JAVA PROGRAMS

PRACTICAL 1

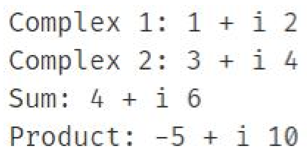
Objective
Design a class Complex having a real part (x) and an imaginary part (y). Provide methods to perform the following on complex numbers:

1. Add two complex numbers.
2. Multiply two complex numbers.
3. toString() method to display complex numbers in the form: x + i y

```
/**** Complex.java ****/
public class Complex {
   private int x;
   private int y;
   /**
    * Parameterized Constructor of Complex class
    *
    * @param real      Real Part
    * @param imaginary Imaginary Part
    */
   public Complex(int real, int imaginary) {
      this.x = real;
      this.y = imaginary;
   }
   /**
    * Add two Complex Objects
    *
    * @param o Complex Object
    * @return Complex Object
    */
   public Complex add(Complex o) {
      return new Complex(
            this.x + o.x,
            this.y + o.y
      );
   }

   /**
    * Multiply two Complex Objects
    *
    * @param o Complex Object
    * @return Complex Object
    */
   public Complex multiply(Complex o) {
      return new Complex(
            this.x * o.x - this.y * o.y,
            this.x * o.y + o.x * this.y
```

```
        );
    }
    /**
     * Type Conversion to String
     *
     * @return String Representation
     */
    @Override
    public String toString() {
        return x + " + i " + y;
    }
}

/**** Main.java ****/
public class Main {
    public static void main(String[] args) {
        Complex c1 = new Complex(1, 2);
        Complex c2 = new Complex(3, 4);
        System.out.println("Complex 1: " + c1);
        System.out.println("Complex 2: " + c2);
        System.out.println("Sum: " + c1.add(c2));
        System.out.println("Product: " + c1.multiply(c2));
    }
}
```

Output



```
Complex 1: 1 + i 2
Complex 2: 3 + i 4
Sum: 4 + i 6
Product: -5 + i 10
```

PRACTICAL 2

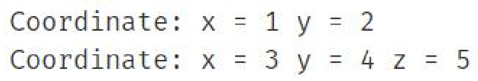
Objective

Create a class TwoDim which contains private members as x and y coordinates in package P1. Define the default constructor, a parameterized constructor and override toString() method to display the co-ordinates. Now reuse this class and in package P2 create another class ThreeDim, adding a new dimension as z as its private member. Define the constructors for the subclass and override toString() method in the subclass also. Write appropriate methods to show dynamic method dispatch. The main() function should be in a package P.

```
/**** P1/TwoDim.java ****/
package P1;
public class TwoDim {
   private int x;
   private int y;
   public TwoDim() {
      this.x = 0;
      this.y = 0;
   }
   public TwoDim(int x, int y) {
      this.x = x;
      this.y = y;
   }
   @Override
   public String toString() {
      return "Coordinate: x = " + x + " y = " + y;
   }
}

/**** P2/ThreeDim.java ****/
package P2;
import P1.*;
public class ThreeDim extends TwoDim {
   private int z;
   public ThreeDim() {
      super(0, 0);
      this.z = 0;
   }
   public ThreeDim(int x, int y, int z) {
      super(x, y);
      this.z = z;
   }
   @Override
   public String toString() {
      return super.toString() + " z = " + z;
   }
}

/**** P/Main.java ****/
package P;
import P1.*;
import P2.*;
public class Main {
   public static void main(String[] args) {
      TwoDim ref;
      ref = new TwoDim(1, 2);
      System.out.println(ref);
```

```
        ref = new ThreeDim(3, 4, 5);
        System.out.println(ref);
    }
}
```

Output



```
Coordinate: x = 1 y = 2
Coordinate: x = 3 y = 4 z = 5
```

PRACTICAL 3

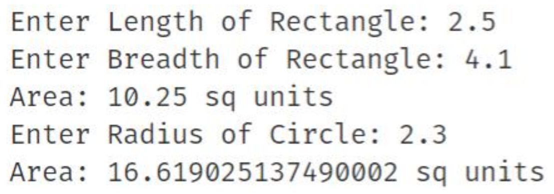
Objective

Define an abstract class Shape in package P1. Inherit two more classes: Rectangle in package P2 and Circle in package P3. Write a program to ask the user for the type of shape and then using the concept of dynamic method dispatch, display the area of the appropriate subclass. Also write appropriate methods to read the data. The main() function should not be in any package.

```
/**** P1/Shape.java ****/
package P1;
public abstract class Shape {
    protected abstract void getData() throws java.io.IOException;
    public abstract double area() throws java.io.IOException;
}

/**** P2/Rectangle.java ****/
package P2;
import java.io.*;
import P1.*;
public class Rectangle extends Shape {
    private double length;
    private double breadth;
    protected void getData() throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(
                    System.in
            ));
        System.out.print("Enter Length of Rectangle: ");
        length = Double.parseDouble(br.readLine());
        System.out.print("Enter Breadth of Rectangle: ");
        breadth = Double.parseDouble(br.readLine());
    }
    public double area() throws IOException {
        getData();
        return length * breadth;
    }
}
/**** P3/Circle.java ****/
package P3;
import java.io.*;
import P1.*;
public class Circle extends Shape {
    private double radius;
    protected void getData() throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(
                    System.in
            ));
        System.out.print("Enter Radius of Circle: ");
        radius = Double.parseDouble(br.readLine());
    }
    public double area() throws IOException {
        getData();
        return Math.PI * radius * radius;
    }
}

/**** Main.java ****/
import java.io.*;
import P1.*;
import P2.*;
import P3.*;
public class Main {
    static int getShapeType() throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(
                    System.in
            ));
        System.out.println("=============\n  SHAPE TYPE  \n=============");
        System.out.println(" (1) Rectangle\n (2) Circle");
        System.out.print("Enter Choice: ");
        return Integer.parseInt(br.readLine());
    }
    public static void main(String[] args) throws IOException {
        Shape ref;
        boolean flag = false;
        while (!flag) {
            switch (getShapeType()) {
                case 1:
                    flag = true;
                    ref = new Rectangle();
                    System.out.println("Area: " + ref.area() + " sq units");
                    break;
                case 2:
```

```
                flag = true;
                ref = new Circle();
                System.out.println("Area: " + ref.area() + " sq units");
                break;
            default:
                System.err.println("Invalid Option");
                break;
        }
    }
  }
}
```

Output



```
Enter Length of Rectangle: 2.5
Enter Breadth of Rectangle: 4.1
Area: 10.25 sq units
Enter Radius of Circle: 2.3
Area: 16.619025137490002 sq units
```

PRACTICAL 4

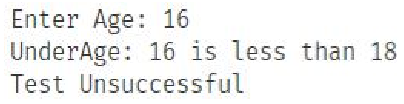
Objective

Create an Exception subclass UnderAge, which prints “Under Age” along with the age value when an object of UnderAge class is printed in the catch statement. Write a class exceptionDemo in which the method test() throws UnderAge exception if the variable age passed to it as argument is less than 18. Write main() method also to show working of the program.

```
/**** UnderAge.java ****/
public class UnderAge extends Exception {
    final private int age;
    public UnderAge(int age) {
        this.age = age;
    }
    @Override
    public String getMessage() {
        return "UnderAge: " + age + " is less than 18";
    }
}

/**** exceptionDemo.java ****/
import java.util.Scanner;
class exceptionDemo {
    static void test(int age) throws UnderAge {
        if (age < 18)
            throw new UnderAge(age);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Age: ");
        int age = sc.nextInt();
        try {
            test(age);
            System.out.println("Test Successful");
        } catch (UnderAge e) {
            System.err.println(e.getMessage());
            System.out.println("Test Unsuccessful");
        } finally {
            sc.close();
        }
    }
}
```

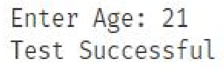
Output

Case 1: Age is lesser than 18



```
Enter Age: 16
UnderAge: 16 is less than 18
Test Unsuccessful
```

Case 2: Age is not lesser than 18



```
Enter Age: 21
Test Successful
```

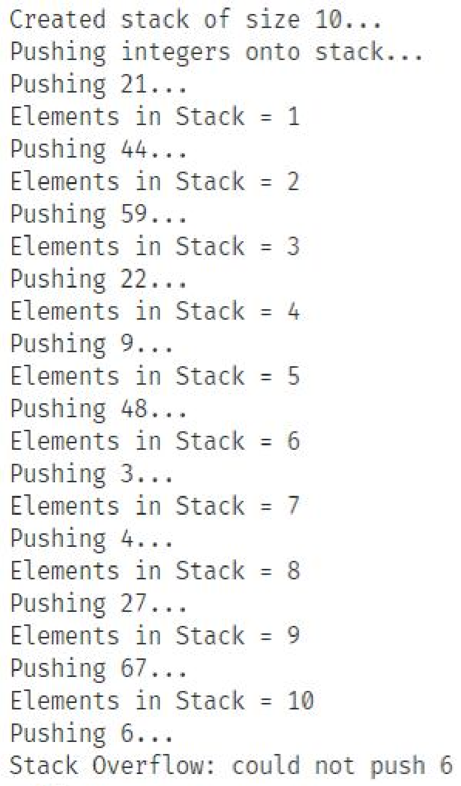
PRACTICAL 5

Objective

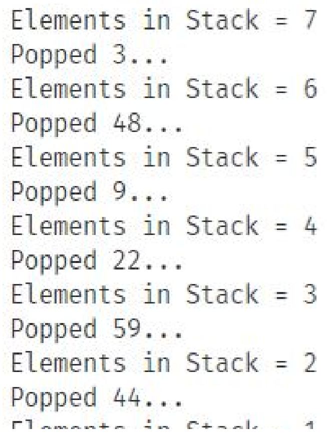
Write a program to implement stack. Use exception handling to manage underflow and overflow conditions.

```
/**** Stack.java ****/
public class Stack {
    // Top of the Stack
    private int tos;

    // Array of Elements
    private int[] array;

    // Size of the Stack
    final private int size;
    /**
     * Public Constructor for Stack Objects
     *
     * @param size   Size of the Stack
     */
    public Stack(int size) {
       this.tos = -1;
       this.size = size;
       this.array = new int[this.size];
    }

    /**
     * Push an element to the top of the stack
     *
     * @param e Element to be pushed
     * @throws StackException Stack Overflow
     */
    public void push(int e) throws StackException {
       if (tos == size - 1)
          throw new StackException("Stack Overflow: could not push " + e);
       else
          this.array[++this.tos] = e;
    }

    /**
     * Pop an element from the stack
     *
     * @return Object on top of the stack
     * @throws StackException Stack Underflow
     */
    public int pop() throws StackException {
       if (this.tos < 0) {
          throw new StackException("Stack Underflow:could not pop");
       } else
          return this.array[this.tos--];
    }

  /**
     * Index of the element at the top of the stack
     *
     * @return Index of Generic Element
     */
    public int getTOS() {
       return this.tos;
    }

    /**
     * Representation of Stack Object
     *
     * @return String Representation
     */
    @Override
    public String toString() {
        return "Stack<size=" + this.size + ">";
    }
}

/**** StackException.java ****/
public class StackException extends Exception {
    final private String message;
    public StackException(String message) {
        this.message = message;
    }
    @Override
    public String getMessage() {
        return this.message;
    }
}



/**** Main.java ****/
import java.util.Random;
public class Main {
    public static void main(String[] args) {
        int r;
        Stack stack = new Stack(10);
        Random random = new Random(1337);
        System.out.println("Created stack of size 10...");
        System.out.println("Pushing integers onto stack...");
        while (true) {
            r = random.nextInt(100);
            System.out.println("Pushing " + r + "...");
            try {
                stack.push(r);
                System.out.println(
"Elements in Stack = " + (stack.getTOS() +1)
);
            } catch (StackException e) {
                System.err.println(e.getMessage());
                break;
            }
        }
        System.out.println("Popping integers from stack...");
        while (true) {
            System.out.println(
"Elements in Stack = " + (stack.getTOS() +1)
);
            try {
                System.out.println("Popped " + stack.pop() + "...");
            } catch (StackException e) {
                System.err.println(e.getMessage());
                break;
            }
        }
    }
}
```

Output



```
Created stack of size 10...
Pushing integers onto stack...
Pushing 21...
Elements in Stack = 1
Pushing 44...
Elements in Stack = 2
Pushing 59...
Elements in Stack = 3
Pushing 22...
Elements in Stack = 4
Pushing 9...
Elements in Stack = 5
Pushing 48...
Elements in Stack = 6
Pushing 3...
Elements in Stack = 7
Pushing 4...
Elements in Stack = 8
Pushing 27...
Elements in Stack = 9
Pushing 67...
Elements in Stack = 10
Pushing 6...
Stack Overflow: could not push 6
```



```
Elements in Stack = 7
Popped 3...
Elements in Stack = 6
Popped 48...
Elements in Stack = 5
Popped 9...
Elements in Stack = 4
Popped 22...
Elements in Stack = 3
Popped 59...
Elements in Stack = 2
Popped 44...
Elements in Stack = 1
```

PRACTICAL 6

Objective

Write a program that copies content of one file to another. Pass the names of the files through command-line arguments.

```
/**** Copy.java ****/
import java.io.*;
public class Copy {
   public static void main(String[] args) throws Exception {
      if (args.length != 2) {
         System.err.println("Usage: java Copy <src> <dest>");
      } else {
         int i;
         FileInputStream fin = new FileInputStream(args[0]);
         FileOutputStream fout = new FileOutputStream(args[1]);
         while ((i = fin.read()) != -1) {
            fout.write(i);
         }
         fin.close();
         fout.close();
         System.out.println(
"Copied contents of" + args[0] + " to " + args[1]
);
      }
   }
}
```

References/Resources:

1. Balaguruswamy, E. (2014). Programming with JAVA: A Primer. 5th edition. India: McGraw Hill Education
2. Horstmann, C. S. (2017). Core Java - Vol. I – Fundamentals (Vol. 10). Pearson Education
3. Schildt, H. (2018). Java: The Complete Reference. 10th edition. McGraw-Hill Education.

NOTE: Please go through the above programs carefully and practice them (on machine if possible).