Unit-2 [OOPs Concept]	
1.	Write a java code to assign and display following Employee information using class. EmpId=291056 EmpName=”xyz” Department=”Sales” Salary=5,25,000
2.	Write a java code to pass a value of student rollno and name using default and parameterized constructor.
3.	Write a Java program to perform calculator operations like addition,multiplication,division,substraction by using single inheritance.
4.	Write a java code to print study information of student like college name,course name,subject name using multi-level inheritance.
5.	Write a java code to print area of rectangle and area of cube by using hierarchical inheritance.
6.	Write a java code to print employee information(empid,empname,salary) using this keyword concept.
7.	Write a java code to print perform calculator operation by creating addition and subtraction interface using interface inheritance.
8.	Write a java code to perform program-7 using interface (achieve by abstraction).
9.	Write a java code to find the area and volume of box using super keyword.
10.	Write a java code to display information about employee using abstract class.
11.	Write a java code to swap two numbers by creating a single package and package name given as ‘swap_pkg’.
12.	Write a java code to print course detail using this keyword. Course detail like cid,cname,description
13.	Write a java code to implement program-12 using encapsulation.
14.	Write a java code to to find area of rectangle and area of circle by creating different class rectangle and circle and achieve Multiple inheritance.
15.	Write a java code to print course detail like cid,cname,description using constructor overloading

Program-1:

```java
public class my_oops2 {
  String fname = "abc";
  String lname = "def";
  int age = 24;

  public static void main(String[] args) {
    my_oops2 m1 = new my_oops2();
    System.out.println("Name: " + m1.fname + " " + m1.lname);
    System.out.println("Age: " + m1.age);
  }
}
```

Program-2

```java
public class c_constructor {
  String sname;

  public c_constructor()
  {
    sname="rutvi";
  }

  public static void main(String[] args) {

    c_constructor myObj = new c_constructor();
    System.out.println(myObj.sname); // Print the value of x
  }
}
```

Program-3:

```java
/*
->abstract method of an abstract class can be defined in its
subclass
->An instance of an abstract class can not be created.
->Constructors are allowed.
->We are not allowed to create objects for an abstract class.
->If a class contains at least one abstract method then compulsory should declare a
class as abstract
*/
public class abstract_class {

    Run main | Debug main | Run | Debug
    public static void main(String args[]) {
        Derived d = new Derived();
        d.fun();
    }
}

abstract class Base {
    Base() {
        System.out.println(x:"Base Constructor Called");
    }

    abstract void fun();
}

class Derived extends Base {
    Derived() {
        System.out.println(x:"Derived Constructor Called");
    }

    void fun() {
        System.out.println(x:"Derived fun() called");
    }
}
```

Program-4:

Single inheritance:

single_inherit.java > Language Support for Java(TM) by Red Hat > single_inherit

```java
public class single_inherit {
    Run main | Debug main | Run | Debug
    public static void main(String args[]) {
        b b1 = new b();
        b1.i = 10;
        b1.j = 20;
        b1.k = 30;

        b1.show();
        b1.display();
    }
}

class a {
    int i, j;

    void show() {
        System.out.println("i=" + i);
        System.out.println("j=" + j);
    }
}

class b extends a {
    int k;

    void display() {
        System.out.println("k=" + k);
        System.out.println("i=" + i);
        System.out.println("j=" + j);
    }
}
```

Program-5:

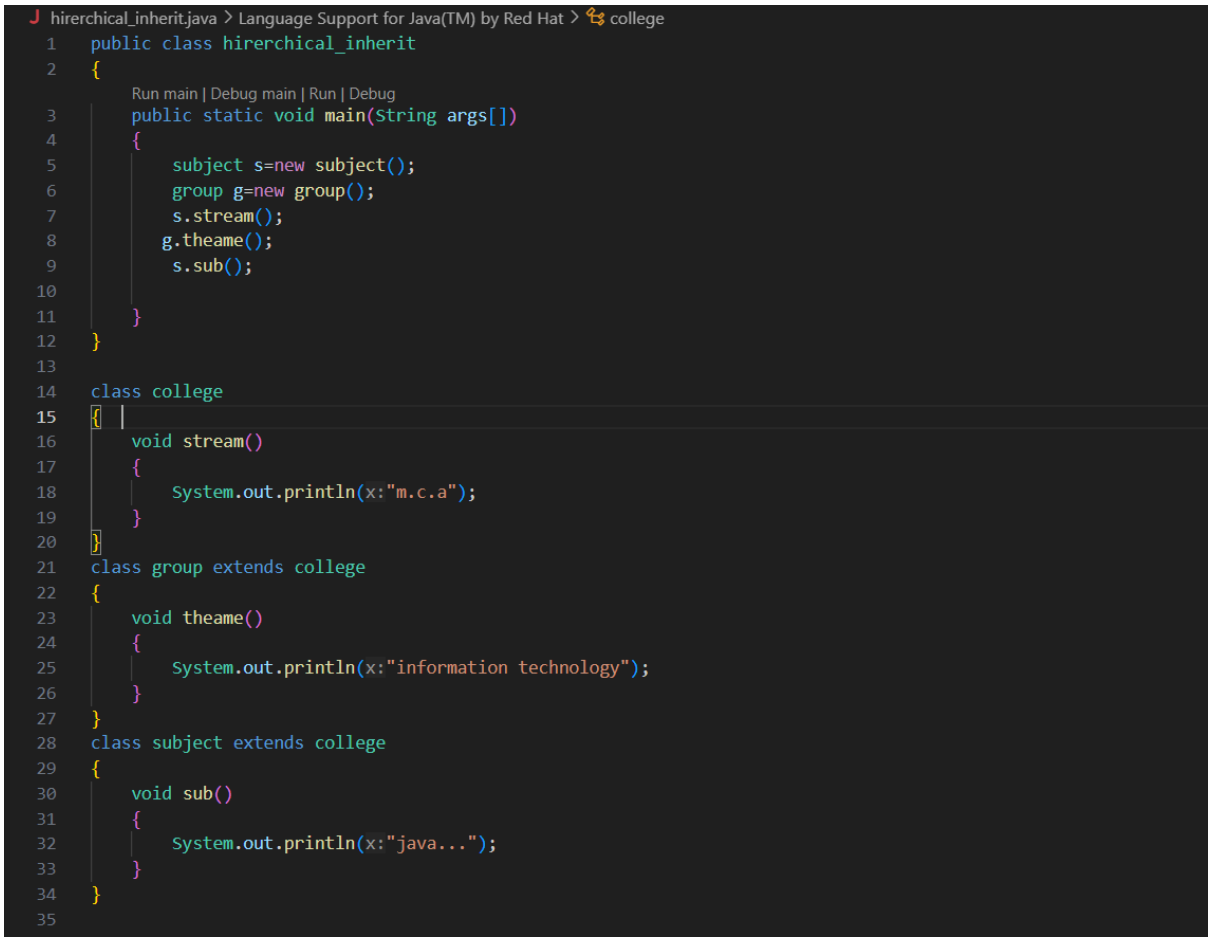
Multilevel inheritance:

multilevel_inherit.java > Language Support for Java(TM) by Red Hat > multilevel_inherit

```java
public class multilevel_inherit
{
    Run main | Debug main | Run | Debug
    public static void main(String args[])
    {
        subject s=new subject();
        s.stream();
        s.theame();
        s.sub();

    }
}

class college
{
    void stream()
    {
        System.out.println(x:"m.c.a");
    }
}
class group extends college
{
    void theame()
    {
        System.out.println(x:"information technology");
    }
}
class subject extends group
{
    void sub()
    {
        System.out.println(x:"java...");
    }
}

```

Program-6:

Hierarchical inheritance:



Program-7:Abstaction(Runtime Polymorphism)

```java
/*
->abstract method of an abstract class can be defined in its
subclass
->An instance of an abstract class can not be created.
->Constructors are allowed.
->We are not allowed to create objects for an abstract class.
->If a class contains at least one abstract method then compulsory should declare a
class as abstract
*/
public class abstract_class {

    Run main | Debug main | Run | Debug
    public static void main(String args[]) {
        Derived d = new Derived();
        d.fun();
    }
}

abstract class Base {
    Base() {
        System.out.println(x:"Base Constructor Called");
    }

    abstract void fun();
}

class Derived extends Base {
    Derived() {
        System.out.println(x:"Derived Constructor Called");
    }

    void fun() {
        System.out.println(x:"Derived fun() called");
    }
}

```

Program-8:

```java
/*
Rules for Java Method Overriding
it is a runtime polymorphisam
The method must have the same name as in the parent class
The method must have the same parameter as in
 the parent class.
*/
public class method_override
{
    Run main | Debug main | Run | Debug
    public static void main(String args[])
    {
        b1 bobj=new b1();
        bobj.disp();
    }
}
class a1
{

    void disp()
    {
        System.out.println(x:"base method call");
    }
}
class b1 extends a1
{

        void disp()
        {
            System.out.println(x:"child method call");

        }
}

```

Program-9:

super_keyword.java > Language Support for Java(TM) by Red Hat > super_keyword

```java
/*
--super keyword always use in sub class constructor
--it must be first statement in derived class
--derived class constructor call the base class constructor

*/
public class super_keyword
{
    Run main | Debug main | Run | Debug
    public static void main(String args[])
    {
        mybox m=new mybox(h:2,w:5,d:3);
        int a=m.area();
        int v=m.volume();
        System.out.println("area="+a);
        System.out.println("volume="+v);
    }
}
class box
{
        int height,width;
        box(int h,int w)
        {
            height=h;
            width=w;
        }
        int area()
        {
            return(height * width);
        }
}
}
class mybox extends box
{
    int depth;
    mybox(int h,int w,int d)
    {
        super(h,w);
        depth=d;
    }
    int volume()
    {
     return(height*width*depth);
    }
}
```

Program-10:

this_keyword.java > Language Support for Java(TM) by Red Hat > this_keyword

```java
/*
The this keyword can be used to refer current class instance variable.
If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.
this.variable_name=variable name;
*/
public class this_keyword {
    Run main | Debug main | Run | Debug
    public static void main(String args[]) {
        Student s1 = new Student(rollno:111, name:"ankit", fee:5000f);
        Student s2 = new Student(rollno:112, name:"sumit", fee:6000f);
        s1.display();
        s2.display();
    }
}

class Student {
    int rollno;
    String name;
    float fee;

    Student(int rollno, String name, float fee) {
        this.rollno = rollno;
        this.name = name;
        this.fee = fee;
    }

    void display() {
        System.out.println(rollno + " " + name + " " + fee);
    }
}

```