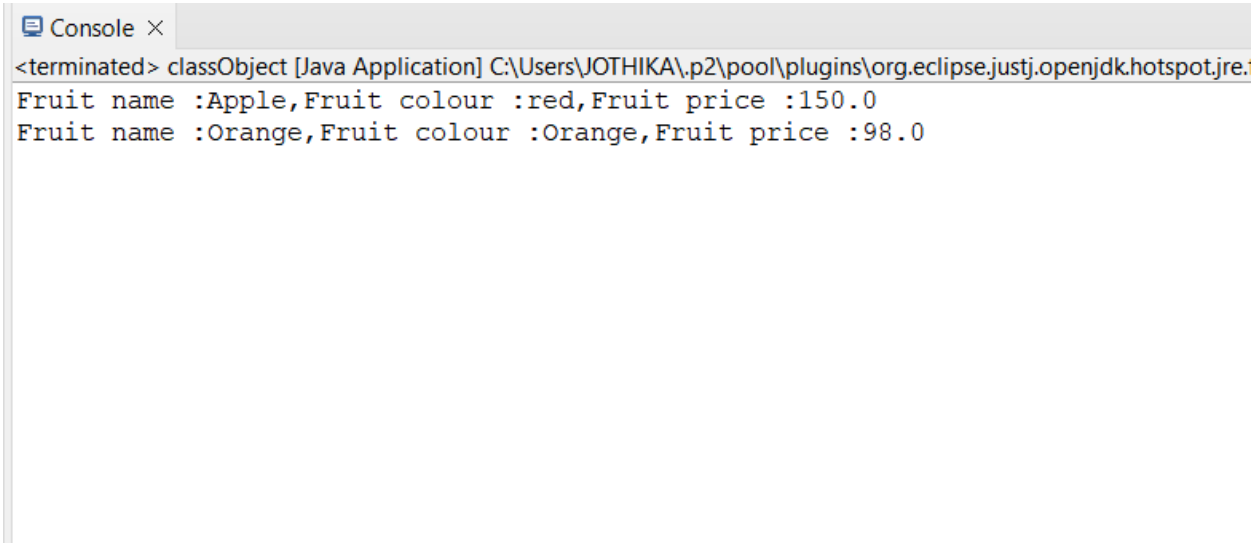
8.Write a program in Java to demonstrate the uses of classes, objects, and the object-oriented pillars in Java

//Class and Object

```
package oops;

	class Fruit{  // fruit is a class
		String  name;
		String colour;
		double price;

		public Fruit(String name, String colour, double price) {
			this.name = name;
			this.colour = colour;
			this.price = price;
		}


		public String getName() {
			return name;
		}


		public void setName(String name) {
			this.name = name;
		}


		public String getColour() {
			return colour;
		}


		public void setColour(String colour) {
			this.colour = colour;
		}


		public double getPrice() {
			return price;
```

```
        }


        public void setPrice(double price) {
            this.price = price;
        }



    @Override
        public String toString() {
            return "Fruit name :" + name + ",Fruit colour :"
              + colour +     ",Fruit price :" + price + "";
        }

}
    public class classObject{
  public static void main(String[] args) {
        Fruit a=new Fruit("Apple","red",150.0);
        Fruit b=new Fruit("Orange","Orange",98.0);
            System.out.println(a.toString());
            System.out.println(b.toString());
}
    }
```

Output

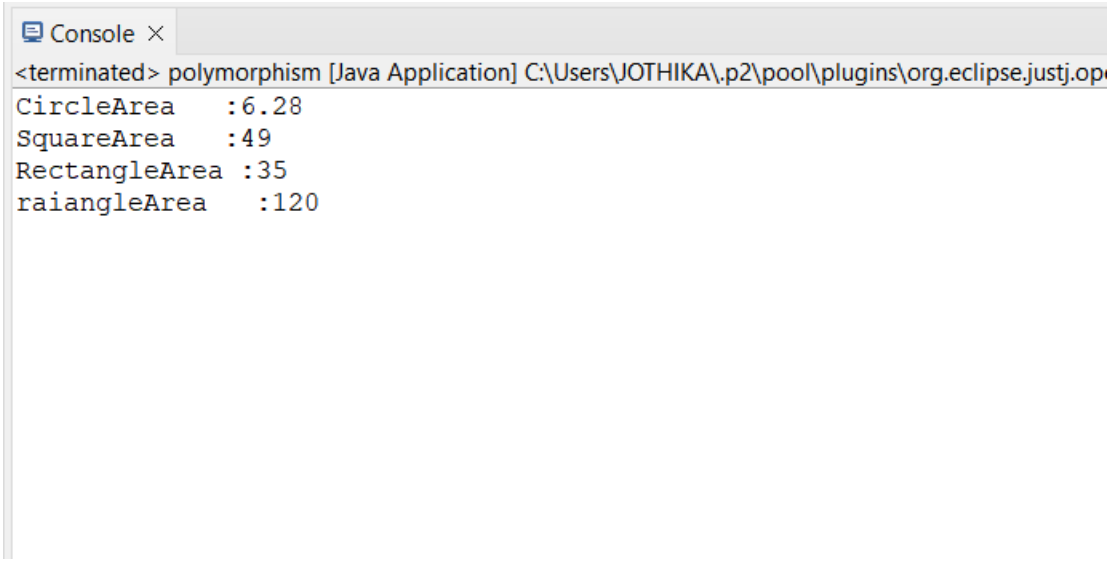


//polymorphism

```
package oops;
        class CircleArea{
            double area(double x) {
                return  3.14 *x;
            }
        }
        class SquareArea{
             int area(int x) {
                 return x*x;
            }
        }
        class RectangleArea{
            int area(int x,int y) {
                return x*y;
            }
        }
        class TraiangleArea{
            int area(int y,int x) {
                return y*x/2;
            }
        }

        public class polymorphism {
```

```
public static void main(String[] args) {
        CircleArea c=new CircleArea();
        SquareArea s=new SquareArea ();
   RectangleArea r=new  RectangleArea();
   TraiangleArea t=new TraiangleArea();
System.out.println("CircleArea   :" +c.area(2));
System.out.println ("SquareArea   :" +s.area(7));
System.out.println ("RectangleArea :"+r.area(5,7));
System.out.println ("raiangleArea   :"+t.area(80, 3));
}}
```

Output

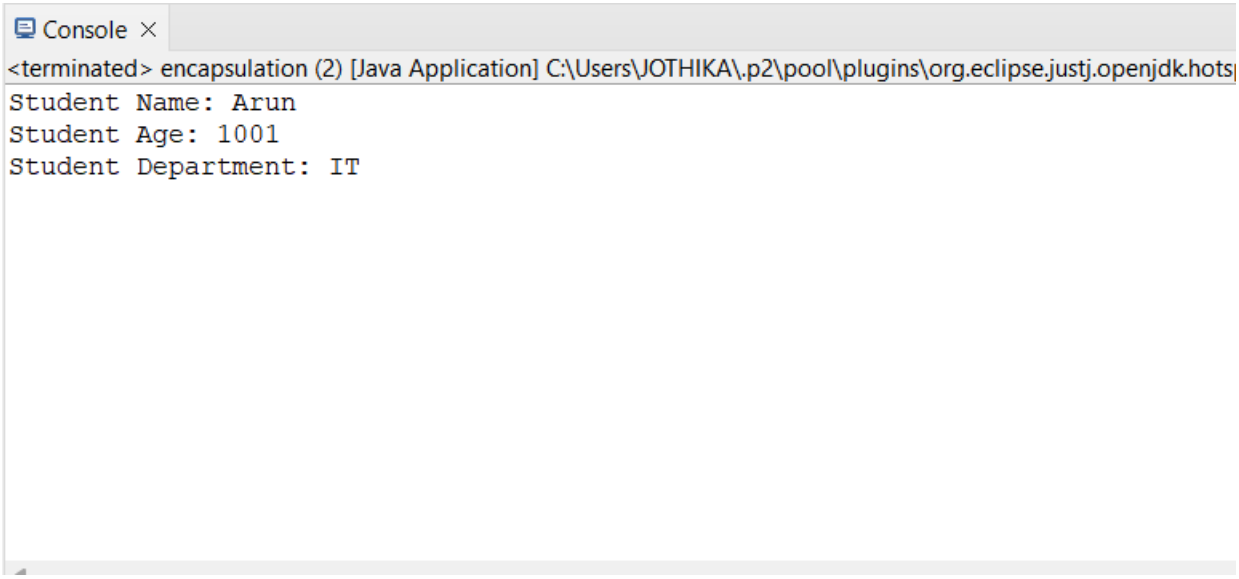


//encapsulation

```
package oops;
class Student{
        private String name;  //it has a private data member,  use getter ,setter
        private int id;
        private String department;

  public String getName()
  {
        return name;
  }
  public int getId()
  {
          return id;
        }
  public String getDepartment()
  {
          return department;
        }
  public void setName( String n)
   {
         this.name=n;
   }
  public void setId( int id)
  {
         this.id=id;
  }


  public void setDepartment( String dept)
  {
         this.department=dept;
  }

 }


public class encapsulation{
```

```
	public static void main(String args[]) {
		Student s=new Student();
		s.setName("Arun");
		s.setId(1001);
		s.setDepartment("IT");
		System.out.println("Student Name: "+s.getName());
		System.out.println("Student Age: "+s.getId());
		System.out.println("Student Department: "+s.getDepartment());
	}

}
```

Output

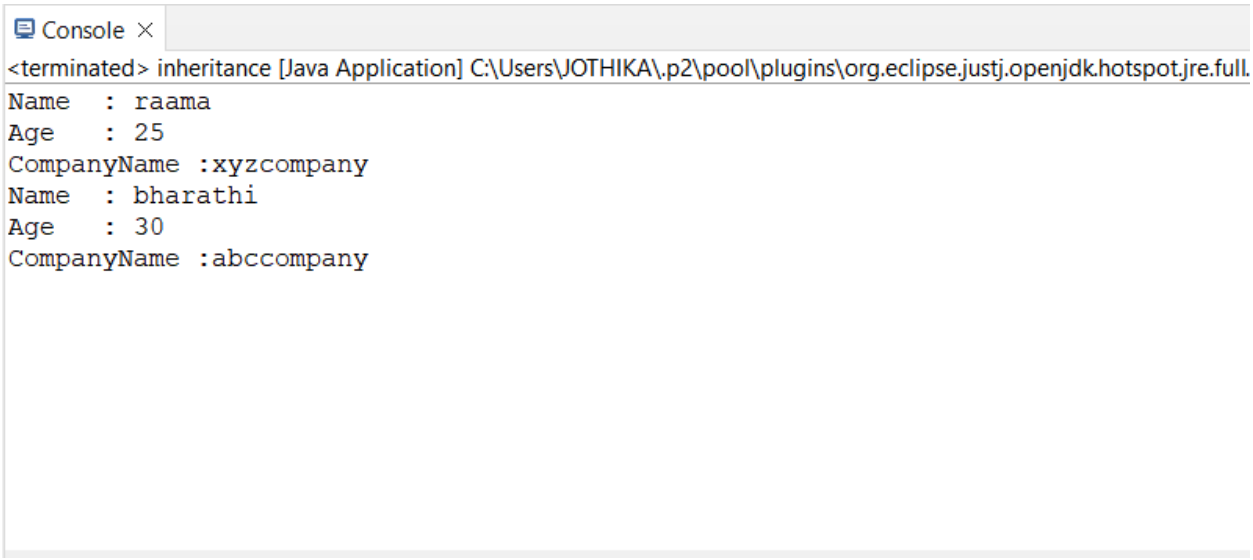


//Inheritance

```
package oops;

 class user{
        String name;
        int age;
  user(String n,int a){
                this.name=n;
                this.age=a;
          }
        public void display() {
                System.out.println("Name  : "+name);
                System.out.println("Age   : "+age);
        }

}

class Mainprogrammer extends user{
        String companyName;
        Mainprogrammer(String n,int a,String c){
                super(n,a);
                this.companyName=c;
        }
        public void display() {
                System.out.println("Name  : "+name);
                System.out.println("Age   : "+age);
                System.out.println("CompanyName :"+companyName);
        }

}
public class inheritance {


        public static void main(String[] args) {
                // TODO Auto-generated method stub
        Mainprogrammer  m=new Mainprogrammer("raama",25,"xyzcompany");
                m.display();
```

```
        Mainprogrammer  m1=new Mainprogrammer("bharathi",30,"abccompany");
                m1.display();

        }

}
```

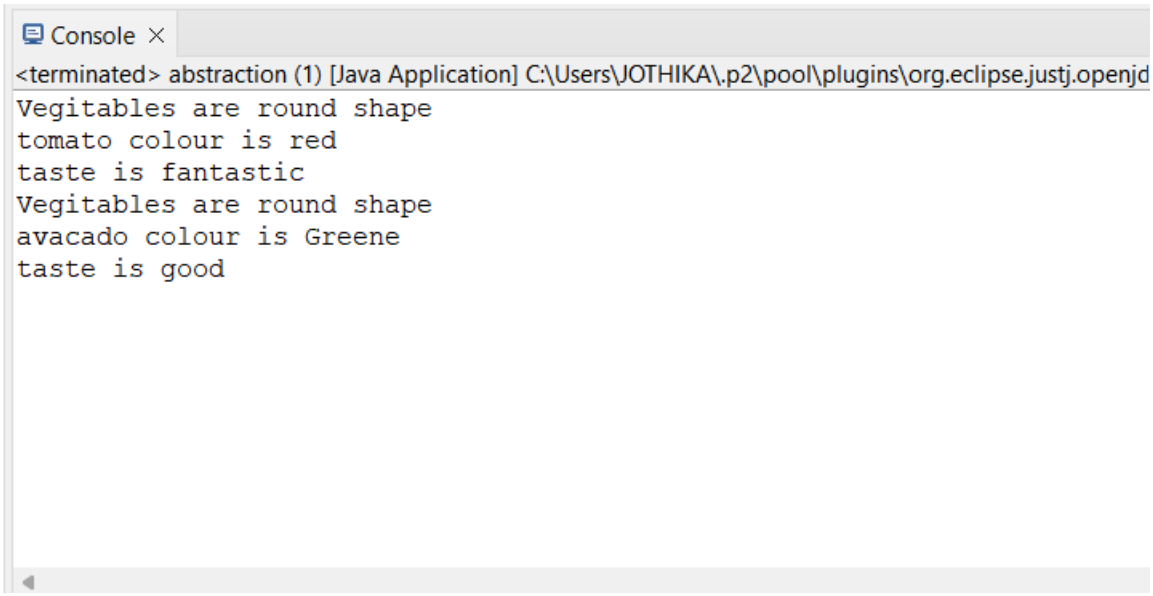
Output



//abstraction

```
package oops;

abstract class Vegitable{
	void shape() {
		System.out.println("Vegitables are round shape");
	}
	abstract void colour();
}
class tomato extends Vegitable{
	void colour() {
		System.out.println("tomato colour is red");
	}
	void taste(){
		System.out.println("taste is fantastic");
	}

}
class avacado extends Vegitable{
	void colour() {
		System.out.println("avacado colour is Greene");
	}
	void taste(){
		System.out.println("taste is good");
	}
}
 public class abstraction{
public static void main(String args[]) {
	 tomato t=new  tomato();
	 avacado a=new avacado();
	 t.shape();
	 t.colour();
	 t.taste();

	 a.shape();
	 a.colour();
	 a.taste();

}}
```

Output


Console
<terminated> abstraction (1) [Java Application] C:\Users\JOTHIKA\.p2\pool\plugins\org.eclipse.justj.openjd
Vegitables are round shape
tomato colour is red
taste is fantastic
Vegitables are round shape
avacado colour is Greene
taste is good