1. Write a program to show method overloading in Java.

**Theory:**

**Method overloading:**

Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters or both. Overloading is related to compile-time (or static) polymorphism.

**Source Code:**

public class Sum {

    public int sum(int x, int y)

    {

        return (x + y);

    }

    public int sum(int x, int y, int z)

    {

        return (x + y + z);

    }

    public double sum(double x, double y)

    {

        return (x + y);

    }

    public static void main(String args[])

    {

        Sum s = new Sum();

        System.out.println(s.sum(20, 45));

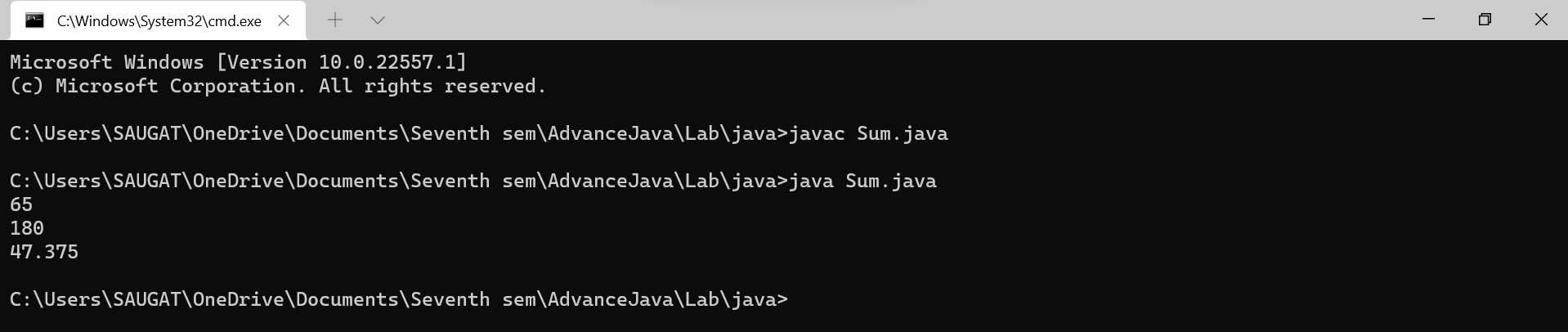
        System.out.println(s.sum(50, 60, 70));

        System.out.println(s.sum(25.25, 22.125));

    }

}

**Output:**



2. Write a program to show method overloading in Java.

**Theory:**

**Method overriding:**

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java. In other words, if a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

**Source Code:**

class Vehicle{

  void run(){

      System.out.println("Vehicle is running");

      }

}

class Bike extends Vehicle{

    public static void main(String args[]){

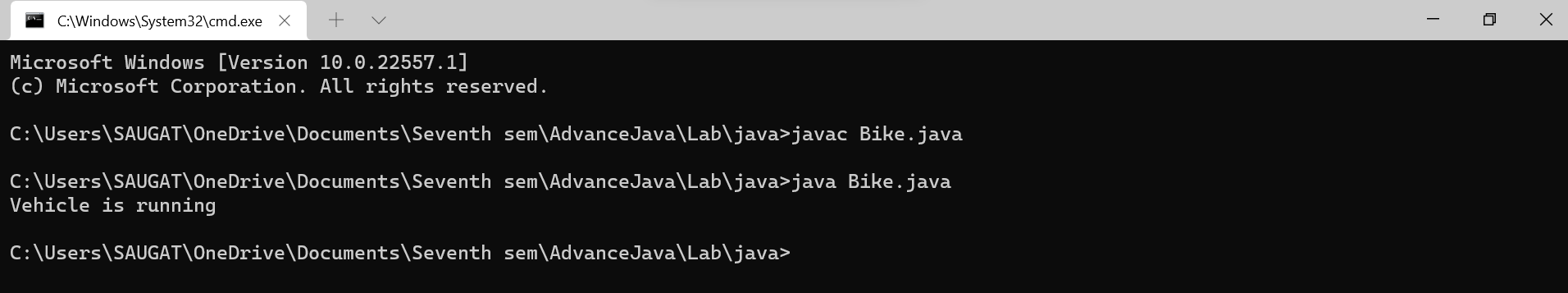
    Bike obj = new Bike();

    obj.run();

    }

  }

**Output:**



3. Write a program to show Inheritance in java.

**Theory:**

**Inheritance:**

**Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

**Source Code:**

public class InheritanceExample {

    public static void main(String[] args) {

        Dog d = new Dog();

        d.eat();

        d.bark();

        Cat c = new Cat();

        c.eat();

        c.meow();

    }

    }

class Animal {

    public void eat() {

        System.out.println("Animal is eating");

    }

 }

 class Dog extends Animal {

    public void bark() {

        System.out.println("Dog is barking");

    }

 }

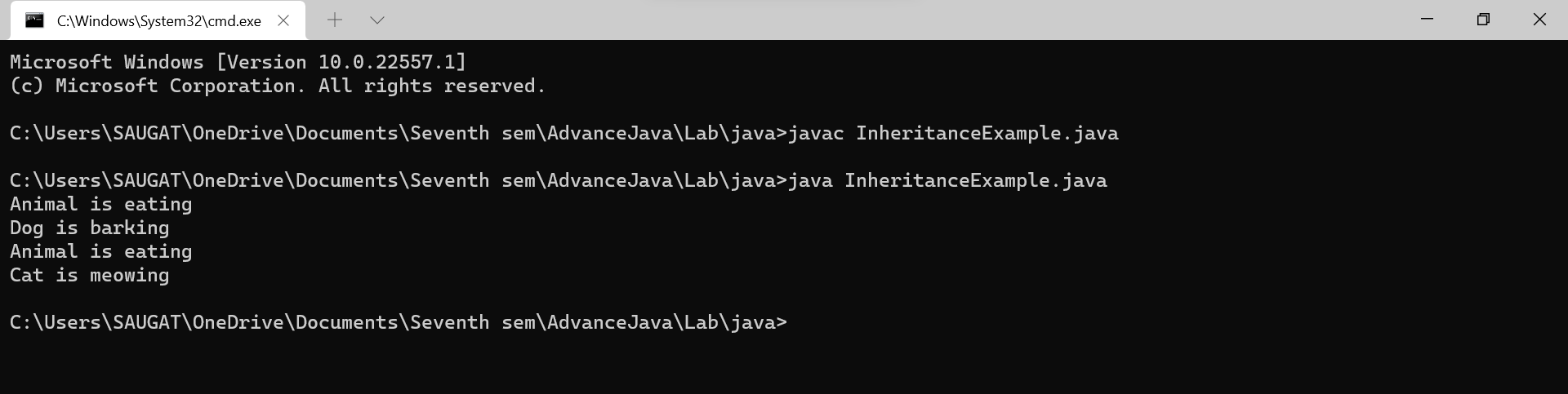
 class Cat extends Animal {

    public void meow() {

        System.out.println("Cat is meowing");

    }

 }

**Output: **

4. Write a program to Handle DivdeByZeroException and IndexOutOfRangeException.

**Theory:**

**Exception Handling:**

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc. The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

**Source Code:**

public class ExceptionDemo {

   public static void main (String args[]) {

      int array[] = {20,20,40};

      int num1 = 15, num2 = 0;

      int result = 0;

      try {

         result = num1/num2;

         System.out.println("The result is" +result);

         for(int i = 2; i >= 0; i--) {

            System.out.println("The value of array is" +array[i]);

         }

      } catch (ArrayIndexOutOfBoundsException e) {

         System.out.println("Error. Array is out of Bounds"+e.toString());

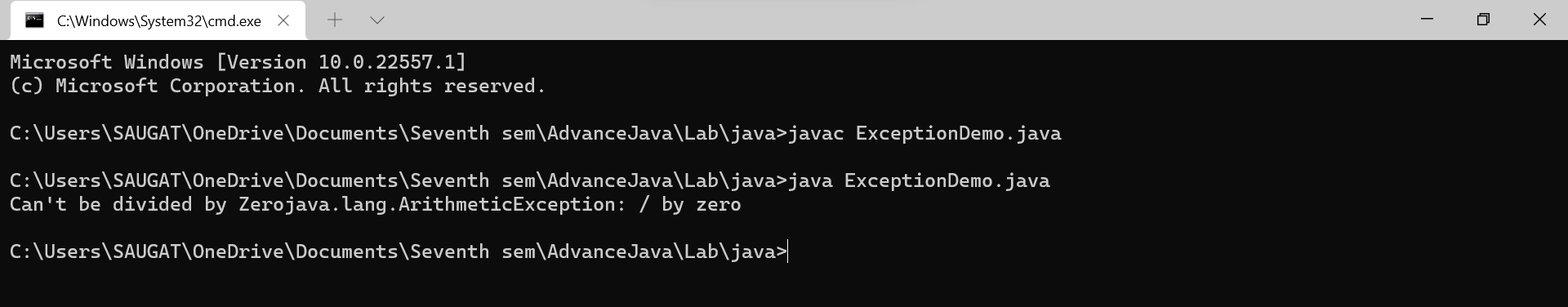
      } catch (ArithmeticException e) {

         System.out.println ("Can't be divided by Zero"+e.toString());

      }

   }

}

**Output:**

5. Write a program to demonstrate the synchronization in java.

**Theory:**

**Synchronization:**

Synchronization in Java is the capability to control the access of multiple threads to any shared resource. Java Synchronization is better option where we want to allow only one thread to access the shared resource. The synchronization is mainly used to:

1. To prevent thread interference.
2. To prevent consistency problem.

**Source Code:**

class Table{

 synchronized void printTable(int n){

   for(int i=1;i<=5;i++){

     System.out.println(n\*i);

     try{

      Thread.sleep(400);

     }catch(Exception e){System.out.println(e);}

   }

 }

}

public class TestSynchronization{

    public static void main(String args[]){

    final Table obj = new Table();

    Thread t1=new Thread(){

    public void run(){

    obj.printTable(5);

    }

    };

    Thread t2=new Thread(){

    public void run(){

    obj.printTable(100);

    }

    };

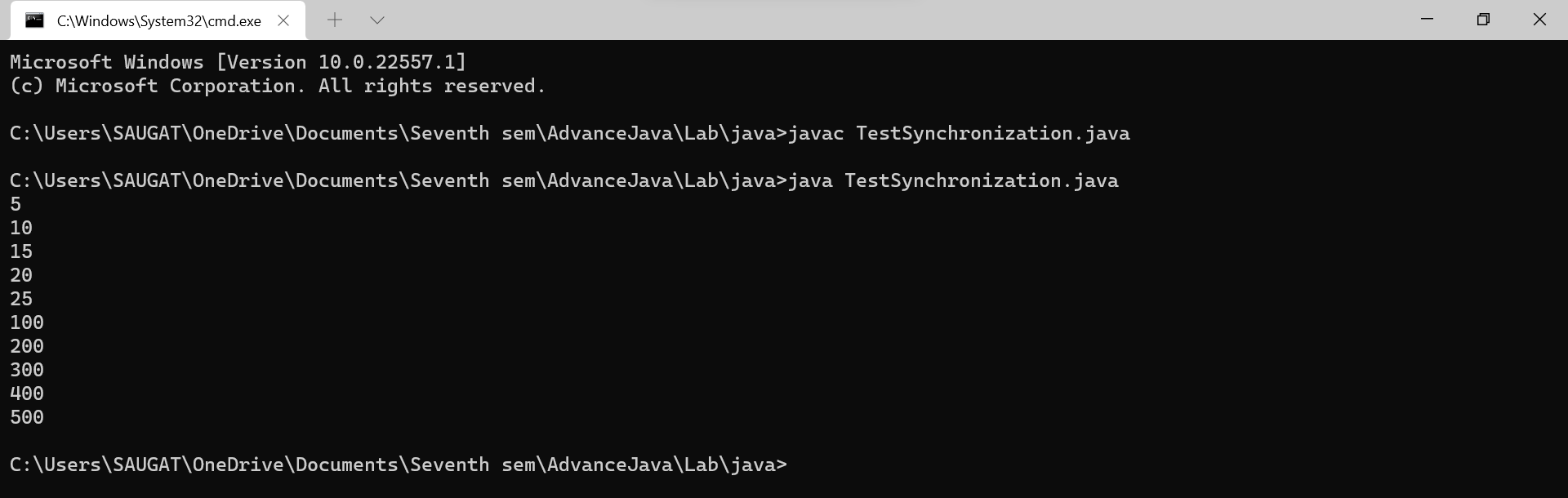
    t1.start();

    t2.start();

    }

    }

**Output:**

****

6. WAP to make the use of FileInputStream and FileOutputStream classes to copy an input file into an output file.

**Theory:**

In Java, streams are the sequence of data that are read from the source and written to the destination. An input stream is used to read data from the source. And, an output stream is used to write data to the destination.

**Source Code:**

import java.io.FileInputStream;

import java.io.FileOutputStream;

public class IOStream {

    public static void main(String[] args) {

        FileInputStream in= null;

        FileOutputStream out= null;

        try{

            in= new FileInputStream("input.txt");

            out= new FileOutputStream("output.txt");

            int c;

            while((c=in.read())!=-1){

                out.write(c);

            }

        }

        catch(Exception e){

            System.out.println(e);

        }

        finally{

            try{

                in.close();

                out.close();

            }

            catch(Exception e){

                System.out.println(e);

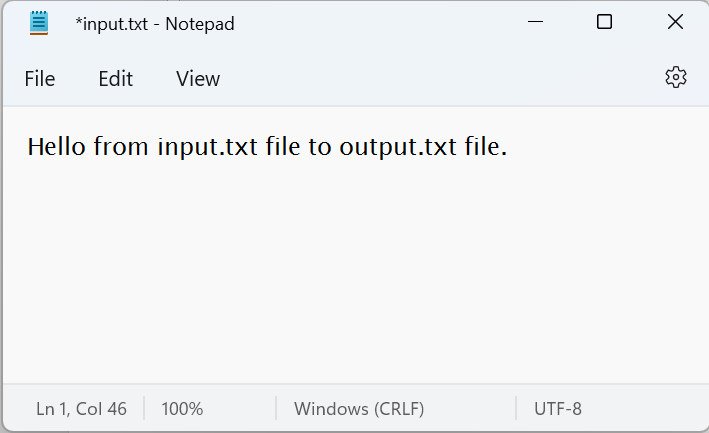
            }

        }

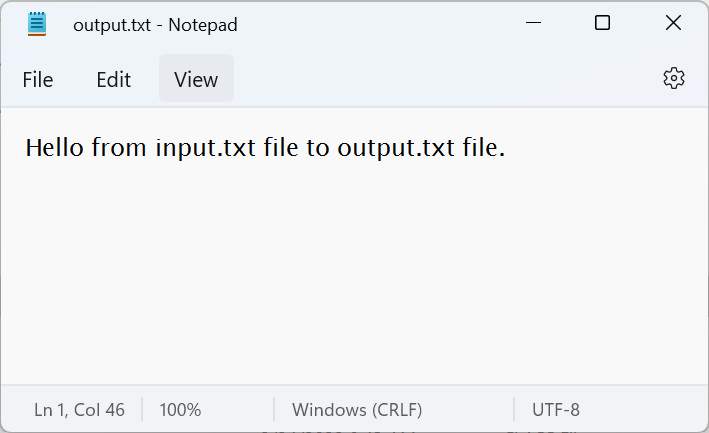
    }

}

**Input file:**



**Output file:**

****

**Conclusion:**

In the first lab of Advanced Java Programming, we successfully implemented the concept of method overloading, overriding, inheritance, exception handling, synchronization problem and file handling program using java programming language.