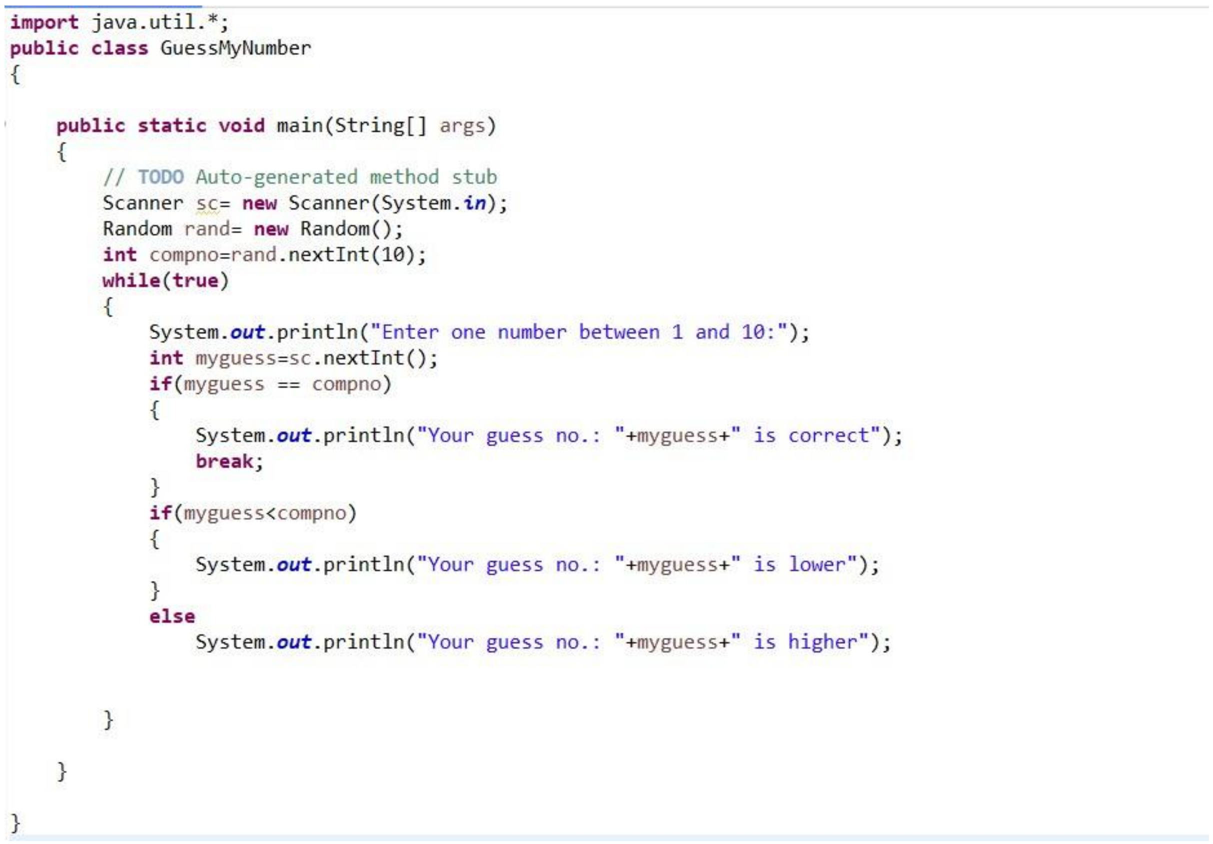
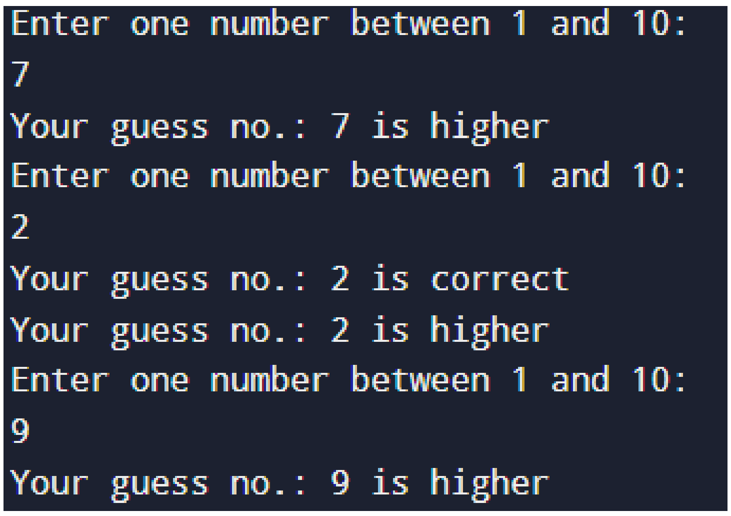
OOP LAB PROGRAMS

PART A

1. Build a program called "GuessMyNumber." The computer will generate a random number between 1 and 10. The user types in a number, and the computer replies “lower” if the random number is lower than the guess, “higher” if the random number is higher , and “correct!” if the guess is correct. The player can continue guessing until the guess is right.



```java
import java.util.*;
public class GuessMyNumber
{

    public static void main(String[] args)
    {
        // TODO Auto-generated method stub
        Scanner sc= new Scanner(System.in);
        Random rand= new Random();
        int compno=rand.nextInt(10);
        while(true)
        {
            System.out.println("Enter one number between 1 and 10:");
            int myguess=sc.nextInt();
            if(myguess == compno)
            {
                System.out.println("Your guess no.: "+myguess+" is correct");
                break;
            }
            if(myguess<compno)
            {
                System.out.println("Your guess no.: "+myguess+" is lower");
            }
            else
                System.out.println("Your guess no.: "+myguess+" is higher");


        }

    }

}
```



```
Enter one number between 1 and 10:
7
Your guess no.: 7 is higher
Enter one number between 1 and 10:
2
Your guess no.: 2 is correct
Your guess no.: 2 is higher
Enter one number between 1 and 10:
9
Your guess no.: 9 is higher
```

2. Create a Java class called Student with the following details as private instance variables within it.

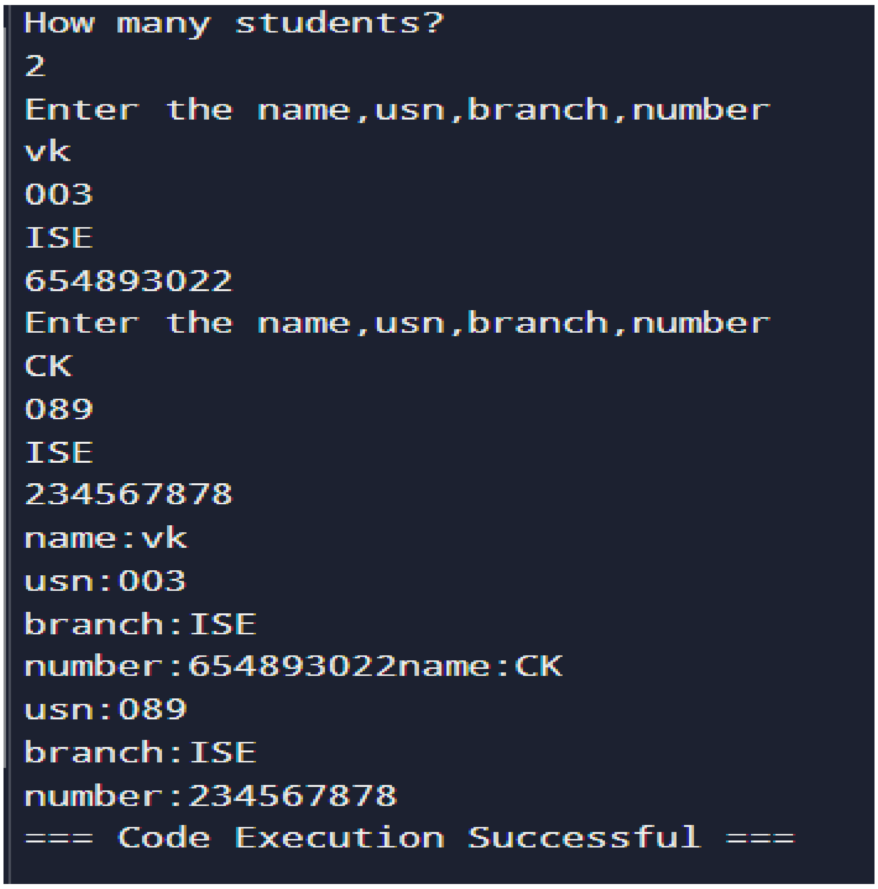
USN,Name,Branch, PhoneNo.

Write a Java program to create ‘n’ number student objects and print the USN, name, branch, and phone number of these objects with suitable headings.

```
import java.util.Scanner; class
Student
{
        String name,usn,branch,number;
public Student()
        {
                name=" ";
usn=" ";
branch=" ";
number=" ";
        }
        public Student(String name,String usn,String branch,String number)
        {
                this.name=name;
this.usn=usn;           this.branch=branch;
        this.number=number;
        }

        void display()
        {
                System.out.print("name:"+name+"");
                System.out.print("usn:"+usn+"");
                System.out.print("branch:"+branch+"");
System.out.print("number:"+number+"");
```

```
		}
}

public class StudentDemo
{

		public static void main(String[] args)
		{
				// TODO Auto-generated method stub
				Scanner read=new Scanner(System.in);
System.out.println("How many students?");			int
size=read.nextInt();			Student objs[]=new
Student [size];			read=new Scanner(System.in);
for(int i=0;i<objs.length;i++)
				{
						String name,usn,branch,number;
						System.out.println("Enter the name,usn,branch,number");
						name=read.nextLine();
usn=read.nextLine();				branch=read.nextLine();
				number=read.nextLine();
						Student obj=new Student(name,usn,branch,number);
						objs[i]=obj;

				}
				for(int i=0;i<objs.length;i++)
				{
						objs[i].display();

				}
```

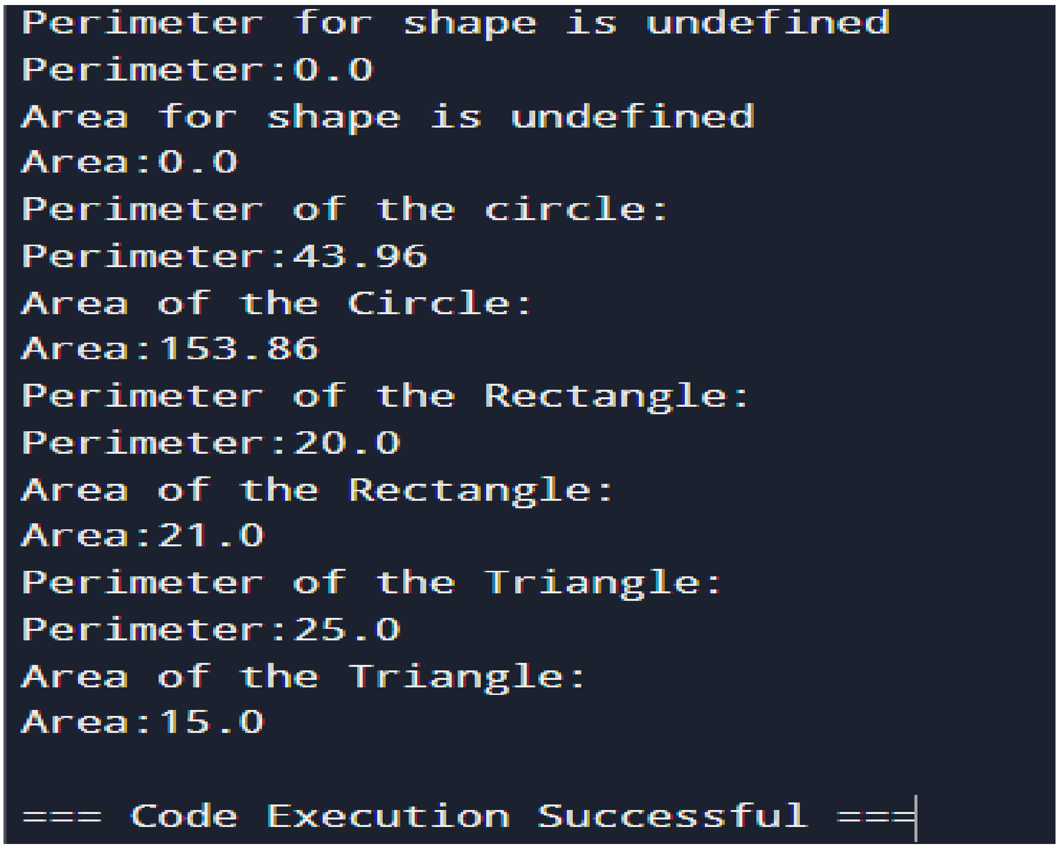
```
        }

}
```



```
How many students?
2
Enter the name,usn,branch,number
vk
003
ISE
654893022
Enter the name,usn,branch,number
CK
089
ISE
234567878
name:vk
usn:003
branch:ISE
number:654893022name:CK
usn:089
branch:ISE
number:234567878
=== Code Execution Successful ===
```

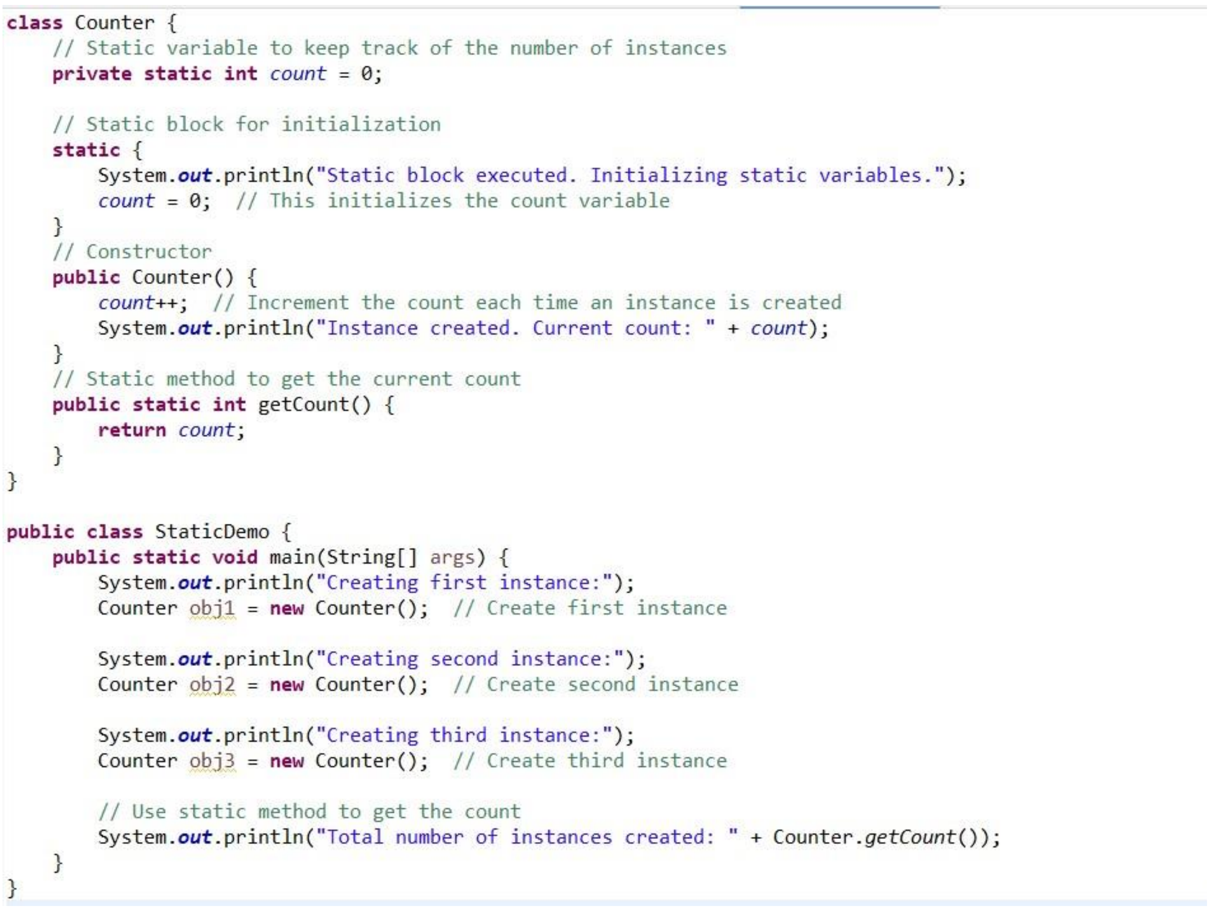
3. Write a Java program to create a class called Shape with methods called getPerimeter() and getArea(). Create a subclass called Circle, Rectangle, and Triangle that overrides the getPerimeter() and getArea() methods to calculate the area and perimeter of all the subclasses.

```
class Shape{

        double dim1;

double dim2;    double

dim3;   double dim4;

        Shape(double p, double q, double r, double h){

                dim1=p;

dim2=q;                 dim3=r;

        dim4=h;

        }

        double getPerimeter() {

                System.out.println("Perimeter for shape is undefined");

                return 0;

        }

        double getArea() {

                System.out.println("Area for shape is undefined");

                return 0;

        }

}


class Circle extends Shape{

        Circle(double a){

                super(a,0,0,0);

        }

        double getPerimeter() {

                System.out.println("Perimeter of the circle:");

                return 2*3.14*dim1;

        }

        double getArea() {
                System.out.println("Area of the Circle:");

                return 3.14*dim1*dim1;

        }

}


class Rectangle extends Shape{

        Rectangle(double a,double b){

                super(a,b,0,0);

        }

        double getPerimeter() {

                System.out.println("Perimeter of the Rectangle:");

                return 2*(dim1+dim2);

        }

        double getArea() {

                System.out.println("Area of the Rectangle:");

                return dim1*dim2;

        }

}


class Triangle extends Shape{

        Triangle(double a, double b, double c, double height){

                super(a,b,c,height);

        }

        double getPerimeter() {

                System.out.println("Perimeter of the Triangle:");

                return dim1+dim2+dim3;

        }

        double getArea() {

         System.out.println("Area of the Triangle:");  return

        0.5*dim2*dim4;
        }
}

public class Dispatch{  public static void
main(String[] args) {
                Shape S=new Shape(10,10,10,10);
                Circle C=new Circle(4);
                Rectangle R=new Rectangle(5,7);
                Triangle T=new Triangle(9,8,7,6);
                Shape figref;

                figref=S;
                System.out.println("Perimeter:"+figref.getPerimeter());
                System.out.println("Area:"+figref.getArea());

                figref=C;
                System.out.println("Perimeter:"+figref.getPerimeter());
                System.out.println("Area:"+figref.getArea());

                figref=R;
                System.out.println("Perimeter:"+figref.getPerimeter());
                System.out.println("Area:"+figref.getArea());

                figref=T;
                System.out.println("Perimeter:"+figref.getPerimeter());
                System.out.println("Area:"+figref.getArea());
        }
}
```

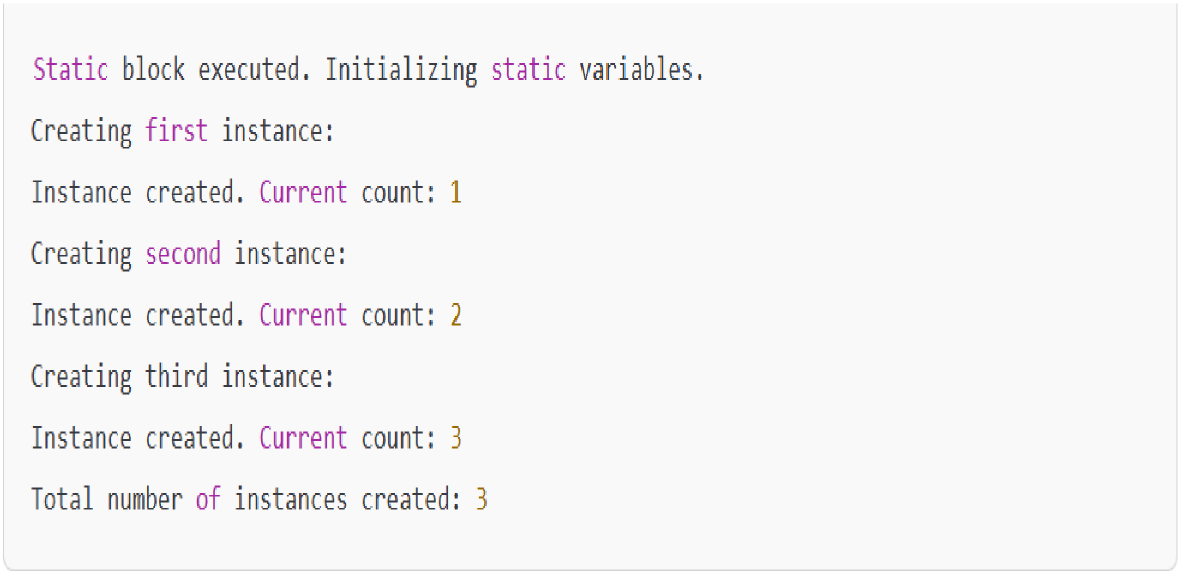


```
Perimeter for shape is undefined
Perimeter:0.0
Area for shape is undefined
Area:0.0
Perimeter of the circle:
Perimeter:43.96
Area of the Circle:
Area:153.86
Perimeter of the Rectangle:
Perimeter:20.0
Area of the Rectangle:
Area:21.0
Perimeter of the Triangle:
Perimeter:25.0
Area of the Triangle:
Area:15.0

=== Code Execution Successful ===
```

4. Write a Java program to demonstrate static variables, methods and blocks.

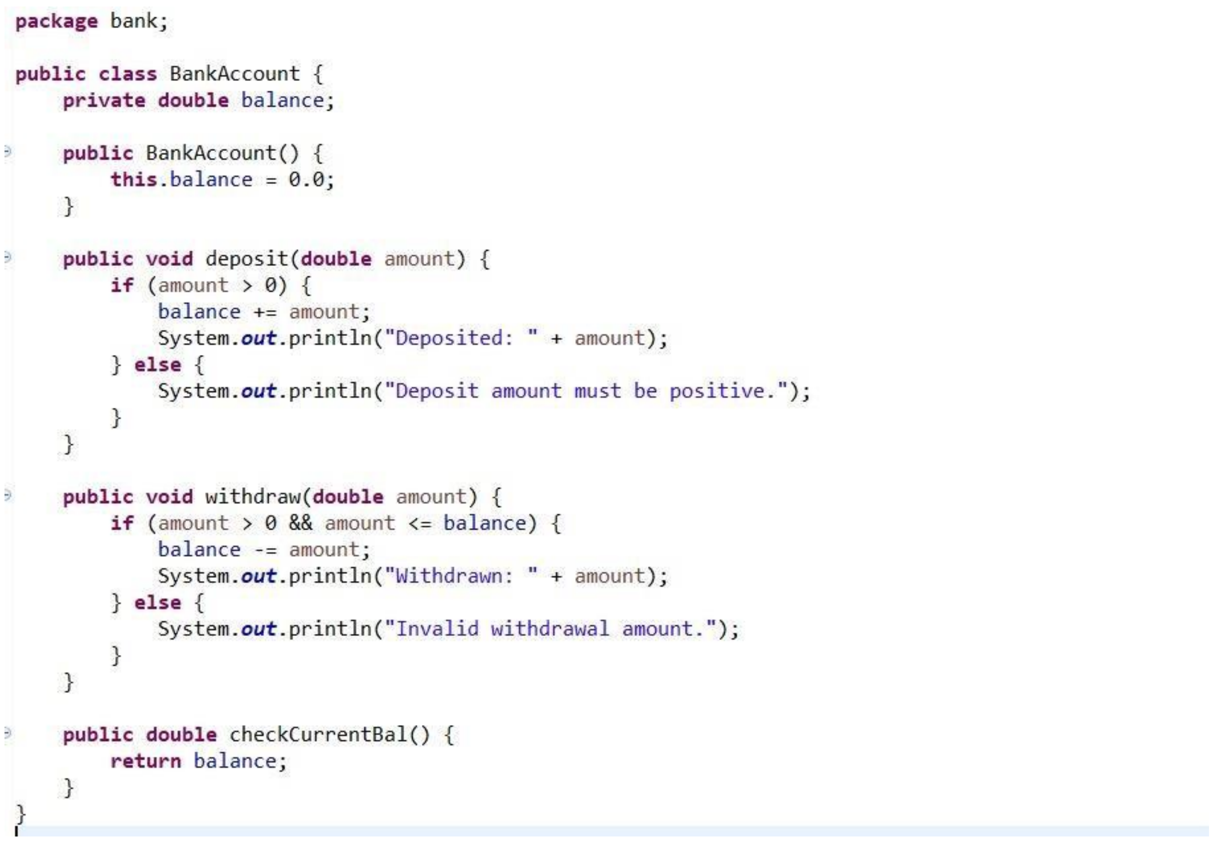


```
class Counter {
    // Static variable to keep track of the number of instances
    private static int count = 0;

    // Static block for initialization
    static {
        System.out.println("Static block executed. Initializing static variables.");
        count = 0;  // This initializes the count variable
    }
    // Constructor
    public Counter() {
        count++;  // Increment the count each time an instance is created
        System.out.println("Instance created. Current count: " + count);
    }
    // Static method to get the current count
    public static int getCount() {
        return count;
    }
}

public class StaticDemo {
    public static void main(String[] args) {
        System.out.println("Creating first instance:");
        Counter obj1 = new Counter();  // Create first instance

        System.out.println("Creating second instance:");
        Counter obj2 = new Counter();  // Create second instance

        System.out.println("Creating third instance:");
        Counter obj3 = new Counter();  // Create third instance

        // Use static method to get the count
        System.out.println("Total number of instances created: " + Counter.getCount());
    }
}
```

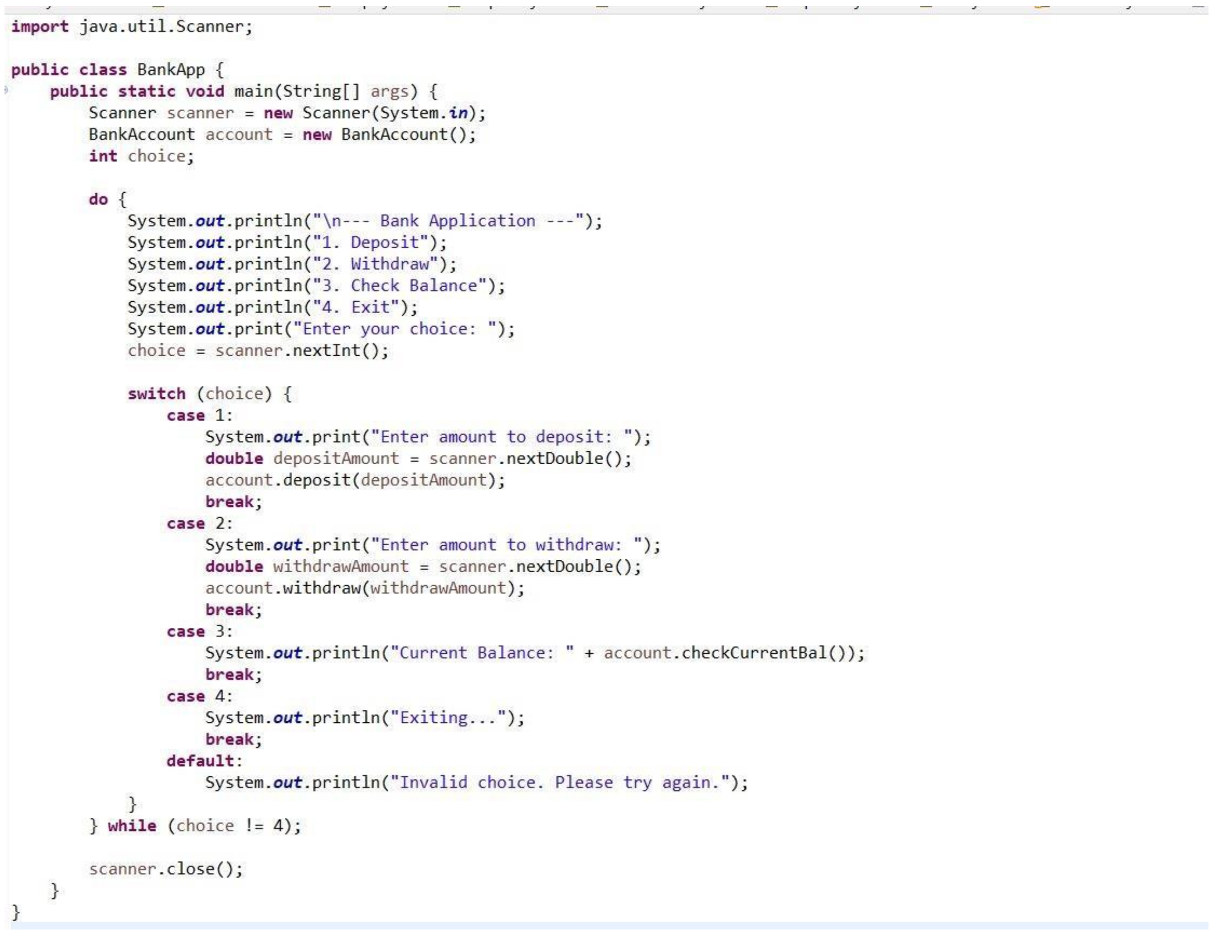


```
Static block executed. Initializing static variables.

Creating first instance:

Instance created. Current count: 1

Creating second instance:

Instance created. Current count: 2

Creating third instance:

Instance created. Current count: 3

Total number of instances created: 3
```

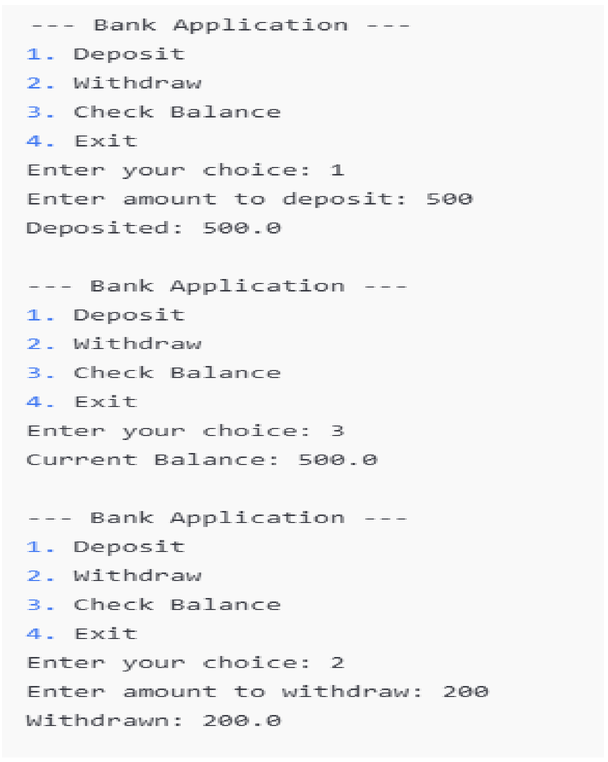
5. Develop a Java application to implement bank applications that perform deposit() and withdraw() operations using packages. Also write a checkCurrentBal() function to verify the amount credited and debited based on the operations performed.



```java
package bank;

public class BankAccount {
    private double balance;

    public BankAccount() {
        this.balance = 0.0;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
        } else {
            System.out.println("Deposit amount must be positive.");
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Invalid withdrawal amount.");
        }
    }

    public double checkCurrentBal() {
        return balance;
    }
}
```

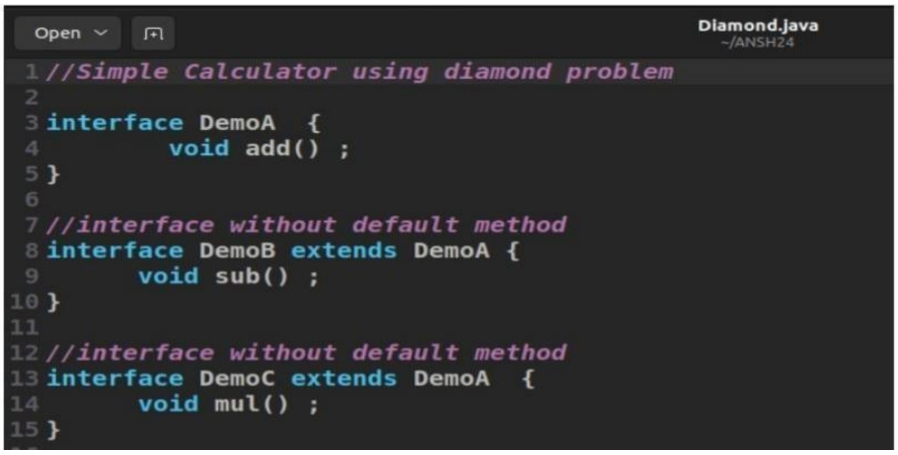


```java
import java.util.Scanner;

public class BankApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        BankAccount account = new BankAccount();
        int choice;

        do {
            System.out.println("\n--- Bank Application ---");
            System.out.println("1. Deposit");
            System.out.println("2. Withdraw");
            System.out.println("3. Check Balance");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter amount to deposit: ");
                    double depositAmount = scanner.nextDouble();
                    account.deposit(depositAmount);
                    break;
                case 2:
                    System.out.print("Enter amount to withdraw: ");
                    double withdrawAmount = scanner.nextDouble();
                    account.withdraw(withdrawAmount);
                    break;
                case 3:
                    System.out.println("Current Balance: " + account.checkCurrentBal());
                    break;
                case 4:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 4);

        scanner.close();
    }
}
```

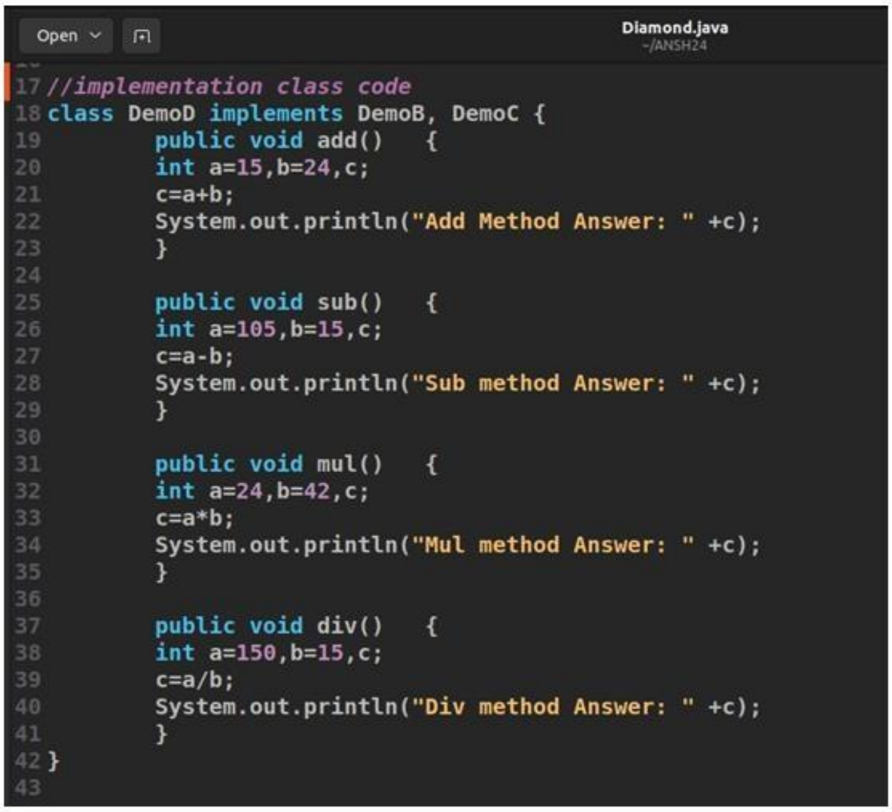


```
--- Bank Application ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice: 1
Enter amount to deposit: 500
Deposited: 500.0

--- Bank Application ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice: 3
Current Balance: 500.0

--- Bank Application ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice: 2
Enter amount to withdraw: 200
Withdrawn: 200.0
```

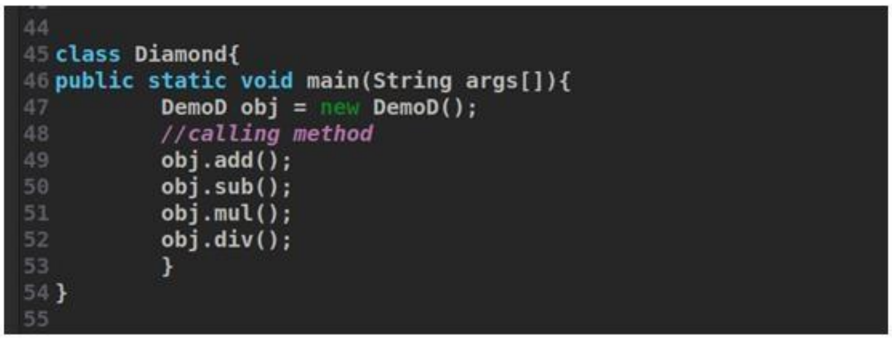
6. Write a program to demonstrate a simple calculator using the diamond problem in Java.



```java
//Simple Calculator using diamond problem

interface DemoA  {
        void add() ;
}

//interface without default method
interface DemoB extends DemoA {
      void sub() ;
}

//interface without default method
interface DemoC extends DemoA   {
      void mul() ;
}
```



```java
//implementation class code
class DemoD implements DemoB, DemoC {
        public void add()   {
        int a=15,b=24,c;
        c=a+b;
        System.out.println("Add Method Answer: " +c);
        }

        public void sub()   {
        int a=105,b=15,c;
        c=a-b;
        System.out.println("Sub method Answer: " +c);
        }

        public void mul()   {
        int a=24,b=42,c;
        c=a*b;
        System.out.println("Mul method Answer: " +c);
        }

        public void div()   {
        int a=150,b=15,c;
        c=a/b;
        System.out.println("Div method Answer: " +c);
        }
}
```



```java
class Diamond{
public static void main(String args[]){
        DemoD obj = new DemoD();
        //calling method
        obj.add();
        obj.sub();
        obj.mul();
        obj.div();
        }
}
```

1. Addition

   Choice: `+`

   Inputs: `10`, `20`

   Output: `Sum = 30`

2. Subtraction

   Choice: `-`

   Inputs: `30`, `10`

   Output: `Sub = 20`

3. Multiplication

   Choice: `*`

   Inputs: `5`, `6`

   Output: `Mul = 30`

4. Division

   Choice: `/`

   Inputs: `20`, `5`

   Output: `Div = 4`

5. Division by Zero

   Choice: `/`

   Inputs: `10`, `0`

   Output: `Error: Division by zero is not allowed.`

PART B

1.Write a Java program to read ‘n’ number of integers into an array.

Raise an appropriate exception (ArithmeticException,

NumberFormatException, ArrayOutOfBoundsException) while Performing

the following operations:

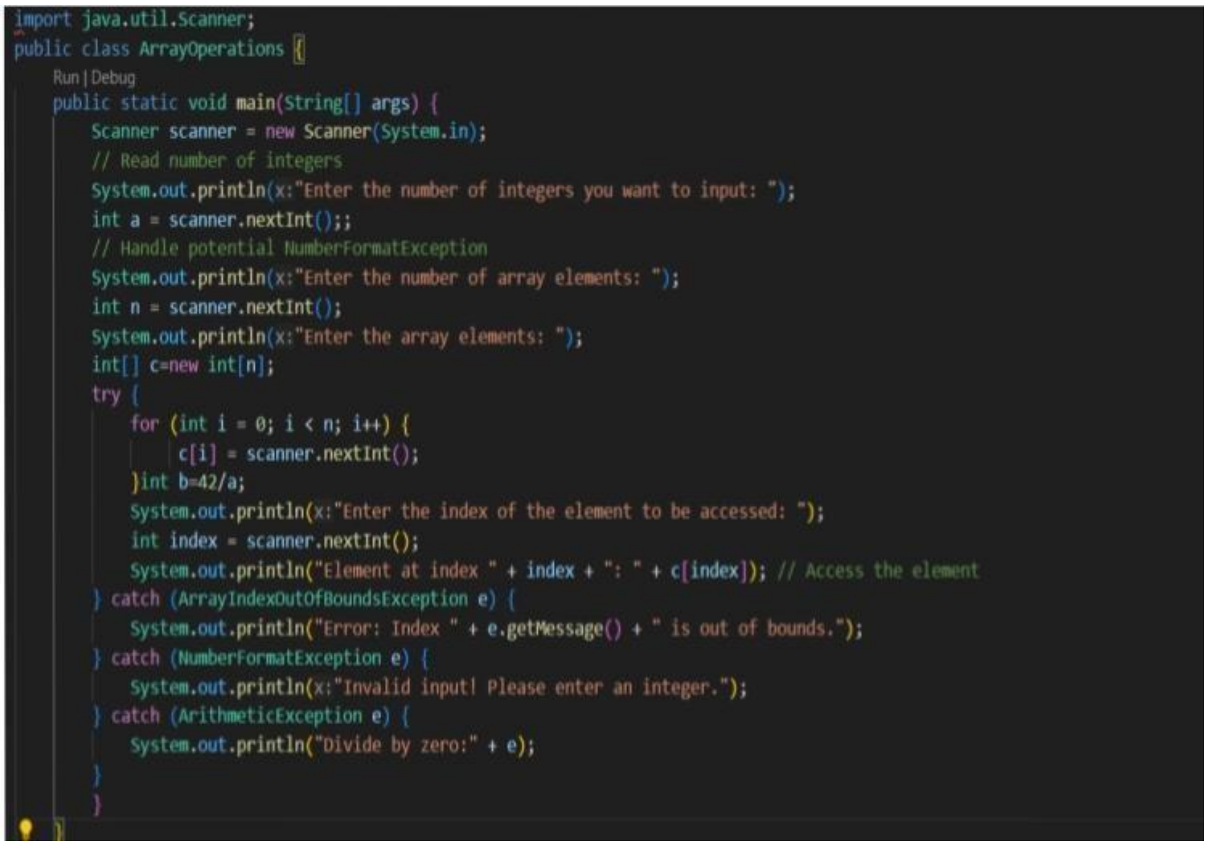
a)Dividing each element by the smallest element in an array.

b)Reading elements from the keyboard

c)Accessing the element from the index specified by the keyboard entry.

Concept of exception handling using multiple catch blocks to be used in

This scenario.



```java
import java.util.Scanner;
public class ArrayOperations {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Read number of integers
        System.out.println(x:"Enter the number of integers you want to input: ");
        int a = scanner.nextInt();;
        // Handle potential NumberFormatException
        System.out.println(x:"Enter the number of array elements: ");
        int n = scanner.nextInt();
        System.out.println(x:"Enter the array elements: ");
        int[] c=new int[n];
        try {
            for (int i = 0; i < n; i++) {
                c[i] = scanner.nextInt();
            }int b=42/a;
            System.out.println(x:"Enter the index of the element to be accessed: ");
            int index = scanner.nextInt();
            System.out.println("Element at index " + index + ": " + c[index]); // Access the element
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Index " + e.getMessage() + " is out of bounds.");
        } catch (NumberFormatException e) {
            System.out.println(x:"Invalid input! Please enter an integer.");
        } catch (ArithmeticException e) {
            System.out.println("Divide by zero:" + e);
        }
        }
}
```

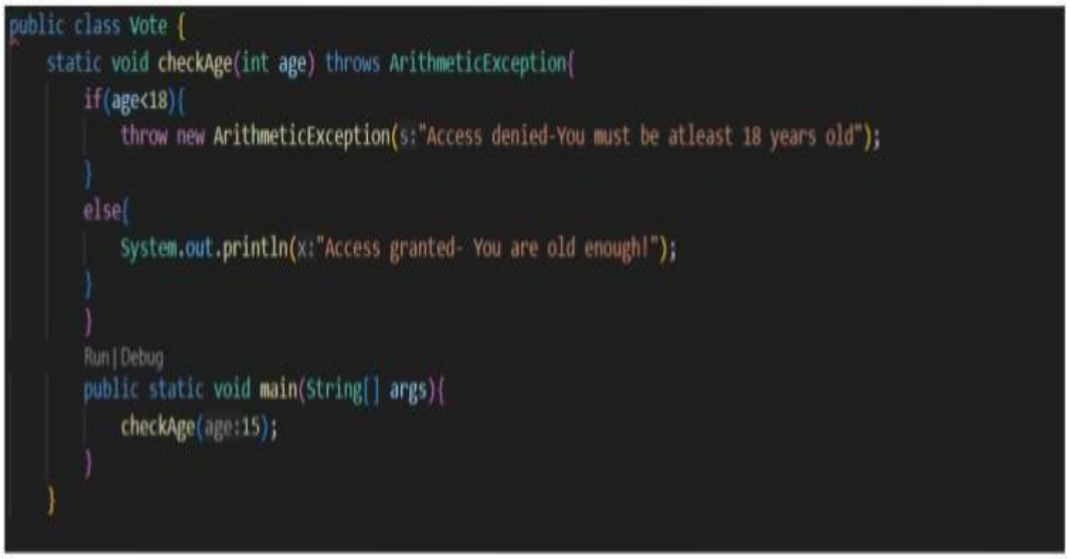
Enter the number of integers: 4

Enter the integers:

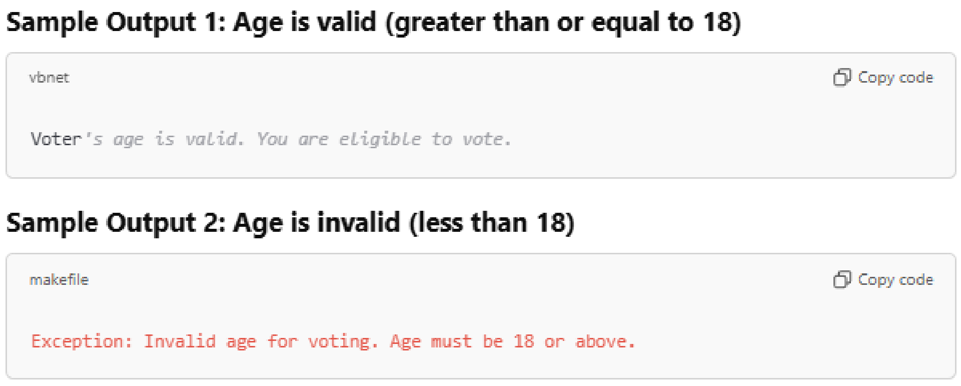
```
10
5
20
15
Smallest element in the array is 5.
Dividing each element by the smallest element:
10 / 5 = 2
5 / 5 = 1
20 / 5 = 4
15 / 5 = 3

Enter the number of integers: 3
Enter the integers:
10
0
20
Smallest element in the array is 0.
Exception: ArithmeticException: Cannot divide by zero

Enter the number of integers: 3
Enter the integers:
10
abc
20
Exception: NumberFormatException: For input string: "abc"

Enter the number of integers: 3
Enter the integers:
10
20
30
Enter the index to access: 5
Exception: ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3
```

2.Write a Java program to demonstrate the use of the throw and throws

Keyword. (Make use of ArithmeticException to verify a voter’s age.)



```
public class Vote {
    static void checkAge(int age) throws ArithmeticException{
        if(age<18){
            throw new ArithmeticException(s:"Access denied-You must be atleast 18 years old");
        }
        else{
            System.out.println(x:"Access granted- You are old enough!");
        }
    }
    Run|Debug
    public static void main(String[] args){
        checkAge(age:15);
    }
}
```



Sample Output 1: Age is valid (greater than or equal to 18)

vbnet   Copy code

```
Voter's age is valid. You are eligible to vote.
```

Sample Output 2: Age is invalid (less than 18)

makefile   Copy code

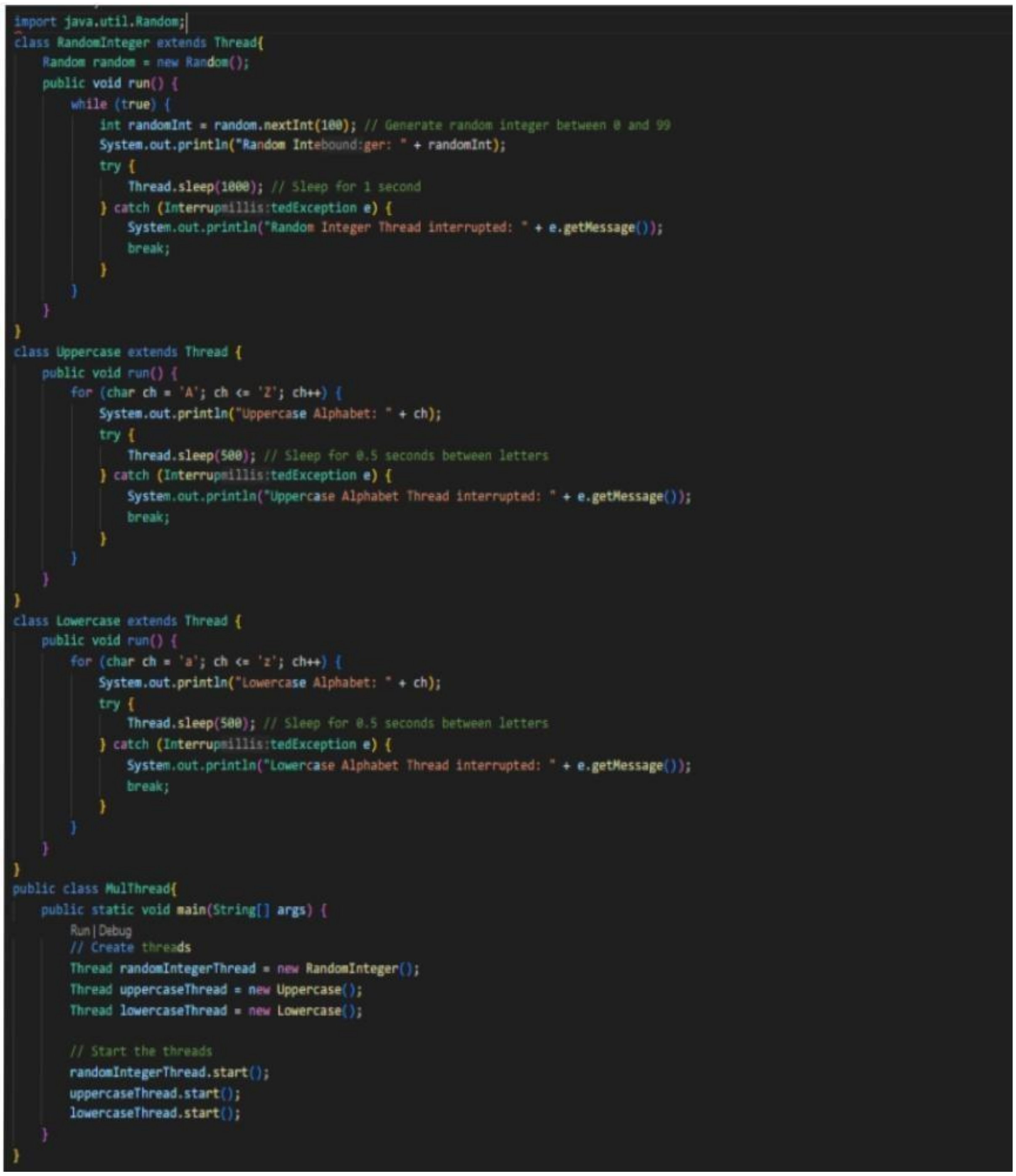
```
Exception: Invalid age for voting. Age must be 18 or above.
```

3.Design a Java program that implements a multithreaded application

That has three threads. The first thread generates a random integer for Every

1 second; the second thread prints uppercase alphabets from A to

Z; the third thread prints lowercase alphabets from a to z.



```java
import java.util.Random;
class RandomInteger extends Thread{
    Random random = new Random();
    public void run() {
        while (true) {
            int randomInt = random.nextInt(100); // Generate random integer between 0 and 99
            System.out.println("Random Intebound:ger: " + randomInt);
            try {
                Thread.sleep(1000); // Sleep for 1 second
            } catch (Interrupmillis:tedException e) {
                System.out.println("Random Integer Thread interrupted: " + e.getMessage());
                break;
            }
        }
    }
}
class Uppercase extends Thread {
    public void run() {
        for (char ch = 'A'; ch <= 'Z'; ch++) {
            System.out.println("Uppercase Alphabet: " + ch);
            try {
                Thread.sleep(500); // Sleep for 0.5 seconds between letters
            } catch (Interrupmillis:tedException e) {
                System.out.println("Uppercase Alphabet Thread interrupted: " + e.getMessage());
                break;
            }
        }
    }
}
class Lowercase extends Thread {
    public void run() {
        for (char ch = 'a'; ch <= 'z'; ch++) {
            System.out.println("Lowercase Alphabet: " + ch);
            try {
                Thread.sleep(500); // Sleep for 0.5 seconds between letters
            } catch (Interrupmillis:tedException e) {
                System.out.println("Lowercase Alphabet Thread interrupted: " + e.getMessage());
                break;
            }
        }
    }
}
public class MulThread{
    public static void main(String[] args) {
        Run | Debug
        // Create threads
        Thread randomIntegerThread = new RandomInteger();
        Thread uppercaseThread = new Uppercase();
        Thread lowercaseThread = new Lowercase();

        // Start the threads
        randomIntegerThread.start();
        uppercaseThread.start();
        lowercaseThread.start();
    }
}
```

Output:

```
Random Number: 42

A

a

Random Number: 56

B
```

b

Random Number: 91

C

c

Random Number: 28

D

d

Random Number: 63

E

e

Random Number: 12

F

f

Random Number: 77

G

g

Random Number: 35

H

h

Random Number: 84

I

i

Random Number: 19

J

j

Random Number: 49

K

k

Random Number: 8

L

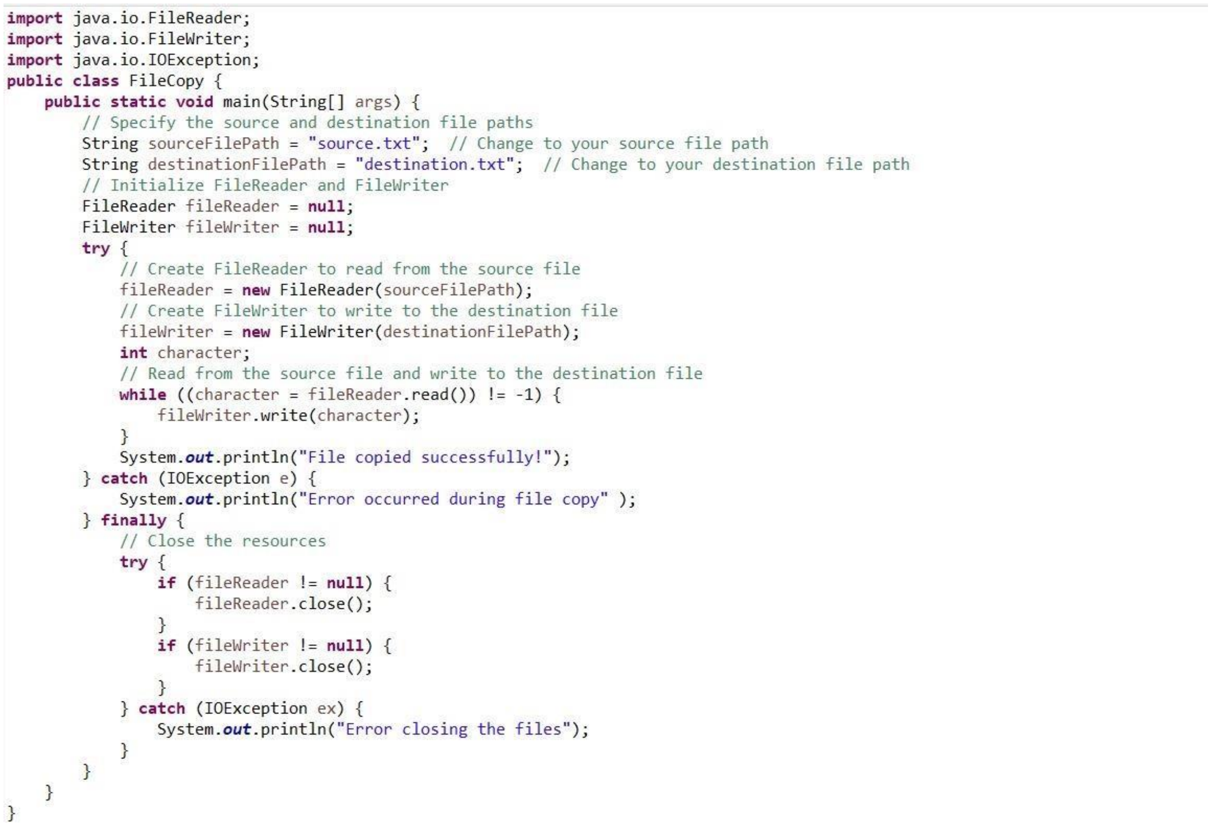
l

```
Random Number: 92

M

m

Random Number: 21

N

n

Random Number: 60

O

o

Random Number: 5

P

p

Random Number: 70

Q

q

Random Number: 39

R

r

Random Number: 53

S

s

Random Number: 87

T

t

Random Number: 14

U

u

Random Number: 46

V

v

Random Number: 99
```

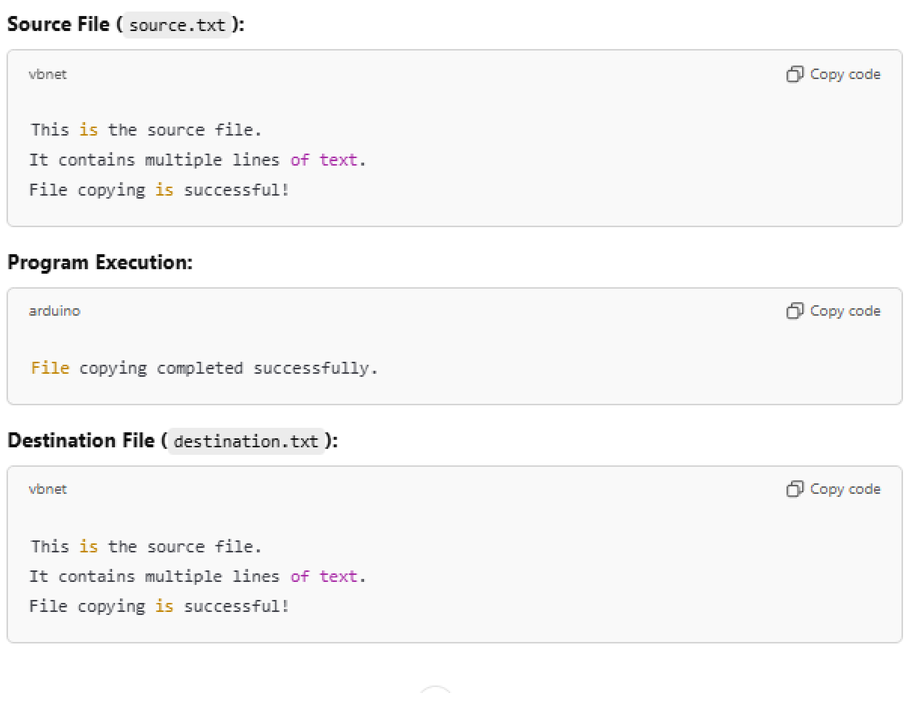
```
W
w
Random Number: 66
X
x
Random Number: 18
Y
y
Random Number: 82
Z
z
```

4.Write a Java program using FileReader and the FileWriter class to

Copy text file content from one file to another.



```java
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class FileCopy {
    public static void main(String[] args) {
        // Specify the source and destination file paths
        String sourceFilePath = "source.txt";  // Change to your source file path
        String destinationFilePath = "destination.txt";  // Change to your destination file path
        // Initialize FileReader and FileWriter
        FileReader fileReader = null;
        FileWriter fileWriter = null;
        try {
            // Create FileReader to read from the source file
            fileReader = new FileReader(sourceFilePath);
            // Create FileWriter to write to the destination file
            fileWriter = new FileWriter(destinationFilePath);
            int character;
            // Read from the source file and write to the destination file
            while ((character = fileReader.read()) != -1) {
                fileWriter.write(character);
            }
            System.out.println("File copied successfully!");
        } catch (IOException e) {
            System.out.println("Error occurred during file copy" );
        } finally {
            // Close the resources
            try {
                if (fileReader != null) {
                    fileReader.close();
                }
                if (fileWriter != null) {
                    fileWriter.close();
                }
            } catch (IOException ex) {
                System.out.println("Error closing the files");
            }
        }
    }
}
```



Source File (`source.txt`):

```
This is the source file.
It contains multiple lines of text.
File copying is successful!
```

Program Execution:

```
File copying completed successfully.
```

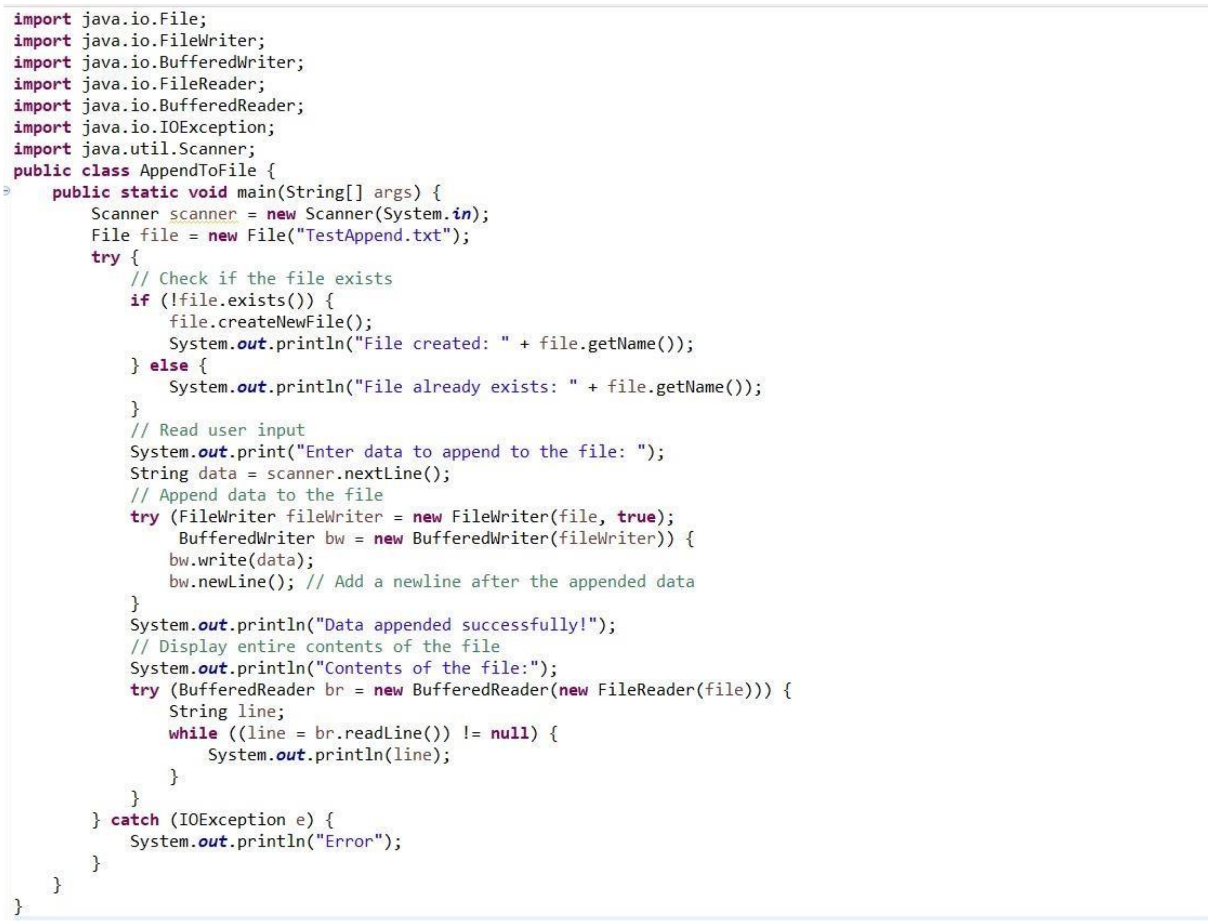
Destination File (`destination.txt`):

```
This is the source file.
It contains multiple lines of text.
File copying is successful!
```

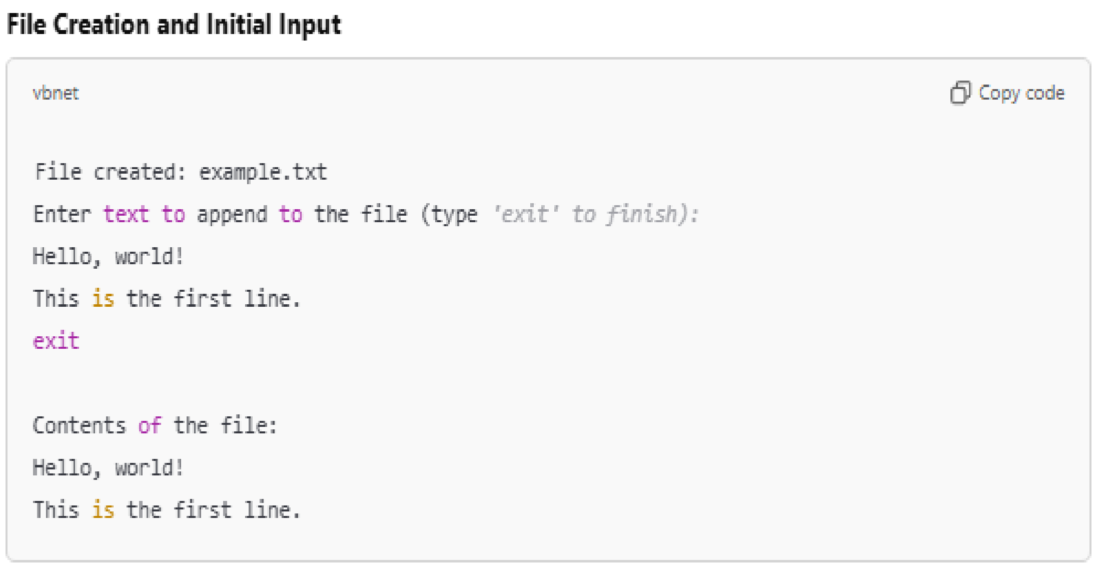
5. Develop a program to create a file if it does not exist and open the file

If it does. Read the contents from the user and append them to the file.

Display the entire contents of the file.



```java
import java.io.File;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.util.Scanner;
public class AppendToFile {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        File file = new File("TestAppend.txt");
        try {
            // Check if the file exists
            if (!file.exists()) {
                file.createNewFile();
                System.out.println("File created: " + file.getName());
            } else {
                System.out.println("File already exists: " + file.getName());
            }
            // Read user input
            System.out.print("Enter data to append to the file: ");
            String data = scanner.nextLine();
            // Append data to the file
            try (FileWriter fileWriter = new FileWriter(file, true);
                 BufferedWriter bw = new BufferedWriter(fileWriter)) {
                bw.write(data);
                bw.newLine(); // Add a newline after the appended data
            }
            System.out.println("Data appended successfully!");
            // Display entire contents of the file
            System.out.println("Contents of the file:");
            try (BufferedReader br = new BufferedReader(new FileReader(file))) {
                String line;
                while ((line = br.readLine()) != null) {
                    System.out.println(line);
                }
            }
        } catch (IOException e) {
            System.out.println("Error");
        }
    }
}
```

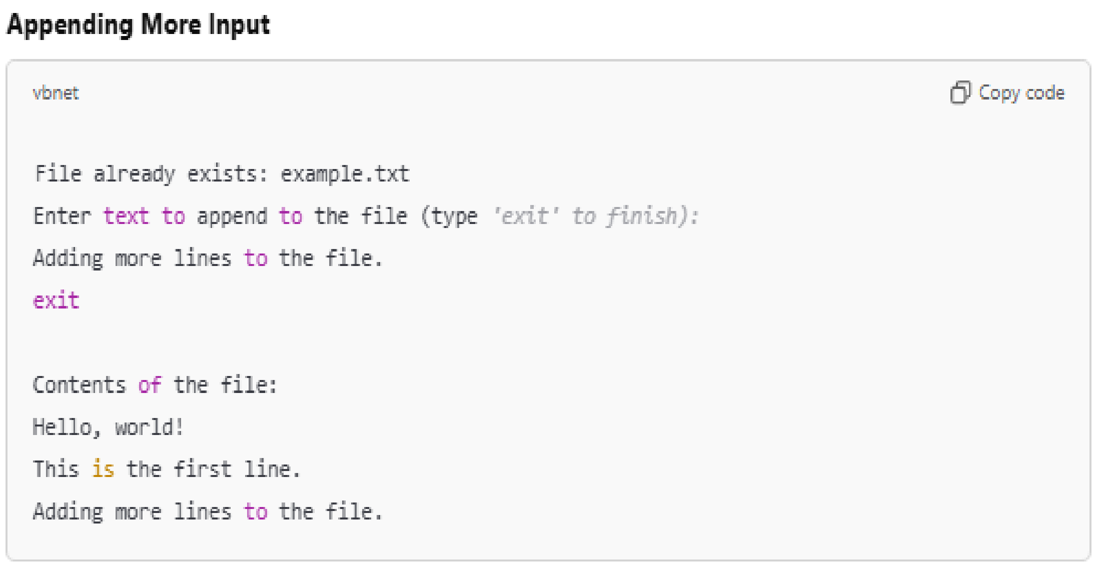


File Creation and Initial Input

vbnet

Copy code

```
File created: example.txt
Enter text to append to the file (type 'exit' to finish):
Hello, world!
This is the first line.
exit

Contents of the file:
Hello, world!
This is the first line.
```



Appending More Input

vbnet

Copy code

```
File already exists: example.txt
Enter text to append to the file (type 'exit' to finish):
Adding more lines to the file.
exit

Contents of the file:
Hello, world!
This is the first line.
Adding more lines to the file.
```

6.Develop a multithreaded Java program to demonstrate

Synchronization of a method. The method is going to print the message

Passed by three child thread instances by embedding square braces

Onto it. E.g [Message].

The messages are:

“Mindset is everything! ”

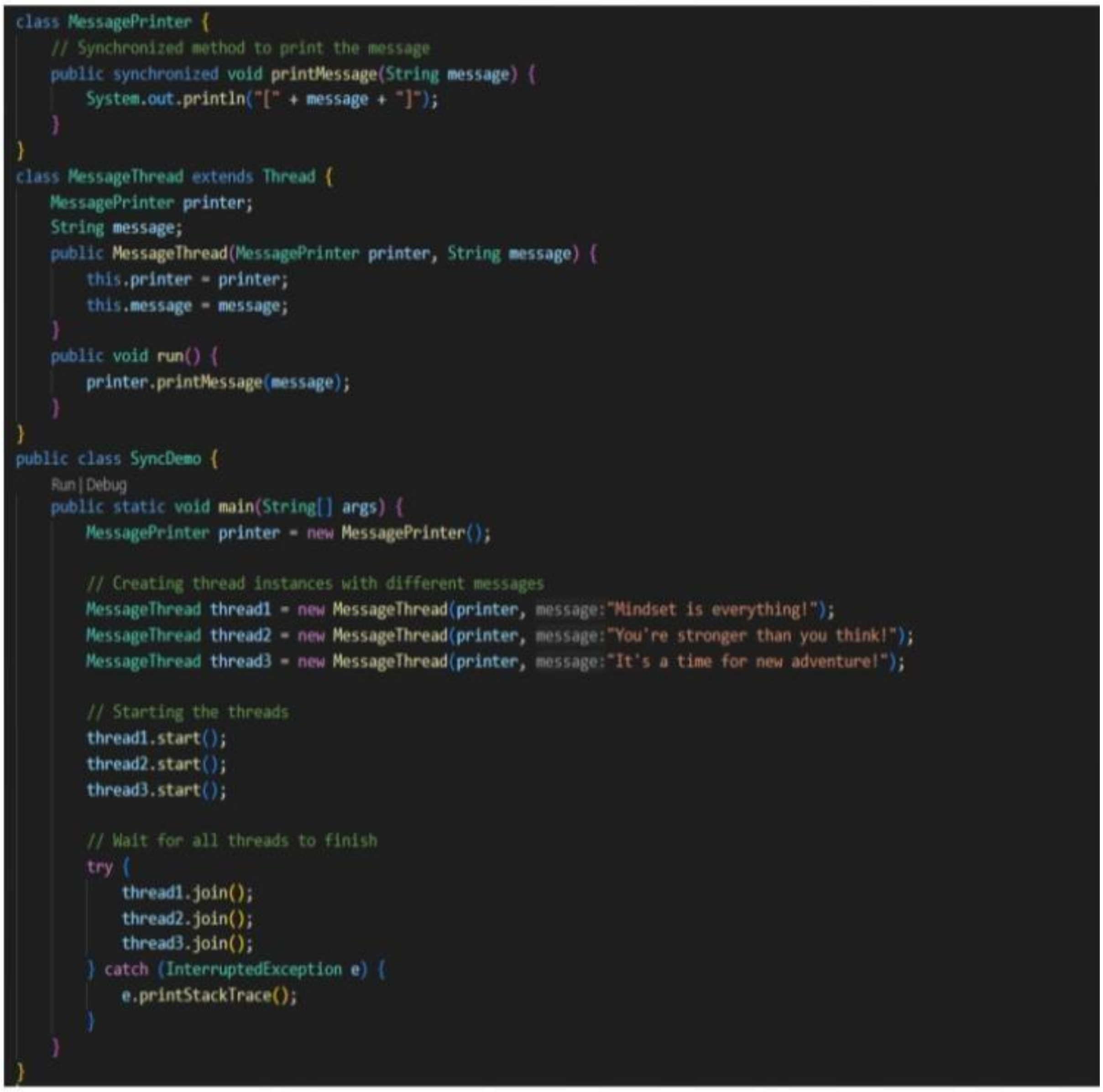
“You’re stronger than you think! ”

“It’s a time for new adventure!”

These are passed to the method by three instances of child threads, Respectively.

Show the possible outputs with and without synchronization

Of the method.



```java
class MessagePrinter {
    // Synchronized method to print the message
    public synchronized void printMessage(String message) {
        System.out.println("[" + message + "]");
    }
}
class MessageThread extends Thread {
    MessagePrinter printer;
    String message;
    public MessageThread(MessagePrinter printer, String message) {
        this.printer = printer;
        this.message = message;
    }
    public void run() {
        printer.printMessage(message);
    }
}
public class SyncDemo {
    Run | Debug
    public static void main(String[] args) {
        MessagePrinter printer = new MessagePrinter();

        // Creating thread instances with different messages
        MessageThread thread1 = new MessageThread(printer, message:"Mindset is everything!");
        MessageThread thread2 = new MessageThread(printer, message:"You're stronger than you think!");
        MessageThread thread3 = new MessageThread(printer, message:"It's a time for new adventure!");

        // Starting the threads
        thread1.start();
        thread2.start();
        thread3.start();

        // Wait for all threads to finish
        try {
            thread1.join();
            thread2.join();
            thread3.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

With Synchronization

Using a synchronized method ensures that only one thread can execute the method at a time. The outputs will always appear in full, without any interleaving:

```python
[Mindset is everything!]
[You're stronger than you think!]
[It's a time for new adventure!]
```

Without Synchronization

Without synchronization, threads may interleave during execution, causing mixed outputs:

```python
[Mindset is
everything!]
[You're stronger than
you think!]
[It's
a time for new adventure!]
```