  {
                                       c.yen_rupee();
                                       break;
                                  }
                             case 6:
                                  {
                                       c.rupee_yen();
                                       break;
                                  }
                             default:
                                   System.exit(0);
                        }      //switch close
                   }while(ch==1);    //do-while close
              }                  // main close
         }    //class close
```

Output:

```
D:\>java  currency
1.dollar to rupee
2.rupee to dollar
3.Euro to rupee
4..rupee to Euro
5.Yen to rupee
6.Rupee to Yen
Enter 0 to  quit and 1 to continue
1
Enter dollars to convert into Rupees:
30
Dollar =30.0equal to INR=2010.0
1.dollar to rupee
2.rupee to dollar
3.Euro to rupee
4..rupee to Euro
5.Yen to rupee
6.Rupee to Yen
Enter 0 to  quit and 1 to continue
2
Enter Rupee to convert into Dollars:
10
Rupee =10.0equal to Dollars=0.14925373134328357
```