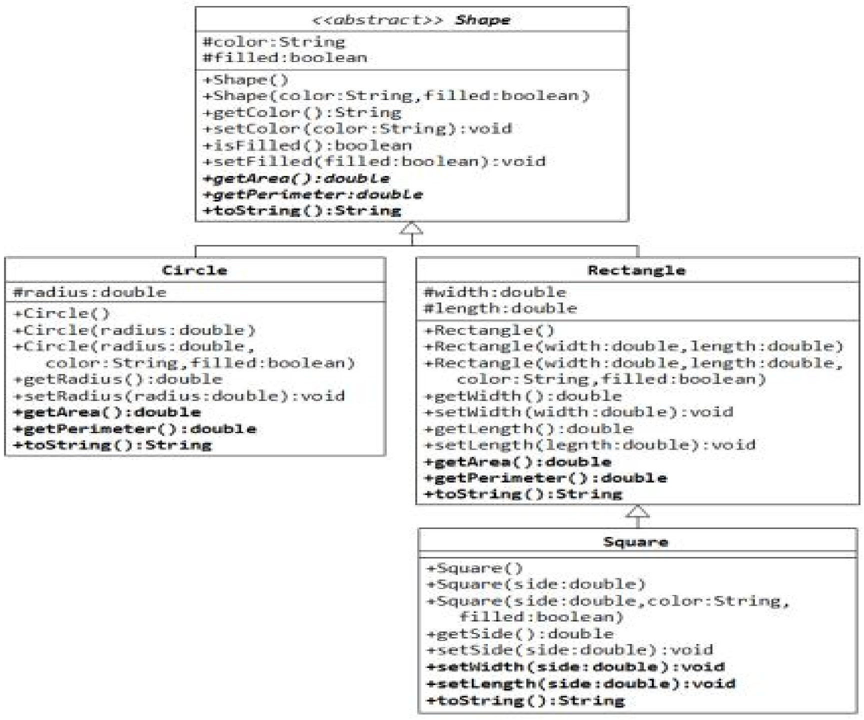
Ex no: 3	**Develop programs incorporating packages, abstract classes, and interfaces to structure code modularly and enforce abstraction.**
Date:	

<u>Aim:</u>

To implement the following relationship and create a main class to invoke all the methods.

11. Write a Java program to implement the following relationship and create a Main class to invoke all the methods.



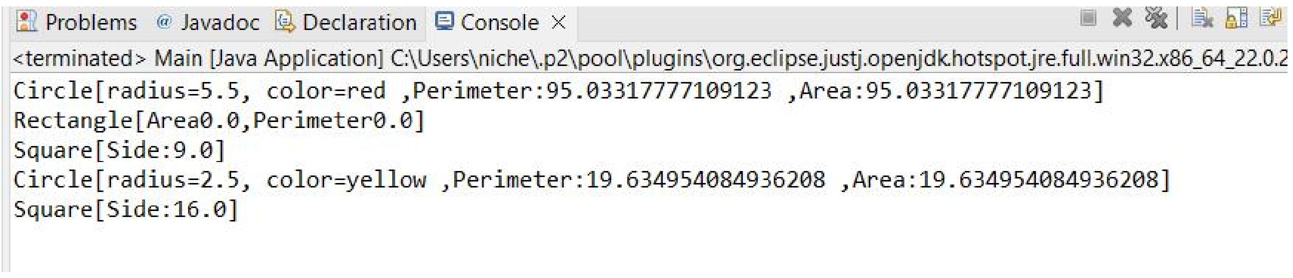
CODE:

```
public abstract class Shape {
        protected String color;
        protected boolean filled;
        public Shape() {}
        public Shape(String color,boolean filled) {
                this.color=color;
                this.filled=filled;}
        public String getColor(){
                return color;}
        public void setColor(String color) {
                this.color=color;}
        public boolean isFilled() {
                return filled;}
        public void setFilled(boolean filled) {
                this.filled=filled;}
        public abstract double getArea();
        public abstract double getPerimeter();
        public abstract String toString();}
public class Circle extends Shape{
            private double radius;
             public Circle(double radius, String color,boolean filled) {
                this.radius = radius;
                this.color = color;
                this.filled=filled; }
            public void Circle(double radius) {
                this.radius = radius; }
            public double getRadius() {
                return radius;}
            public double getArea() {
                return Math.PI * radius * radius;}
            public double getPerimeter() {
                return (Math.PI * radius * radius);}
            public String toString() {
                return "Circle[radius=" + radius + ", color=" + color + " ,Perimeter:" + getPerimeter() + " ,Area:"
+ getArea() + "]"; }
}
//Rectangle Class
public class Rectangle extends Shape{
        protected double width;
        protected double length;
        public Rectangle() {}

        public Rectangle(double width,double length){
                this.width=width;
                this.length=length;}
        public double getWidth(){
                return width;}
        public void setWidth(double width) {
                this.width=width;}
        public double getLength() {
                return length;}
        public void setLength(double length) {
                this.length=length;}
        public double getArea() {
                return width*length;}
        public double getPerimeter() {
                return 2*(length+width);}
        public String toString(){
                return "Rectangle[Area"+getArea()+",Perimeter"+getPerimeter()+"]" ;}
}
public class Square extends Rectangle{
        protected double side;
        public Square() {}
        public Square(double side) {
                this.side=side;}
        public Square(double side,String color,boolean filled) {
                this.side=side;
                this.color=color;
                this.filled=filled;}
        public double getSide() {
                return side;}
        public void setSide(double side){
                this.side=side;}
        public void setWidth(double width) {
                this.width=width;}
        public void setLength(double length) {
                this.length=length;}
        public String toString() {
                return "Square[Side:"+side*side+"]";}
}

package dhanush;
public class Main {
public static void main(String[] args) {
Shape circle = new Circle(5.5, "red", false);
Shape rectangle = new Rectangle();
Shape square = new Square(3.0, "green", true);
System.out.println(circle);
System.out.println(rectangle);
System.out.println(square);
Circle c = new Circle(0, "green", false);
c.setRadius(2.5);
c.setColor("yellow");
c.setFilled(false);
System.out.println(c);
Rectangle r = new Rectangle();
r.setWidth(2.0);
r.setLength(3.0);
r.setColor("orange");
r.setFilled(true);
Square s = new Square();
s.setSide(4.0);
s.setColor("purple");
s.setFilled(true);
System.out.println(s);
}}
```

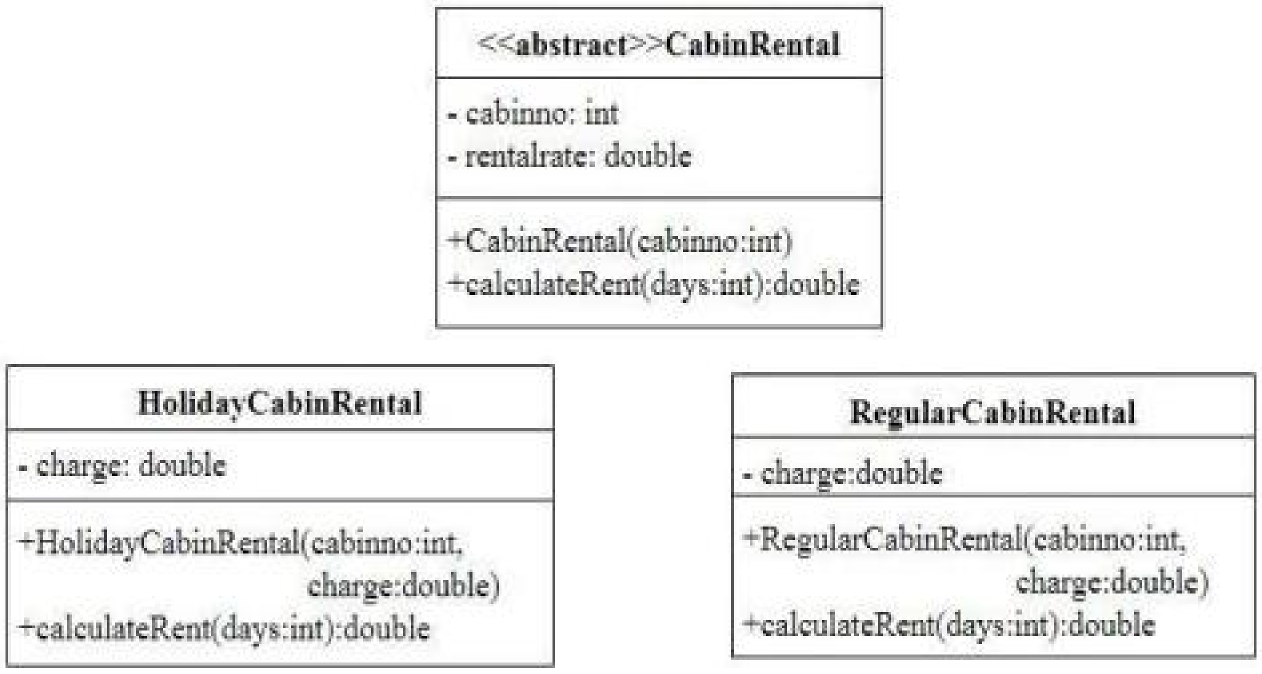
OUTPUT::



```
<terminated> Main [Java Application] C:\Users\niche\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2
Circle[radius=5.5, color=red ,Perimeter:95.03317777109123 ,Area:95.03317777109123]
Rectangle[Area0.0,Perimeter0.0]
Square[Side:9.0]
Circle[radius=2.5, color=yellow ,Perimeter:19.634954084936208 ,Area:19.634954084936208]
Square[Side:16.0]
```

12. Identify the relationship between the classes and write a Java program to implement the relationship. Create a test class to calculate the Total Rent,

Total Rent= ((days*rentalrate)/7)+charge.


<<abstract>>CabinRental
- cabinno: int
- rentalrate: double
+CabinRental(cabinno:int)
+calculateRent(days:int):double
HolidayCabinRental
- charge: double
+HolidayCabinRental(cabinno:int, charge:double)
+calculateRent(days:int):double
RegularCabinRental
- charge:double
+RegularCabinRental(cabinno:int, charge:double)
+calculateRent(days:int):double

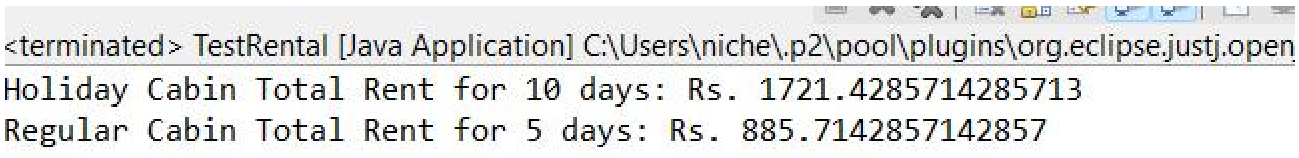
Hint:
1) Rental rate should be assigned based on cabin number. If the cabin number is 1or 2 or 3 then the rental rate is Rs.950 per week. Otherwise Rs.1100 per week.
2) Set HolidayCabinRental Charge as Rs.150 and RegularCabinRental as Rs.100. use getter and setter method wherever necessary.

CODE:

```
Public abstract class CabinRental {
   protected int cabinNo;
protected double rentalRate;
   public CabinRental(int cabinNo) {
      this.cabinNo = cabinNo;
      if (cabinNo == 1 || cabinNo == 2 || cabinNo == 3) {
         this.rentalRate = 950;
      } else {
         this.rentalRate = 1100;}
   }    public abstract double calculateRent(int days);
}
class HolidayCabinRental extends CabinRental {
   private double charge;
   public HolidayCabinRental(int cabinNo) {
      super(cabinNo);
      this.charge = 150;  // Holiday cabin charge}
   public double calculateRent(int days) {
      return ((days * rentalRate) / 7) + charge;}
}
class RegularCabinRental extends CabinRental {
   private double charge;
   public RegularCabinRental(int cabinNo) {
      super(cabinNo);
```

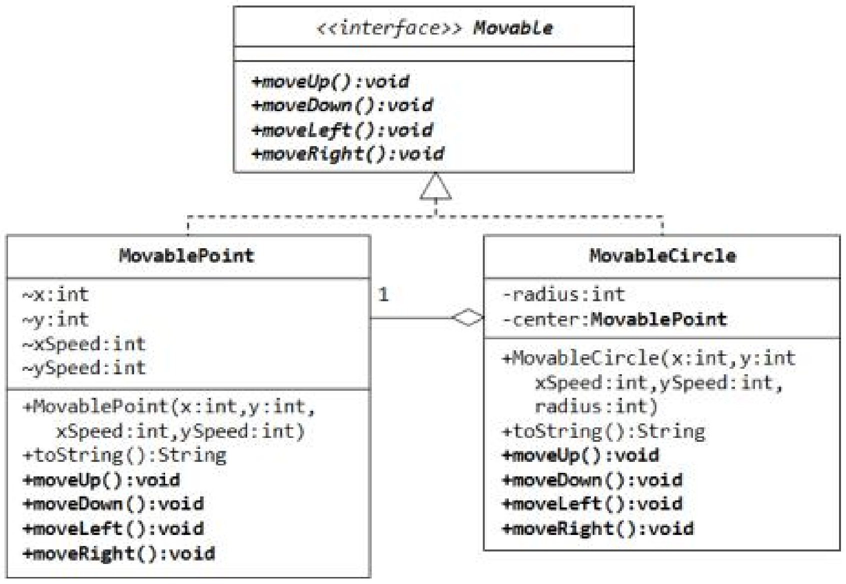
```
        this.charge = 100;  // Regular cabin charge; }
    public double calculateRent(int days) {
        return ((days * rentalRate) / 7) + charge;}
}
public class CabinRentalTest {
    public static void main(String[] args) {
        CabinRental holidayCabin = new HolidayCabinRental(10);
        System.out.println("Holiday Cabin Total Rent for 10 days: Rs. " + holidayCabin.calculateRent(10));
        CabinRental regularCabin = new RegularCabinRental(5);
        System.out.println("Regular Cabin Total Rent for 5 days: Rs. " + regularCabin.calculateRent(5));
    }}
```

OUTPUT::



```
<terminated> TestRental [Java Application] C:\Users\niche\.p2\pool\plugins\org.eclipse.justj.open
Holiday Cabin Total Rent for 10 days: Rs. 1721.4285714285713
Regular Cabin Total Rent for 5 days: Rs. 885.7142857142857
```

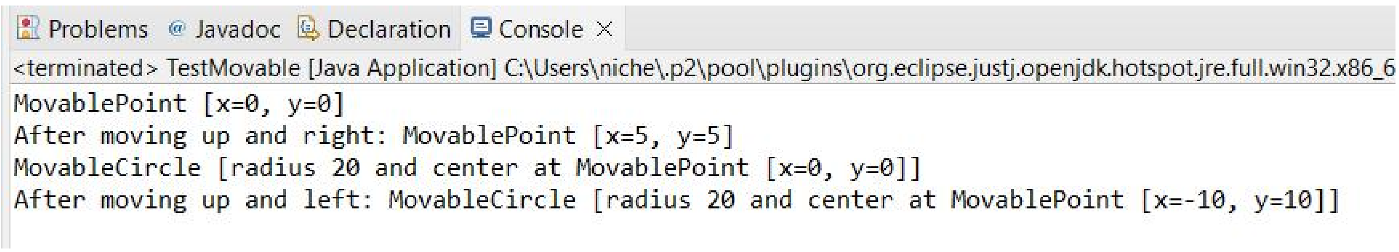
13. Write a Java program to implement the following relationship and create a Main class to invoke all the methods.



CODE:

```
Public interface Movable {
    void moveUp();
    void moveDown();
    void moveLeft();
    void moveRight();}
class MovablePoint implements Movable {
    int x, y;
    int xSpeed, ySpeed;
    public MovablePoint(int x, int y, int xSpeed, int ySpeed) {
        this.x = x;
        this.y = y;
        this.xSpeed = xSpeed;
        this.ySpeed = ySpeed;}
    public void moveUp() {
        y += ySpeed;}
    public void moveDown() {
        y -= ySpeed;}
    public void moveLeft() {
        x -= xSpeed;}
    public void moveRight() {
        x += xSpeed;}
    public String toString() {
        return "MovablePoint at (" + x + ", " + y + ") with speed (" + xSpeed + ", " + ySpeed + ")"; }
}
class MovableCircle implements Movable {
    private int radius;
    private MovablePoint center;
    public MovableCircle(int x, int y, int xSpeed, int ySpeed, int radius) {
        this.center = new MovablePoint(x, y, xSpeed, ySpeed);
        this.radius = radius;}
    public void moveUp() {
        center.moveUp();}
    public void moveDown() {
        center.moveDown();}
    public void moveLeft() {
        center.moveLeft();}
    public void moveRight() {
        center.moveRight();}
    public String toString() {
        return "MovableCircle [ radius " + radius + " and center at " + center.toString()+"]";
    }}
public class Main {
    public static void main(String[] args) {
        MovablePoint point = new MovablePoint(0, 0, 5, 5);
        System.out.println(point);
        point.moveUp();
        point.moveRight();
        System.out.println("After moving up and right: " + point);
        MovableCircle circle = new MovableCircle(0, 0, 10, 10, 20);
        System.out.println(circle);
        circle.moveUp();
        circle.moveLeft();
        System.out.println("After moving up and left: " + circle);}}
```

OUTPUT::



Problems @ Javadoc Declaration Console

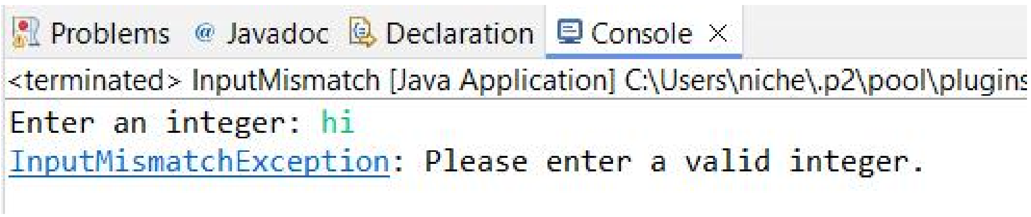
```
<terminated> TestMovable [Java Application] C:\Users\niche\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_6
MovablePoint [x=0, y=0]
After moving up and right: MovablePoint [x=5, y=5]
MovableCircle [radius 20 and center at MovablePoint [x=0, y=0]]
After moving up and left: MovableCircle [radius 20 and center at MovablePoint [x=-10, y=10]]
```

14. Write Java programs to handle the following exceptions:
a. InputMismatchException
CODE:

```
package dhanush;
import java.util.*;
public class InputMismatch {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
try { System.out.print("Enter an integer: ");
int num = sc.nextInt();
System.out.println("You entered: " + num);
} catch (InputMismatchException e) {
System.out.println("InputMismatchException: Please enter a valid integer.");
```

}}}

OUTPUT::



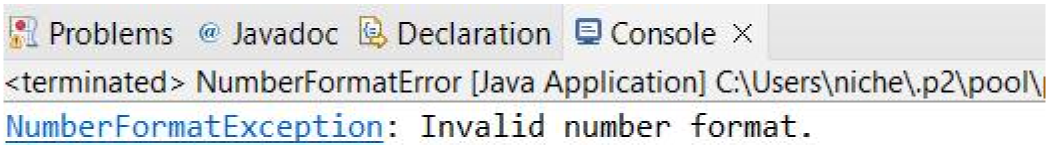
```
<terminated> InputMismatch [Java Application] C:\Users\niche\.p2\pool\plugins
Enter an integer: hi
InputMismatchException: Please enter a valid integer.
```

b. NumberFormatException

CODE:

```
package dhanush;
public class NumberFormatError {
            public static void main(String[] args) {
               String num = "abc";
               try {
                  int num1 = Integer.parseInt(num);
                  System.out.println("Converted number: " + num);
               } catch (NumberFormatException e) {
                  System.out.println("NumberFormatException: Invalid number format.");  }  }}
```

OUTPUT::



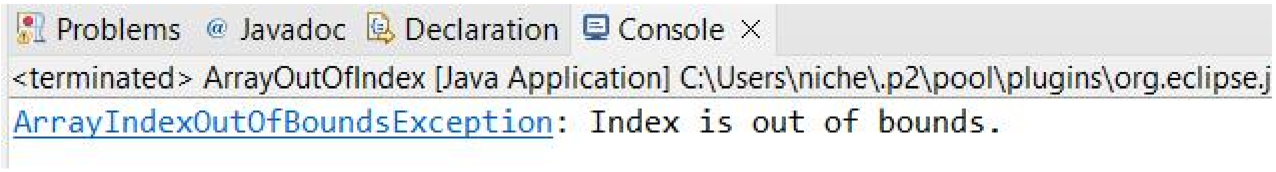
```
<terminated> NumberFormatError [Java Application] C:\Users\niche\.p2\pool\
NumberFormatException: Invalid number format.
```

c. ArrayIndexOutofBoundsException

CODE:

```
public class ArrayOutOfIndex {
            public static void main(String[] args) {
               int[] numbers = {1, 2, 3};
               try {
                  System.out.println("Accessing element at index 3: " + numbers[3]);
               } catch (ArrayIndexOutOfBoundsException e) {
                  System.out.println("ArrayIndexOutOfBoundsException: Index is out of bounds.");
               }}}
```

OUTPUT::



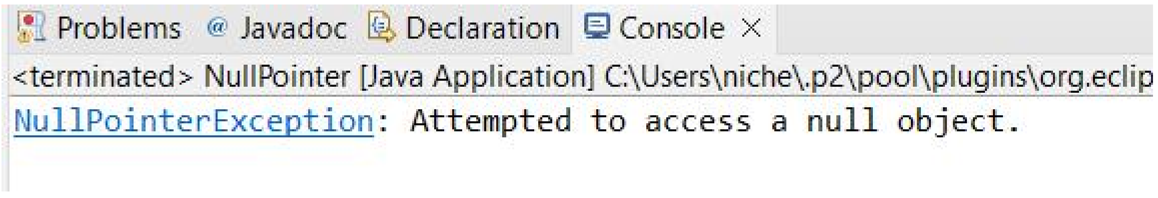
```
<terminated> ArrayOutOfIndex [Java Application] C:\Users\niche\.p2\pool\plugins\org.eclipse.j
ArrayIndexOutOfBoundsException: Index is out of bounds.
```

d. NullPointerException

CODE:

```
public class NullPointer {
            public static void main(String[] args) {
               String str = null;
               try {
                  System.out.println("Length of the string: " + str.length());
               } catch (NullPointerException e) {
                  System.out.println("NullPointerException: Attempted to access a null object."); }  }}
```

OUTPUT::



```
<terminated> NullPointer [Java Application] C:\Users\niche\.p2\pool\plugins\org.eclip
NullPointerException: Attempted to access a null object.
```

15. Write a Java program based on the following statements: • Create a CourseException class that extends Exception and whose constructor receives a String that holds a college course's department (for example, CIS), a course number (for example, 101), and a number of credits (for example, 3). Save the file as CourseException.java.

• Create a Course class with the same fields and whose constructor requires values for each field. Upon construction, throw a CourseException if the department does not consist of three letters, if the course number does not consist of three digits between 100 and 499 inclusive, or if the credits are less than 0.5 or more than 6. Save the class as Course.java.

• Write an application that establishes an array of at least six Course objects with valid and invalid values. Display an appropriate message when a Course object is created successfully and when one is not.

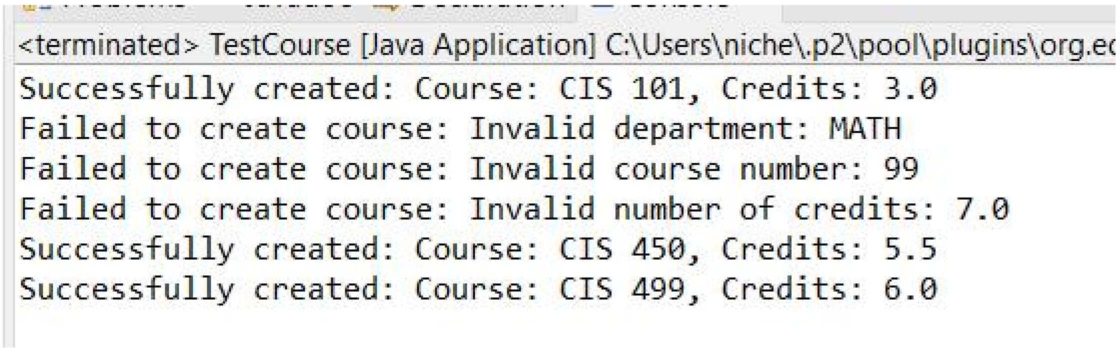
CODE:

```
package dhanush;
public class CourseException extends Exception {
            public CourseException(String message) {
               super(message);}}
package dhanush;

public class Course {
            private String department;
            private int courseNumber;
            private double credits;
            public Course(String department, int courseNumber, double credits) throws CourseException {
               if (department.length() != 3 || !department.matches("[a-zA-Z]+")) {
                  throw new CourseException("Invalid department: " + department);
               }
               if (courseNumber < 100 || courseNumber > 499) {
                  throw new CourseException("Invalid course number: " + courseNumber);
               }
               if (credits < 0.5 || credits > 6.0) {
                  throw new CourseException("Invalid number of credits: " + credits);
               }
               this.department = department;
               this.courseNumber = courseNumber;
               this.credits = credits;
            }
            public String toString() {
               return "Course: " + department + " " + courseNumber + ", Credits: " + credits;
            }
         }
package dhanush;

public class TestCourse {
            public static void main(String[] args) {
               Course[] courses = new Course[6];
         String[][] courseData = {
                  {"CIS", "101", "3"},
                  {"MATH", "200", "4"},
                  {"ENG", "99", "3"},
                  {"BIO", "305", "7"},
                  {"CIS", "450", "5.5"},
                  {"CIS", "499", "6"}
               };
            for (int i = 0; i < courseData.length; i++) {
                try {
                    String department = courseData[i][0];
                    int courseNumber = Integer.parseInt(courseData[i][1]);
                    double credits = Double.parseDouble(courseData[i][2]);
                    courses[i] = new Course(department, courseNumber, credits);
                    System.out.println("Successfully created: " + courses[i]);
                } catch (CourseException e) {
                    System.out.println("Failed to create course: " + e.getMessage());
                }
            }
        }
    }
```

OUTPUT::



```
<terminated> TestCourse [Java Application] C:\Users\niche\.p2\pool\plugins\org.ec
Successfully created: Course: CIS 101, Credits: 3.0
Failed to create course: Invalid department: MATH
Failed to create course: Invalid course number: 99
Failed to create course: Invalid number of credits: 7.0
Successfully created: Course: CIS 450, Credits: 5.5
Successfully created: Course: CIS 499, Credits: 6.0
```

RESULT

Thus, the java program was executed successfully and the output was verified.