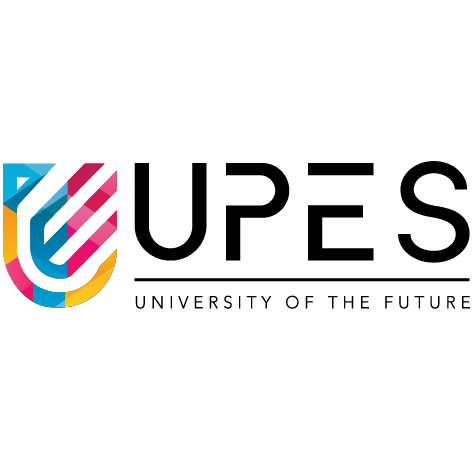
**School of Computer Science**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

**DEHRADUN, UTTARAKHAND**



**Submitted To: Sachi Chaudhry Name: Deepanshu Rawat**

**SAP ID: 500097692 Course: B. Tech CSE**

**Roll No: R2142211410 Program: DevOps (Honours)**

**Semester: III Session: 2022-23**

OBJECT ORIENTED PROGRAMMING LAB FILE

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No.** | **Experiment No.** | **Title** | **Date of Performance** | **Date of Submission** | **Remarks (By Faculty)** |
| **1.** | **Experiment 1** | **Basic Java Packages and Definitions** | **17 Aug 2022** | **19 October 2022** |  |
| **2.** | **Experiment 2** | **Basic Java programming** | **17 Aug 2022** | **19 October 2022** |  |
| **3.** | **Experiment 3** | **Classes, and Objects in Java** | **24 Aug 2022** | **19 October 2022** |  |
| **4.** | **Experiment**  **4** | **Inheritance** | **7 Sept 2022** | **19 October 2022** |  |
| **5.** | **Experiment**  **5** | **Interface** | **14 Sept 2022** | **19 October 2022** |  |
| **6.** | **Experiment**  **6** | **Package** | **21 Sept 2022** | **19 October 2022** |  |
| **7.** | **Experiment**  **7** | **Exceptions** | **28 Sept 2022** |  |  |
| **8.** | **Experiment**  **8** | **String Handling and Wrapper Classes** | **19 Oct 2022** |  |  |
| **9.** | **Experiment**  **9** | **Threads and Collections** | **3 Nov 2022** |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Experiment 1:

**Date of performance: 17 August 2022**

**Date of Submission: 19 October 2022**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

Title: Introduction to Java Environment

Definitions:

JDK: JDK stands for Java Development Kit. It is a software development environment used to develop Java applications and applets. It is a platform-specific software i.e., there are separate installers for Windows, Mac, and Unix systems. Java developers can use it on their Windows, macOS, Solaris, and Linux to code and run Java programs. It contains Java Runtime Environment (JRE)and other development tools like an interpreter, compiler, archiver, and a document generator. We can possibly install more than one JDK version on the same computer.

JRE: Java Runtime Environment (JRE) is an open-access software distribution that has a Java class library, specific tools, and a separate JVM. JRE is one of the interrelated components in the Java Development Kit (JDK). It is the most common environment available on devices for running Java programs. Java source code is compiled and converted to Java bytecode. If you want to run this bytecode on any platform, you need JRE. The JRE loads classes check memory access and get system resources. JRE acts as a software layer on top of the operating system.

JVM: JVM (Java Virtual Machine) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as interpreter.

javac: The **javac** tool reads class and interface definitions, written in the Java programming language, and compiles them into bytecode class files. It can also process annotations in Java source files and classes.

1. SDK: SDK stands for software development kit. Also known as a devkit, the SDK is a set of software-building tools for a specific platform, including the building blocks, debuggers and, often, a framework or group of code libraries such as a set of routines specific to an operating system (OS).

A typical SDK might include some or all of these resources in its set of tools:

* **Compiler:** Translates from one programming language to the one in which you will work
* **Code samples:** Give a concrete example of an application or web page
* **Code libraries (framework):** Provide a shortcut with code sequences that programmers will use repeatedly
* **Testing and analytics tools:** Provide insight into how the application or product performs in testing and production environments
* **Documentation:** Gives developers instructions they can refer to as they go
* **Debuggers:** Help teams spot errors in their code so they can push out code that works as expected

Experiment: 2

**Date of performance: 17 August 2022**

**Date of Submission: 19 October 2022**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

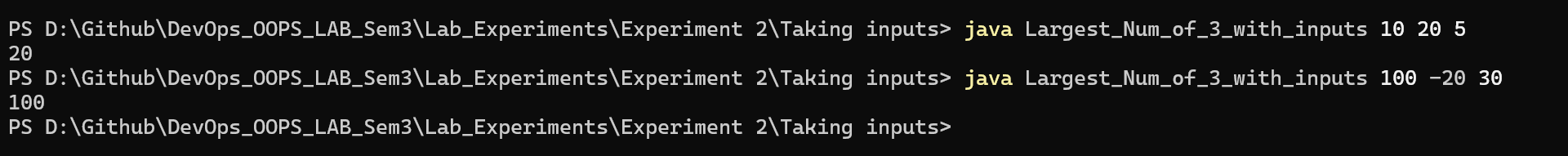
Title: Basic Java Programming

1. Write a program to find the largest of 3 numbers

Code:

*public class* Largest\_Num\_of\_3\_with\_inputs{  
 *int* greatest(*int* a,*int* b,*int* c){  
 *int* max=a;  
 *if*(max<b) max=b;  
 *if*(max<c) max=c;  
 *return* max;  
 }  
 *public static void* main(String[] args){  
 *int* ans;  
 Largest\_Num\_of\_3\_with\_inputs obj=*new* Largest\_Num\_of\_3\_with\_inputs();  
 *int* a=Integer.parseInt(args[0]);  
 *int* b=Integer.parseInt(args[1]);  
 *int* c=Integer.parseInt(args[2]);  
 ans=obj.greatest(a,b,c);  
 System.out.println(ans);  
 }  
}

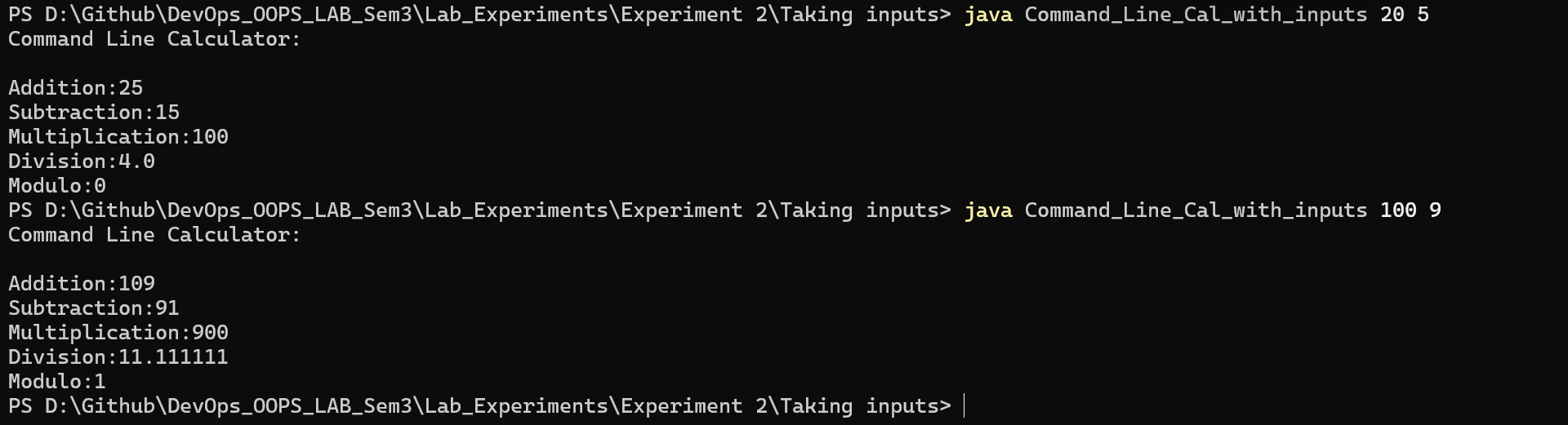
Output:



1. Write a program to implement a command line calculator.

Code:

*public class* Command\_Line\_Cal\_with\_inputs {  
 *int* add(*int* a,*int* b){  
 *return* a+b;  
 }  
 *int* subtract(*int* a,*int* b){  
 *return* a-b;  
 }  
 *int* product(*int* a,*int* b){  
 *return* a\*b;  
 }  
 *float* quotient(*int* a,*int* b){  
 *return* (*float*)a/b;  
 }  
 *int* mod(*int* a,*int* b){  
 *return* a%b;  
 }  
 *public static void* main(String[] args) {  
 Command\_Line\_Cal\_with\_inputs obj=*new* Command\_Line\_Cal\_with\_inputs();  
 *int* a=Integer.parseInt(args[0]);  
 *int* b=Integer.parseInt(args[1]);  
 System.out.println("Command Line Calculator:\n");  
 System.out.println("Addition:"+obj.add(a,b));  
 System.out.println("Subtraction:"+obj.subtract(a,b));  
 System.out.println("Multiplication:"+obj.product(a,b));  
 System.out.println("Division:"+obj.quotient(a,b));  
 System.out.println("Modulo:"+obj.mod(a,b));  
 }  
}

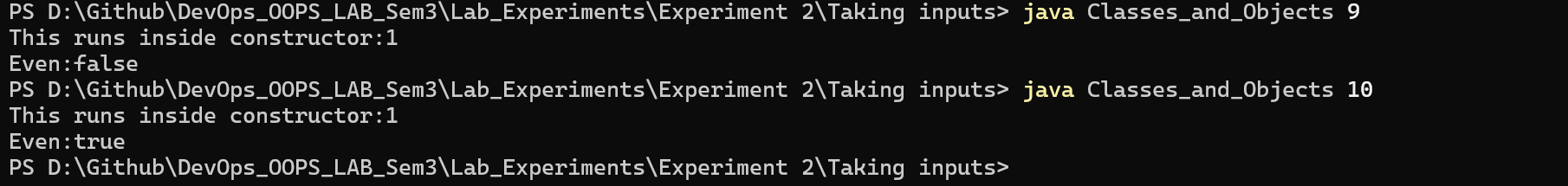
Output:

1. Write a program using classes and object in java.

Code:

*public class* Classes\_and\_Objects {  
 *int* var=0;  
 Classes\_and\_Objects(){  
 var++;  
 System.out.println("This runs inside constructor:"+var);  
 }  
 *boolean* even\_or\_odd(*int* a){  
 *return* (a&1)==0;  
 }  
 *public static void* main(String[] args) {  
 Classes\_and\_Objects obj=*new* Classes\_and\_Objects();  
 *int* a=Integer.parseInt(args[0]);  
 System.out.println("Even:"+obj.even\_or\_odd(a));  
 }  
}

Output:



1. Write a program to accept 10 student’s marks in an array, arrange it into ascending order, convert into the following grades and print marks and grades in the tabular form.

Between 40 and 50: PASS

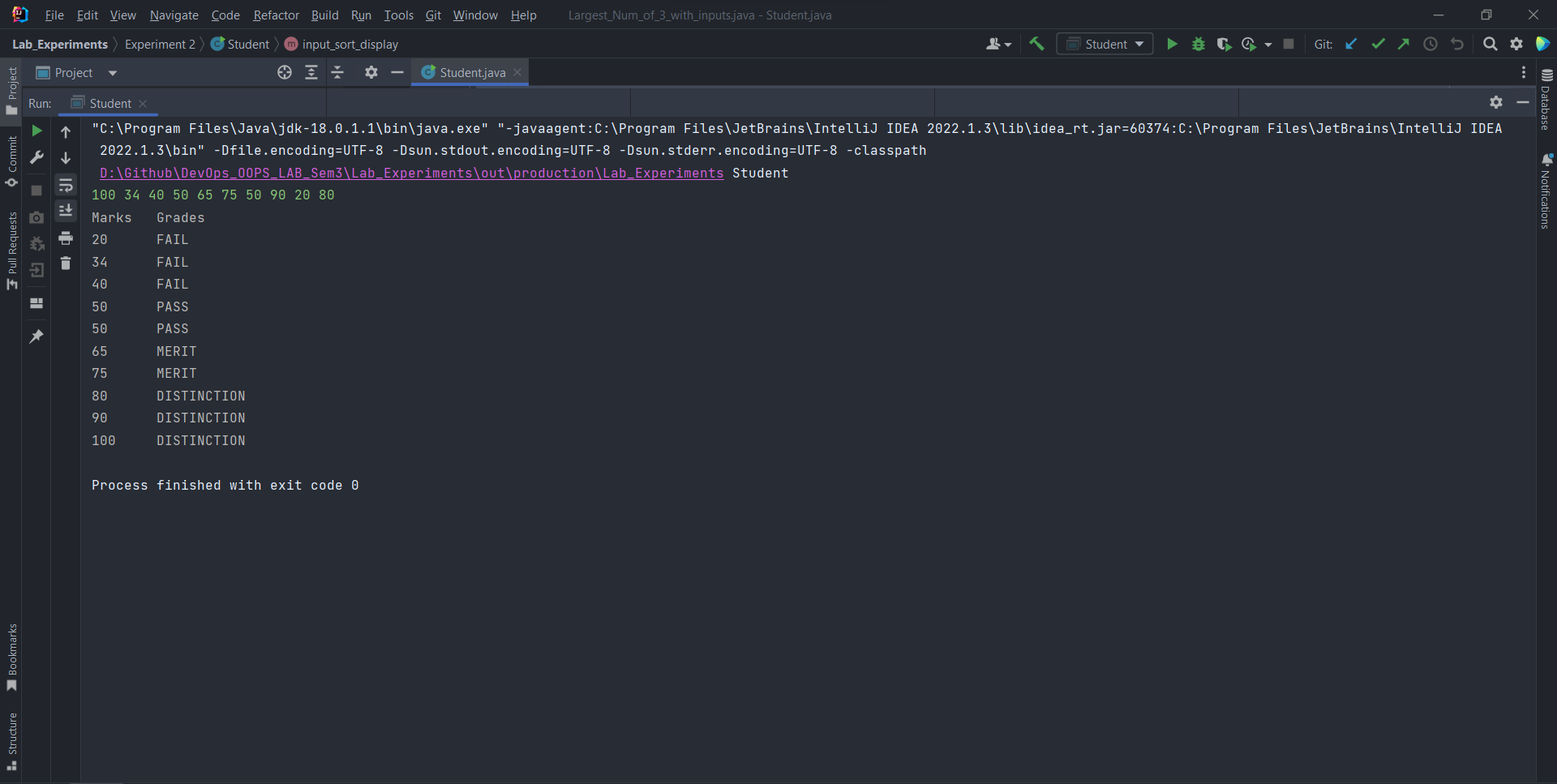
Between 51 and 75: MERIT

and above: DISTINCTION

Code:

*import* java.util.Scanner;  
  
*public class* Student {  
 *void* input\_sort\_display(){  
 Scanner sc=*new* Scanner(System.in);  
 *int*[] arr=*new int*[10];  
 *for*(*int* i=0;i<10;i++){  
 arr[i]= sc.nextInt();  
 }  
*// Used Insertion Sort here  
// Insertion sort starts  
 for*(*int* i=1;i<10;i++){  
 *int* j=i-1;  
 *int* key=arr[i];  
 *while*(j>=0 && arr[j]>key){  
 arr[j+1]=arr[j];  
 j--;  
 }  
 arr[j+1]=key;  
 }  
*// Insertion sort ends here* String[] grades=*new* String[10];  
 *for* (*int* i=0;i<10;i++){  
 *if*(arr[i]<=40){  
 grades[i]="FAIL";  
 }*else if*(arr[i]>40 && arr[i]<=50){  
 grades[i]="PASS";  
 }*else if*(arr[i]>50 && arr[i]<=75){  
 grades[i]="MERIT";  
 }*else*{  
 grades[i]="DISTINCTION";  
 }  
 }  
 System.out.println("Marks\tGrades");  
 *for*(*int* i=0;i<10;i++){  
 System.out.println(arr[i]+"\t\t"+grades[i]);  
 }  
 }  
 *public static void* main(String[] args) {  
 Student obj=*new* Student();  
 obj.input\_sort\_display();  
 }  
}

Output:



Experiment: 3

**Date of performance: 24 August 2022**

**Date of Submission: 19 October 2022**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

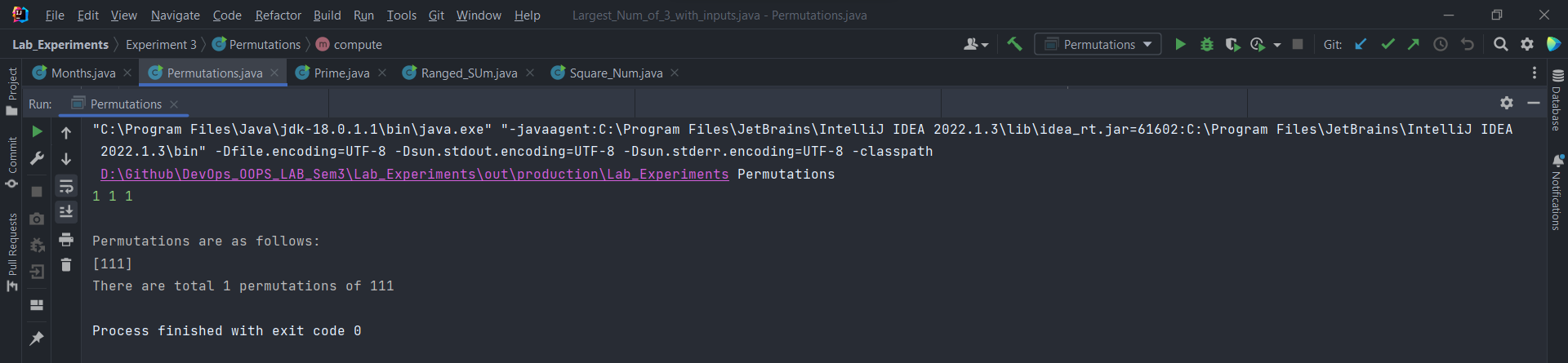
Title: Basic Java Programming

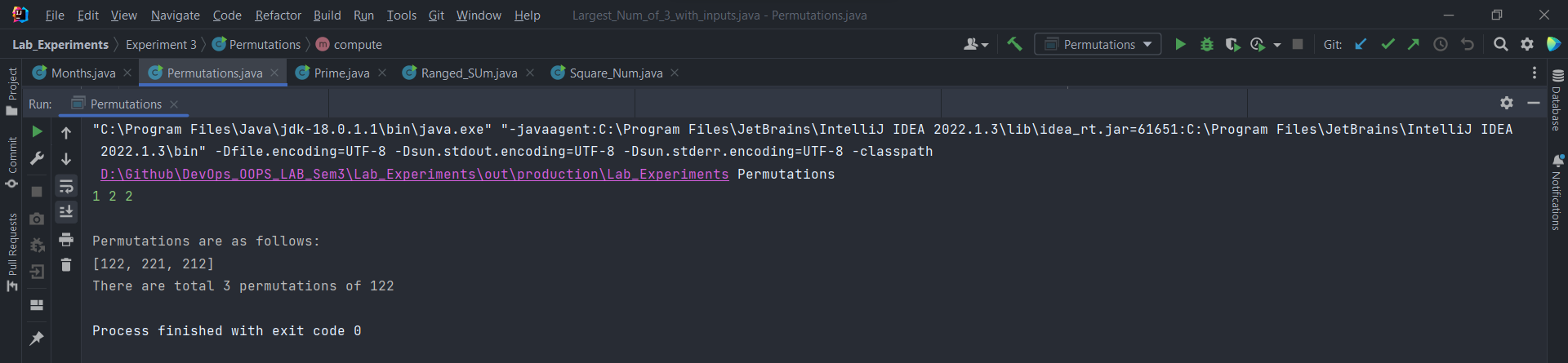
1. Write a program to accept three digits (i.e., 0 - 9) and print all its possible combinations. (For example, if the three digits are 1, 2, 3 than all possible combinations are: 123, 132,213, 231, 312, 321.)

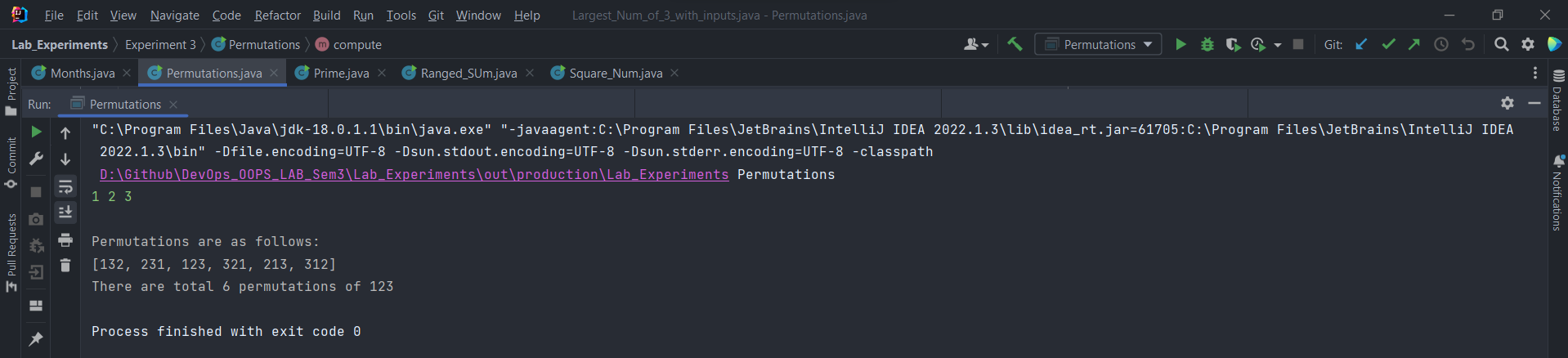
Code:

*import* java.util.HashSet;  
*import* java.util.Scanner;  
  
*public class* Permutations {  
 *void* compute(){  
 HashSet<String>s=*new* HashSet<String>();  
 Scanner sc=*new* Scanner(System.in);  
 *int*[] arr=*new int*[3];  
 arr[0]=sc.nextInt();  
 arr[1]=sc.nextInt();  
 arr[2]=sc.nextInt();  
 *int* c=0;  
 System.out.println("\nPermutations are as follows:");  
 *for*(*int* i=0;i<3;i++){  
 *for*(*int* j=0;j<3;j++){  
 *for*(*int* k=0;k<3;k++){  
 *if*(i!=j && j!=k && i!=k){  
 s.add(arr[i]+""+arr[j]+""+arr[k]);  
 }  
 }  
 }  
 }  
 c=s.size();  
 System.out.println(s);  
 System.out.println("There are total "+c+" permutations of "+arr[0]+""+arr[1]+""+arr[2]);  
 }  
 *public static void* main(String[] args) {  
 Permutations obj=*new* Permutations();  
 obj.compute();  
 }  
}

Output:





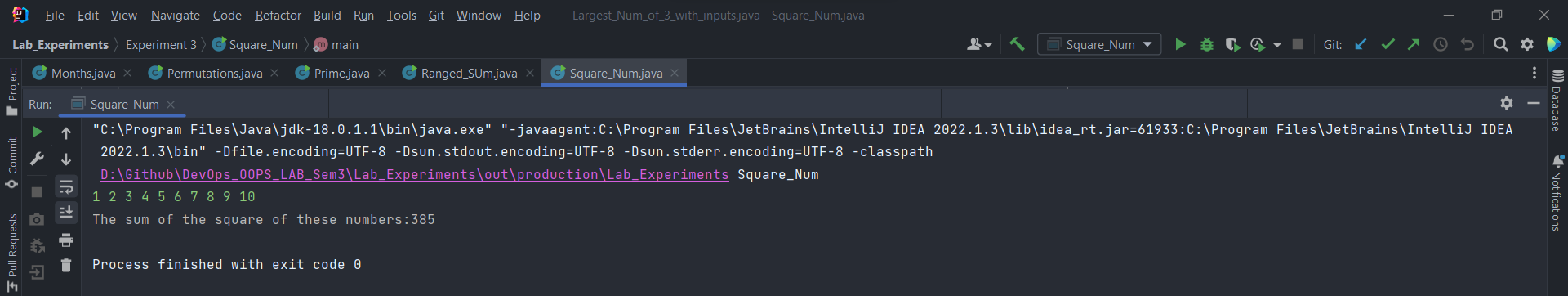


1. Write a Java Program to accept 10 numbers in an array and compute the square of each number. Print the sum of these numbers.

Code:

*import* java.util.Scanner;  
  
*public class* Square\_Num {  
 *void* compute(){  
 *int*[] arr=*new int*[10];  
 *int* sum=0;  
 Scanner sc=*new* Scanner(System.in);  
 *for*(*int* i=0;i<10;i++){  
 arr[i]=sc.nextInt();  
 sum+=arr[i]\*arr[i];  
 }  
 System.out.println("The sum of the square of these numbers:"+sum);  
 }  
 *public static void* main(String[] args) {  
 Square\_Num obj=*new* Square\_Num();  
 obj.compute();  
 }  
}

Output:

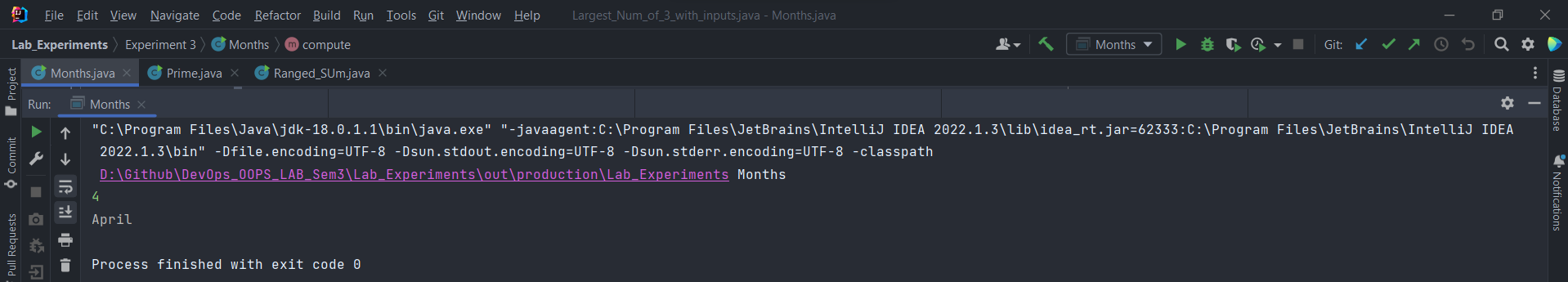


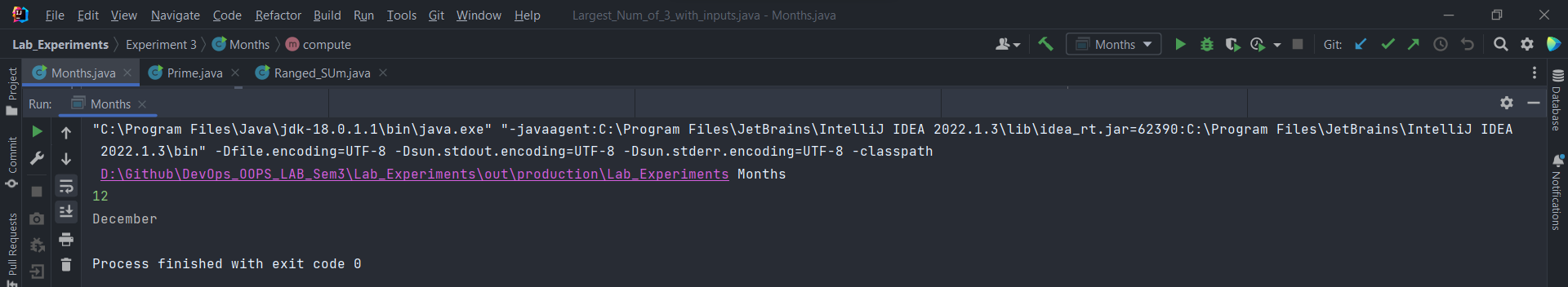
1. Write a program to input a number of a month (1 - 12) and print its equivalent name of the month. (e.g., 1 to Jan, 2 to Feb. 12 to Dec.)

Code:

*import* java.util.Scanner;  
  
*public class* Months {  
 *void* compute(){  
 Scanner sc=*new* Scanner(System.in);  
 *int* n=sc.nextInt();  
 *switch* (n){  
 *case* 1:  
 System.out.println("January");  
 *break*;  
 *case* 2:  
 System.out.println("February");  
 *break*;  
 *case* 3:  
 System.out.println("March");  
 *break*;  
 *case* 4:  
 System.out.println("April");  
 *break*;  
 *case* 5:  
 System.out.println("May");  
 *break*;  
 *case* 6:  
 System.out.println("June");  
 *break*;  
 *case* 7:  
 System.out.println("July");  
 *break*;  
 *case* 8:  
 System.out.println("August");  
 *break*;  
 *case* 9:  
 System.out.println("September");  
 *break*;  
 *case* 10:  
 System.out.println("October");  
 *break*;  
 *case* 11:  
 System.out.println("November");  
 *break*;  
 *case* 12:  
 System.out.println("December");  
 *break*;  
 *default*:  
 System.out.println("Please enter a valid month number");  
 *break*;  
 }  
 }  
 *public static void* main(String[] args) {  
 Months obj=*new* Months();  
 obj.compute();  
 }  
}

Output:



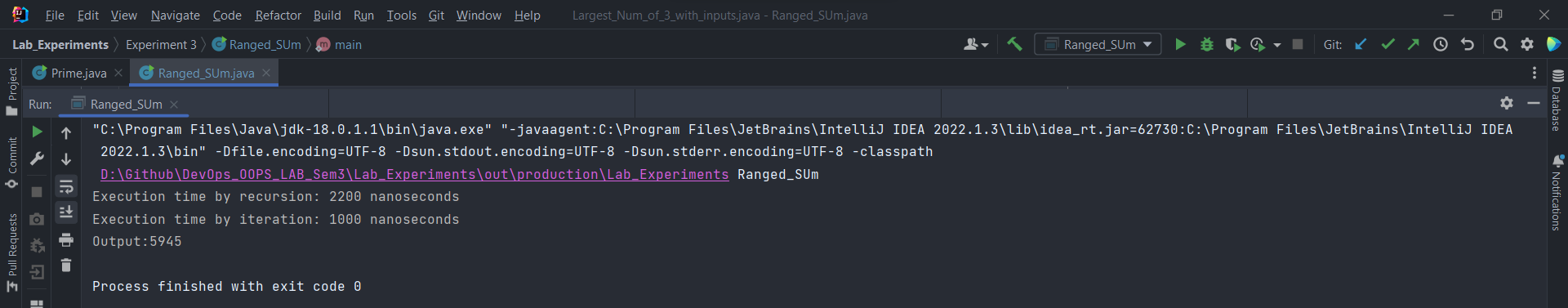


1. Write a program to find the sum of all integers greater than 40 and less than 250 that are divisible by 5.

Code:

*public class* Ranged\_SUm {  
 *int* compute(){  
 *int* i=45;  
 *int* sum=0;  
 *while*(i<250){  
 sum+=i;  
 i+=5;  
 }  
 *return* sum;  
*// System.out.println("The sum of all integers greater than 40 and less than 250 that are divisible by 5: "+sum);* }  
 *int* calc(*int* x){  
 *if*(x==40) *return* 0;  
 *return* x+calc(x-5);  
 }  
 *public static void* main(String[] args) {  
 Ranged\_SUm obj=*new* Ranged\_SUm();  
 *long* start = System.nanoTime();  
  
 *// call the method  
 int* ans=obj.calc(245);  
 *// get the end time  
 long* end = System.nanoTime();  
  
 *// execution time  
 long* execution = end - start;  
*// obj.compute();  
// System.out.println(obj.calc(245));  
 long* start1 = System.nanoTime();  
  
 *// call the method  
 int* t=obj.compute();  
  
 *// get the end time  
 long* end1 = System.nanoTime();  
  
 *// execution time  
 long* execution1 = end1 - start1;  
 System.out.println("Execution time by recursion: " + execution + " nanoseconds");  
 System.out.println("Execution time by iteration: " + execution1 + " nanoseconds");  
  
 System.out.println("Output:"+ans);  
  
 }  
}

Output:



Experiment: 4

**Date of performance: 31 August 2022**

**Date of Submission: 19 October 2022**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

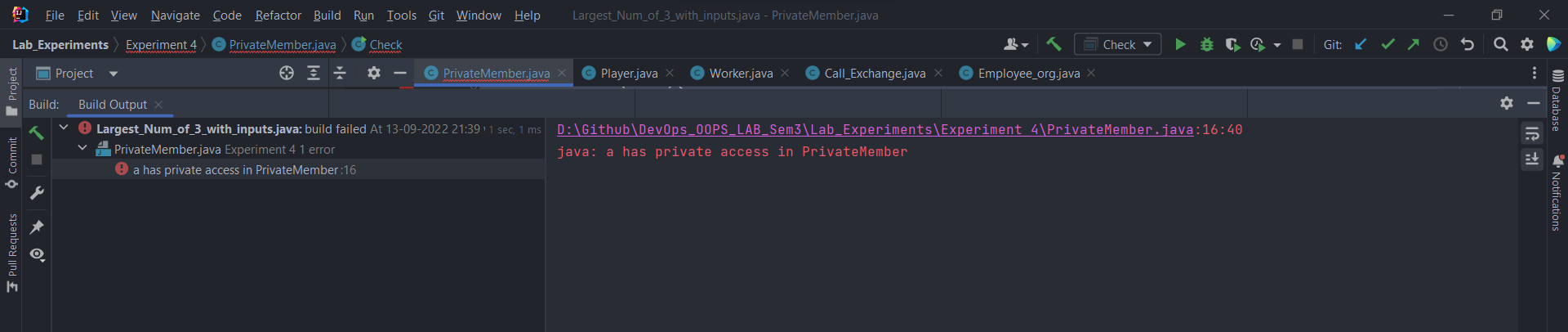
Title: Inheritance

1. Write a Java program to show that private member of a super class cannot be accessed from derived classes.

Code:

class PrivateMember {  
 private int a;  
 PrivateMember(int b){  
 this.a=b;  
 }  
 void display(){  
 System.out.println(this.a);  
 }  
}  
class Check extends PrivateMember{  
 Check(){  
 super(10);  
 }  
 void printing(){  
 System.out.println("This is inside derived class");  
// Cannot be accessed because private members are visible only to inside class in which they are declared  
// System.out.println("Var:"+super.a);  
 }  
 public static void main(String[] args) {  
 Check obj = new Check();  
 obj.display();  
 obj.printing();  
 }  
}

Output:

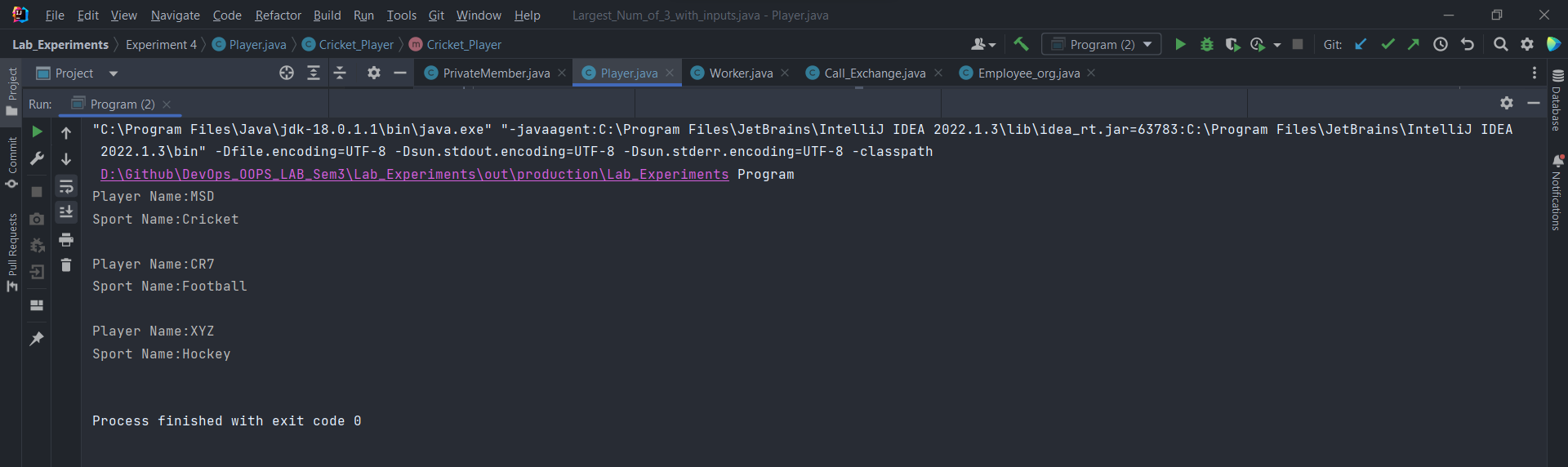


1. Write a program in Java to create a Player class. Inherit the classes Cricket \_Player, Football\_Player and Hockey\_ Player from Player class.

Code:

class Player {  
 String name;  
 String sport;  
 Player(String name,String sport){  
 this.name = name;  
 this.sport = sport;  
 System.out.println("Player Name:" + this.name);  
 System.out.println("Sport Name:" + this.sport + "\n");  
 }  
}  
class Cricket\_Player extends Player{  
  
 Cricket\_Player(String name, String sport) {  
 super(name, sport);  
 }  
}  
class Football\_Player extends Player{  
 Football\_Player(String name, String sport) {  
 super(name, sport);  
 }  
}  
class Hockey\_Player extends Player{  
 Hockey\_Player(String name, String sport) {  
 super(name, sport);  
 }  
}  
class Program{  
 public static void main(String[] args) {  
 Cricket\_Player ob1=new Cricket\_Player("MSD","Cricket");  
  
 Football\_Player ob2=new Football\_Player("CR7","Football");  
  
 Hockey\_Player ob3=new Hockey\_Player("XYZ","Hockey");  
 }  
}

Output:

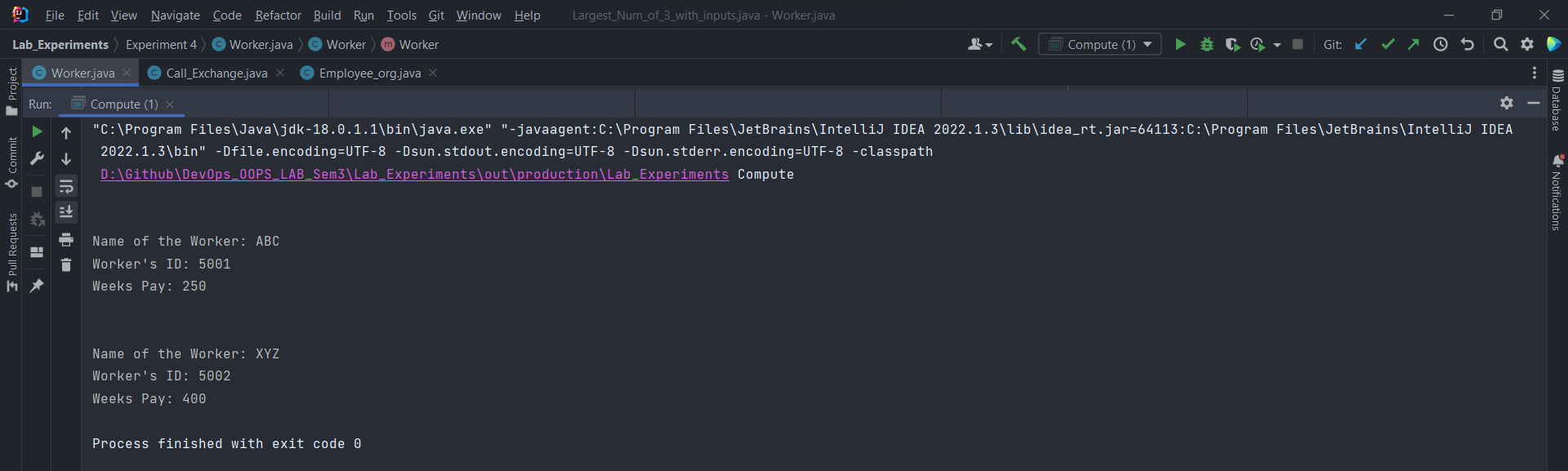


1. Write a class Worker and derive classes *DailyWorker* and *SalariedWorker* from it. Every worker has a name and a salary rate. Write method *ComPay (int hours)* to compute the week pay of every worker. A Daily Worker is paid on the basis of the number of days he/she works. The *SalariedWorker* gets paid the wage for 40 hours a week no matter what the actual hours are. Test this program to calculate the pay of workers. You are expected to use the concept of polymorphism to write this program.

Code:

*public class* Worker {  
 String name;  
 String ID;  
 Worker(String name,String ID){  
 *this*.name=name;  
 *this*.ID=ID;  
 }  
 *void* display(){  
*// Function to display Worker's info* System.out.println("\n");  
 System.out.println("Name of the Worker: "+*this*.name);  
 System.out.println("Worker's ID: "+*this*.ID);  
 }  
}  
*class* DailyWorker *extends* Worker{  
 *int* sal\_rate;  
 DailyWorker(String name, String ID,*int* sal\_rate) {  
 *super*(name,ID);  
 *this*.sal\_rate=sal\_rate;  
 }  
 *void* ComPay(*int* hrs){  
 *super*.display();  
 System.out.println("Weeks Pay: "+sal\_rate\*hrs);  
 }  
}  
*class* SalariedWorker *extends* Worker{  
 *int* sal\_rate;  
 SalariedWorker(String name,String ID,*int* sal\_rate){  
 *super*(name,ID);  
 *this*.sal\_rate=sal\_rate;  
 }  
 *void* ComPay(){  
 *super*.display();  
 System.out.println("Weeks Pay: "+40\*sal\_rate);  
 }  
}  
*class* Compute{  
 *public static void* main(String[] args) {  
 DailyWorker ob1=*new* DailyWorker("ABC","5001",5);  
 ob1.ComPay(50);  
 SalariedWorker ob2=*new* SalariedWorker("XYZ","5002",10);  
 ob2.ComPay();  
 }  
}

Output:

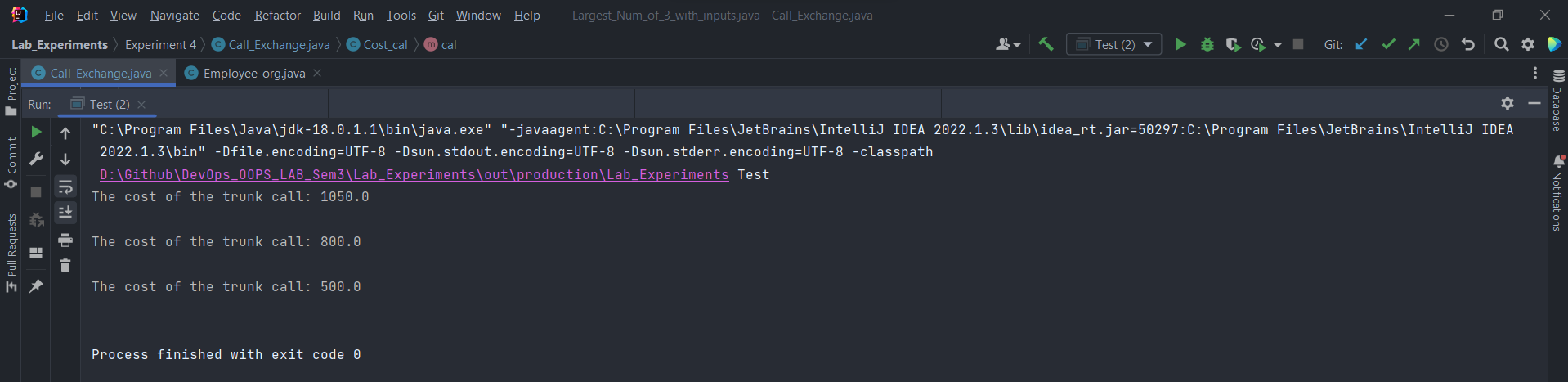


1. Consider the trunk calls of a telephone exchange. A trunk call can be ordinary, urgent or lightning. The charges depend on the duration and the type of the call. Write a program using the concept of polymorphism in Java to calculate the charges.

Code:

*public class* Call\_Exchange {  
 String type;  
 Call\_Exchange(String type){  
 *this*.type=type;  
 }  
 *double* Cal\_rate(){  
 *if*(*this*.type.compareTo("lightning")==0){  
 *return* 3.5;  
 }*else if*(*this*.type.compareTo("urgent")==0){  
 *return* 2.0;  
 }*else if*(*this*.type.compareTo("ordinary")==0){  
 *return* 1.0;  
 }*else*{  
 *return* 0;  
 }  
 }  
}  
*class* Cost\_cal *extends* Call\_Exchange{  
 *int* duration;  
 Cost\_cal(String name,*int* duration){  
 *super*(name);  
 *this*.duration=duration;  
 }  
 *void* cal(){  
 *double* rate=*super*.Cal\_rate();  
 System.out.println("The cost of the trunk call: "+*this*.duration\*rate+"\n");  
 }  
}  
*class* Test{  
 *public static void* main(String[] args) {  
 Cost\_cal obj=*new* Cost\_cal("lightning",300);  
 Cost\_cal ob1=*new* Cost\_cal("urgent",400);  
 Cost\_cal ob2=*new* Cost\_cal("ordinary",500);  
 obj.cal();  
 ob1.cal();  
 ob2.cal();  
 }  
}

Output:

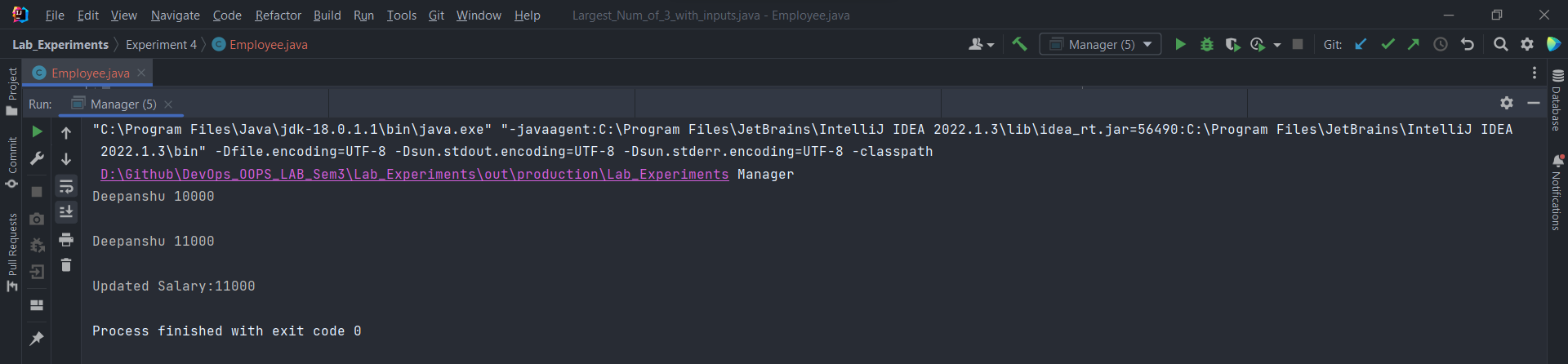


1. Design a class employee of an organization. An employee has a name, empid, and salary. Write the default constructor, a constructor with parameters (name, empid, and salary) and methods to return name and salary. Also write a method *increaseSalary* that raises the employee’s salary by a certain user specified percentage. Derive a subclass Manager from employee. Add an instance variable named department to the manager class. Supply a test program that uses theses classes and methods.

Code:

*class* Employee{  
 String name;  
 *int* empid;  
 *int* salary;  
 Employee(String name,*int* empid,*int* salary){  
 *this*.name=name;  
 *this*.empid=empid;  
 *this*.salary=salary;  
 }  
 *public void* employeeDetails(Employee obj){  
 System.out.println(obj.name+" "+String.valueOf(obj.salary));  
 }  
 *public void* increaseSalary(Employee obj,*int* incrementPerc){  
 obj.salary=obj.salary+(((incrementPerc)\*obj.salary)/100);  
 }  
}  
*class* Manager *extends* Employee{  
 Manager(String name, *int* empid, *int* salary) {  
 *super*(name, empid, salary);  
 }  
  
 *public static void* main(String[] args) {  
 Manager manager=*new* Manager("",0,0);  
 Employee employee=*new* Employee("Deepanshu",5001,10000);  
 String test="";  
 manager.employeeDetails(employee);  
 System.out.println(test);  
 manager.increaseSalary(employee,10);  
 manager.employeeDetails(employee);  
 System.out.println("\nUpdated Salary:"+employee.salary);  
 }  
  
}

Output:



Experiment 5

**Date of performance: 7 September 2022**

**Date of Submission: 19 October 2022**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

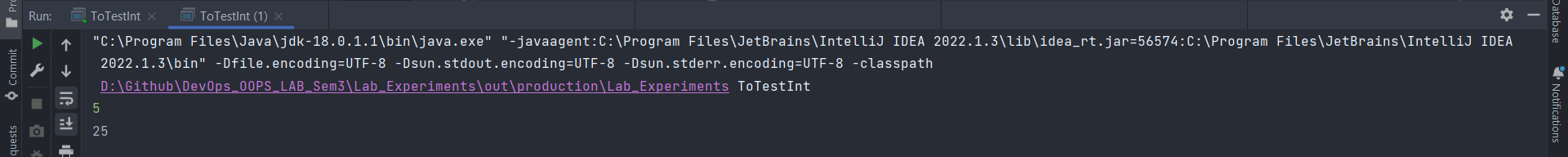
Title: Interface

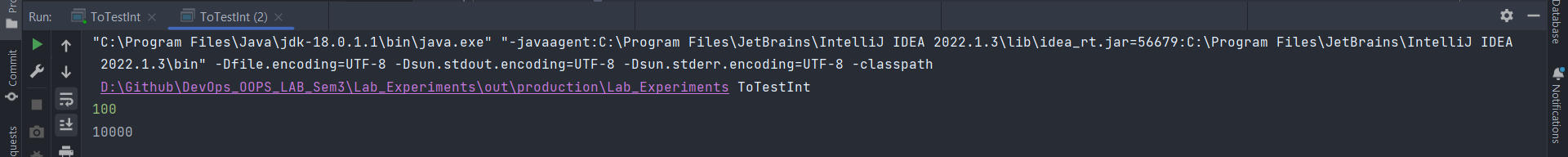
1. Write a program to create interface named test. In this interface the member function is square. Implement this interface in arithmetic class. Create one new class called ToTestInt. In this class use the object of arithmetic class.

Code:

import java.util.Scanner;  
  
interface Test{  
 int square(int a);  
}  
  
class Arithmetic implements Test{  
 public int square(int a){  
 return a\*a;  
 }  
}  
  
public class ToTestInt {  
 public static void main(String[] args) {  
 Arithmetic obj=new Arithmetic();  
 Scanner sc=new Scanner(System.in);  
 int a=sc.nextInt();  
 System.out.println(obj.square(a));  
 }  
}

Output:



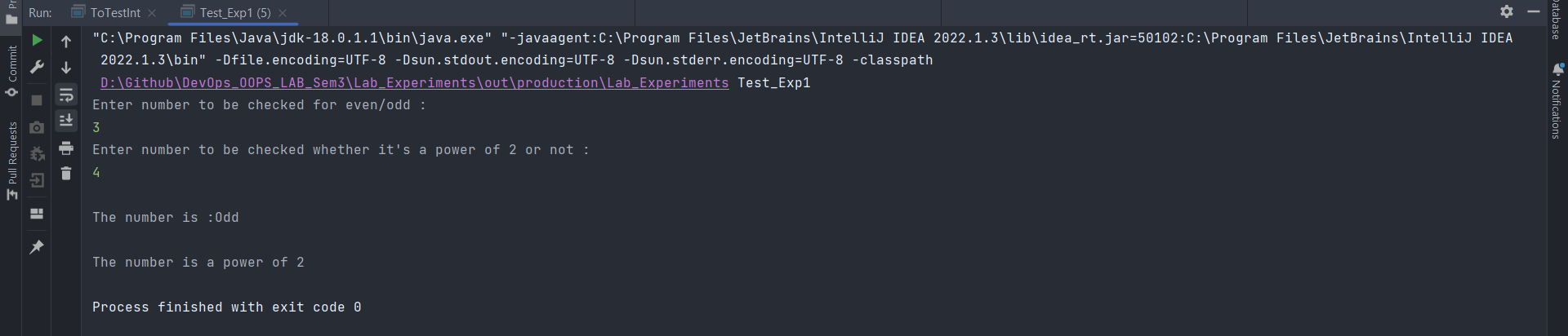


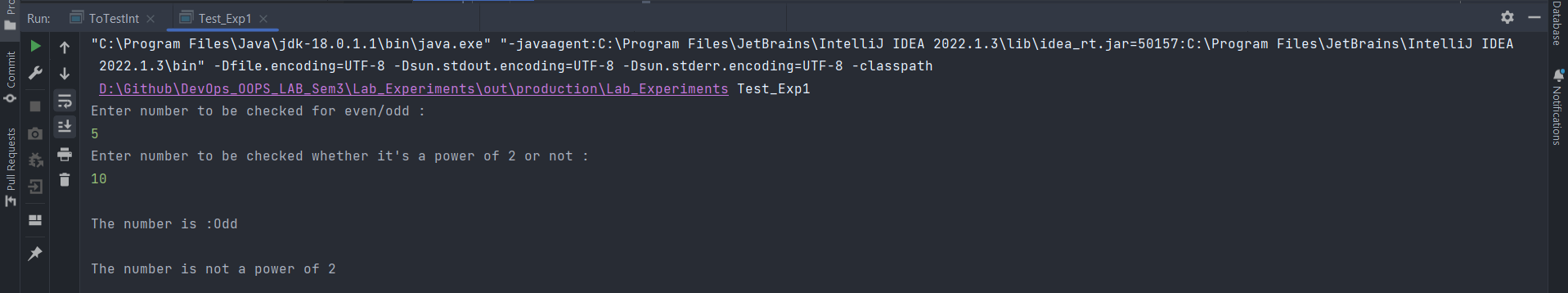
1. Write a program to create interface A, in this interface we have two method meth1 and meth2.Implements this interface in another class named MyClass.

Code:

import java.util.Scanner;  
  
interface A{  
 void meth1(int x);  
 void meth2(int x);  
}  
  
public class MyClass implements A{  
 @Override  
 public void meth1(int x) {  
// checks if number is even or odd  
 if((x&1)==1){  
 System.out.println("\nThe number is :Odd");  
 }else{  
 System.out.println("\nThe number is : Even");  
 }  
 }  
  
 @Override  
 public void meth2(int x) {  
// checks if number is a power of 2  
 if((x&(x-1))==0){  
 System.out.println("\nThe number is a power of 2");  
 }else{  
 System.out.println("\nThe number is not a power of 2");  
 }  
 }  
}  
class Test\_Exp1{  
 public static void main(String[] args) {  
 Scanner sc=new Scanner(System.in);  
 System.out.println("Enter number to be checked for even/odd :");  
 int a=sc.nextInt();  
 System.out.println("Enter number to be checked whether it's a power of 2 or not :");  
 int b=sc.nextInt();  
 MyClass obj=new MyClass();  
 obj.meth1(a);  
 obj.meth2(b);  
 }  
}

Output:



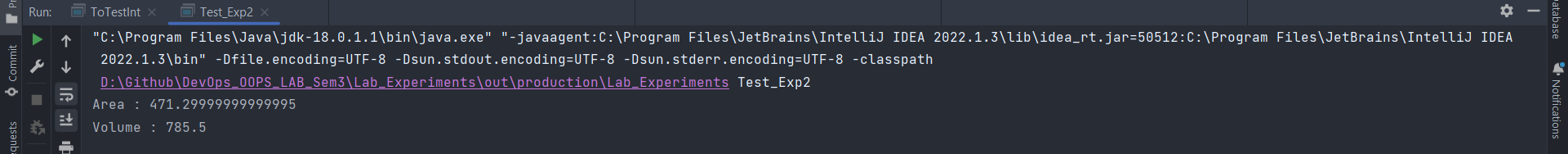


1. Write a program in Java to show the usefulness of Interfaces as a place to keep constant value of the program.

Code:

interface Const{  
 double r = 5.0;  
 double h = 10.0;  
 double pi = 3.142;  
 void area();  
 void volume();  
}  
public class Computing implements Const {  
// Area and Volume for Cylinder  
 @Override  
 public void area() {  
 double ans = (2\*pi\*r\*h)+(2\*pi\*r\*r);  
 System.out.println("Area : "+ans);  
 }  
  
 @Override  
 public void volume() {  
 double ans = pi\*r\*r\*h;  
 System.out.println("Volume : "+ans);  
 }  
}  
class Test\_Exp2{  
 public static void main(String[] args) {  
 Computing obj=new Computing();  
 obj.area();  
 obj.volume();  
 }  
}

Output:

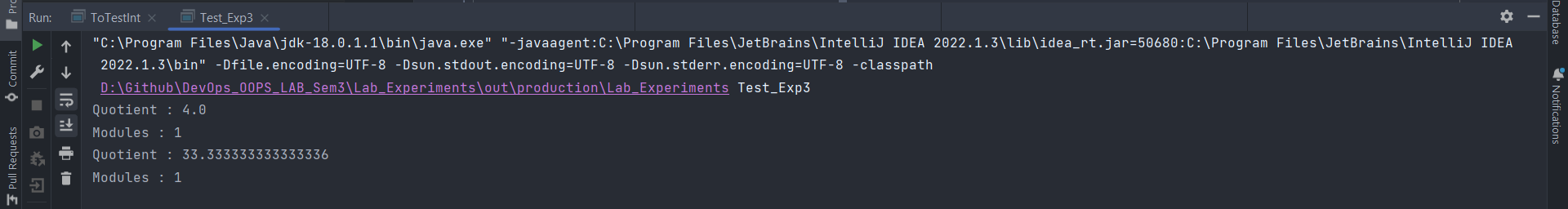


1. Write a program to create an Interface having two methods division and modules. Create a class, which overrides these methods.

Code:

interface Operations{  
 void division(int a, int b);  
 void modulo(int a, int b);  
}  
public class Evaluate implements Operations {  
 @Override  
 public void division(int a, int b) {  
 System.out.println("Quotient : " + ((double)a / b));  
 }  
  
 @Override  
 public void modulo(int a, int b) {  
 System.out.println("Modules : " + a % b);  
 }  
}  
class Test\_Exp3{  
 public static void main(String[] args) {  
 Evaluate obj = new Evaluate();  
 obj.division(20,5);  
 obj.modulo(5,4);  
 obj.division(100,3);  
 obj.modulo(1,4);  
 }  
}

Output:



Experiment 6

**Date of performance: 21 September 2022**

**Date of Submission: 19 October 2022**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

Title: Package

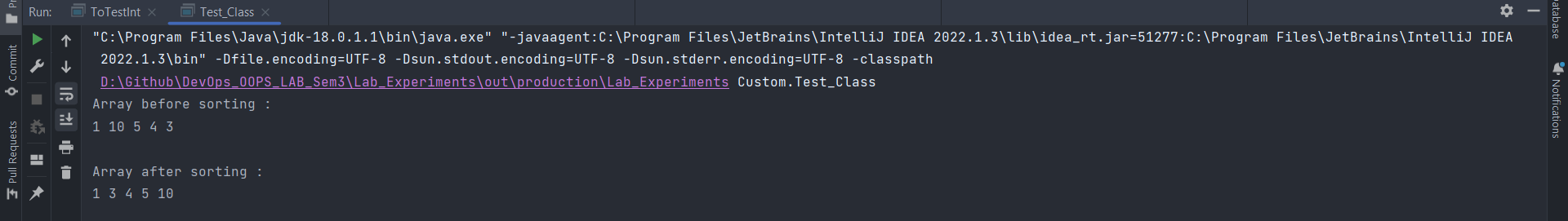
1. Write a Java program to implement the concept of importing classes from user defined package and created packages.

Code:

package Custom;  
  
public class Arr\_Display {  
 public static void display(int[] arr){  
 for(int i : arr){  
 System.out.print(i +" ");  
 }  
 System.out.println("\n");  
 }  
 public static void main(String[] args) {  
 display(new int[]{-1,-1});  
 }  
}

package Custom;  
  
import java.util.Arrays;  
  
import static Custom.Arr\_Display.display;  
  
public class Test\_Class {  
 public void Arr(){  
 int[] arr = {1,10,5,4,3};  
 System.out.println("Array before sorting : ");  
 display(arr);  
 Arrays.sort(arr);  
 System.out.println("Array after sorting : ");  
 display(arr);  
 }  
 public static void main(String[] args) {  
 Test\_Class obj = new Test\_Class();  
 obj.Arr();  
 }  
}

Output:



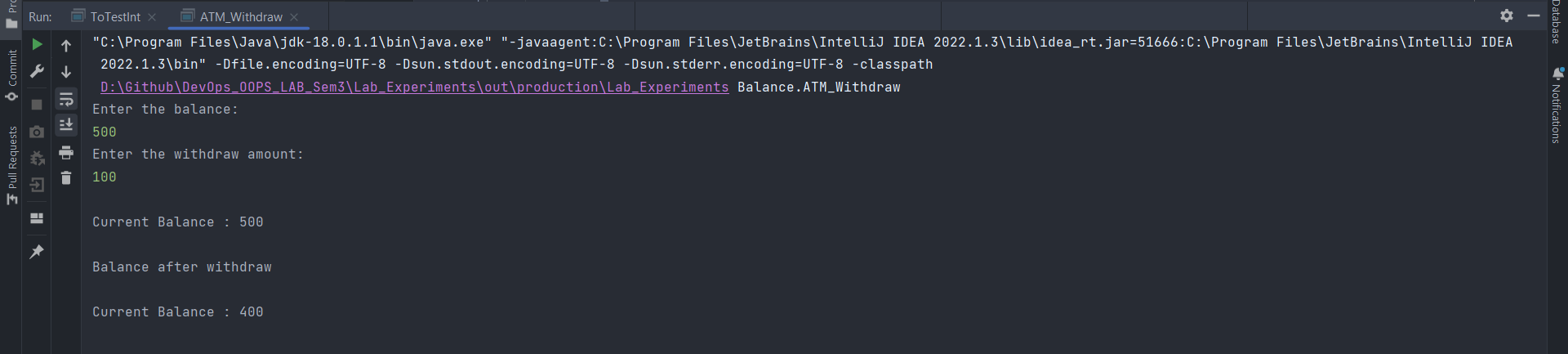
1. Write a program to make a package Balance. This has an Account class with Display\_Balance method. Import Balance package in another program to access Display\_Balance method of Account class.

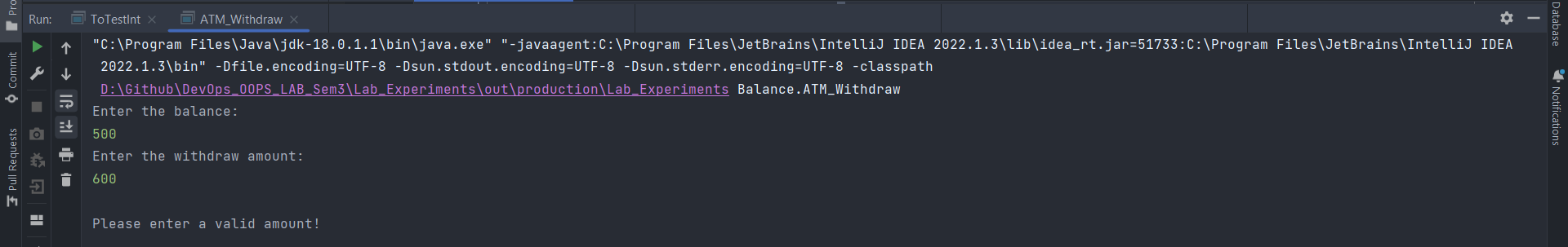
Code:

package Balance;  
  
public class Account {  
 public static void Display\_Balance(int x){  
 System.out.println("Current Balance : " + x + "\n");  
 }  
  
 public static void main(String[] args) {  
 Display\_Balance(0);  
 }  
}

package Balance;  
  
import java.util.Scanner;  
  
import static Balance.Account.Display\_Balance;  
  
public class ATM\_Withdraw {  
 public void info(int bal,int withdraw){  
 if(withdraw > bal){  
 System.out.println("Please enter a valid amount!");  
 }else{  
 Display\_Balance(bal);  
 bal -= withdraw;  
 System.out.println("Balance after withdraw\n");  
 Display\_Balance(bal);  
 }  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
 int bal = sc.nextInt();  
 int wtdraw = sc.nextInt();  
 ATM\_Withdraw obj = new ATM\_Withdraw();  
 obj.info(bal,wtdraw);  
 }  
}

Output:





Experiment 7

**Date of performance: 28 September 2022**

**Date of Submission: 19 October 2022**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

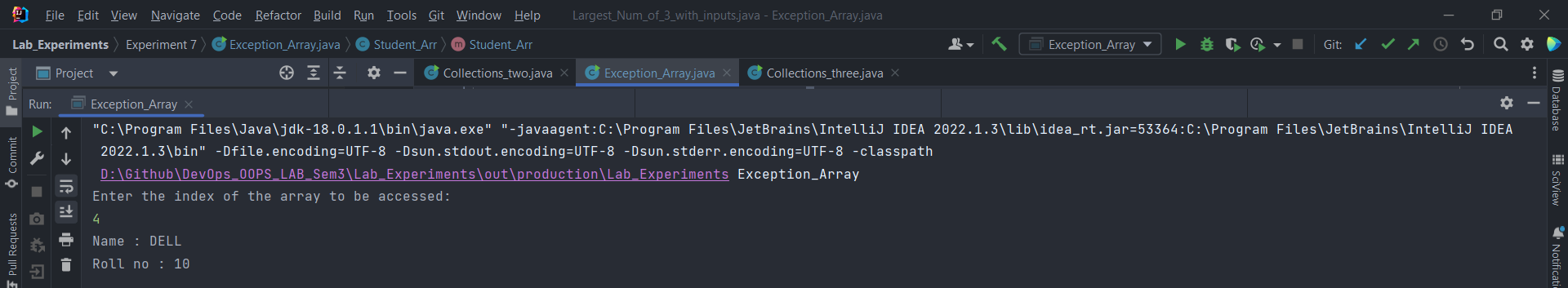
Title: Exceptions

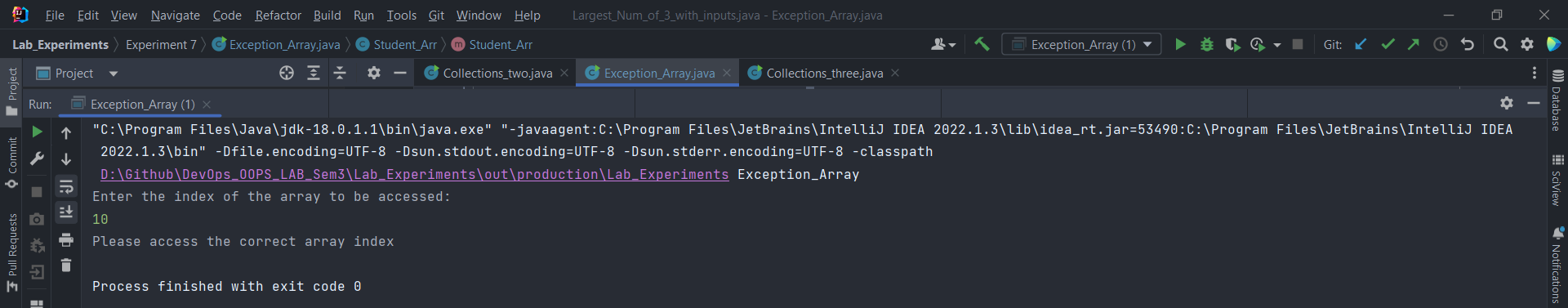
1. Write a program in Java to display the names and roll numbers of students. Initialize respective array variables for 10 students. Handle ArrayIndexOutOfBoundsExeption, so that any such problem doesn’t cause illegal termination of program.

Code:

class Student\_Arr {  
 String name;  
 int roll\_no;  
 Student\_Arr(String name,int roll\_no){  
 this.name = name;  
 this.roll\_no = roll\_no;  
 }  
 public void printing(String s,int rn) {  
 System.out.println("Name : +" + s + "Roll no : " + rn);  
 }  
}  
public class Exception\_Array {  
 public void helper() {  
 Student\_Arr[] arr = new Student\_Arr[10];  
 arr[0] = new Student\_Arr("Deepanshu",692);  
 arr[1] = new Student\_Arr("Deepanshu",692);  
 arr[2] = new Student\_Arr("Deepanshu",692);  
 arr[3] = new Student\_Arr("Deepanshu",692);  
 arr[4] = new Student\_Arr("Deepanshu",692);  
 arr[5] = new Student\_Arr("Deepanshu",692);  
 arr[6] = new Student\_Arr("Deepanshu",692);  
 arr[7] = new Student\_Arr("Deepanshu",692);  
 arr[8] = new Student\_Arr("Deepanshu",692);  
 arr[9] = new Student\_Arr("Deepanshu",692);  
 try {  
 arr[10] = new Student\_Arr("random",10);  
 arr[1].printing("random\_code",123);  
 } catch(ArrayIndexOutOfBoundsException e) {  
 System.out.println("Please access the correct array index");  
 }  
 }  
 public static void main(String[] args) {  
 Exception\_Array obj = new Exception\_Array();  
 obj.helper();  
 }  
}

Output:



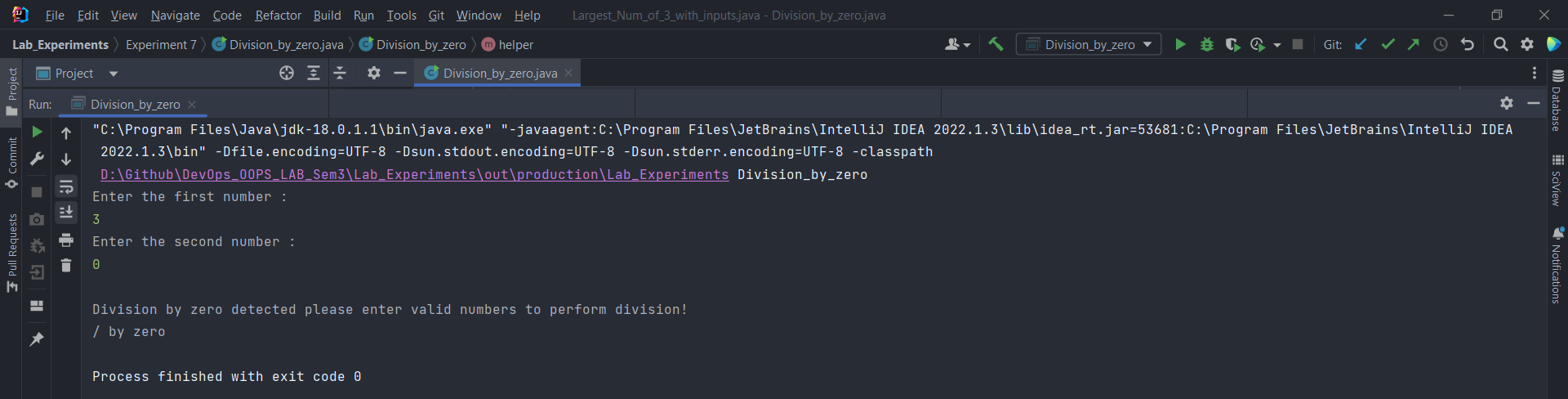


1. Write a Java program to enable the user to handle any chance of divide by zero exception.

Code:

import java.util.Scanner;  
  
class Exception\_zero {  
 public void division(int a,int b) {  
 try {  
 int q = a / b;  
 System.out.println("\nThe division of a by b : "+q);  
 } catch(ArithmeticException e) {  
 System.out.println("\nDivision by zero detected please enter valid numbers to perform division!");  
 System.out.println(e.getMessage());  
 }  
 }  
}  
public class Division\_by\_zero {  
 public void helper(){  
 Exception\_zero obx = new Exception\_zero();  
 Scanner sc = new Scanner(System.in);  
 System.out.println("Enter the first number :");  
 int a = sc.nextInt();  
 System.out.println("Enter the second number :");  
 int b = sc.nextInt();  
 obx.division(a,b);  
 }  
 public static void main(String[] args) {  
 Division\_by\_zero obj = new Division\_by\_zero();  
 obj.helper();  
 }  
}

Output:

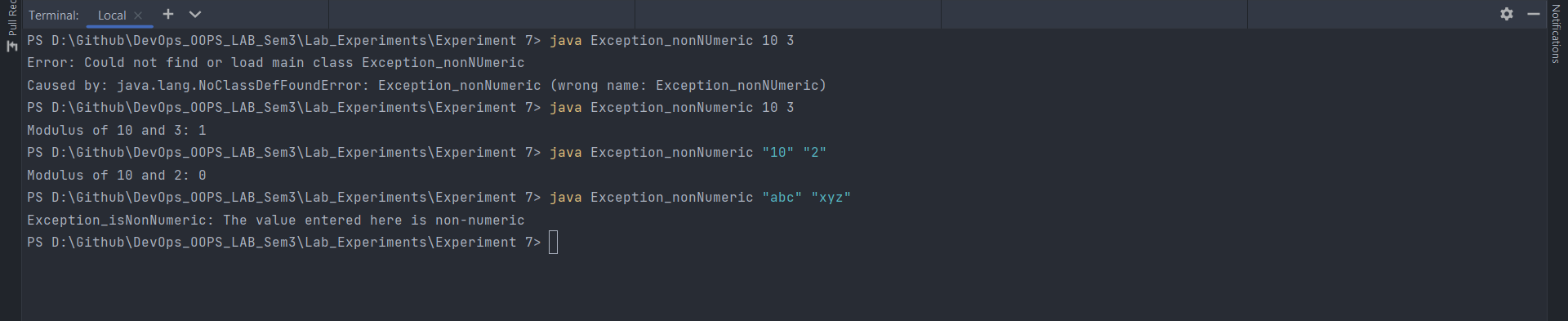


1. Create an exception class, which throws an exception if operand is nonnumeric in calculating modules. (Use command line arguments).

Code:

class Exception\_isNonNumeric extends Exception {  
 Exception\_isNonNumeric(){  
 super("The value entered here is non-numeric");  
 }  
}  
public class Exception\_nonNumeric {  
 public static boolean check\_for\_string(String p,String q) {  
 try {  
 int a = Integer.parseInt(p);  
 int b = Integer.parseInt(q);  
 } catch(NumberFormatException e) {  
 return false;  
 }  
 return true;  
 }  
 public static void main(String[] args) {  
 try {  
 if (check\_for\_string(args[0],args[1])) {  
 int x = Integer.parseInt(args[0]);  
 int y = Integer.parseInt(args[1]);  
 System.out.println("Modulus of " + x + " and " + y + ": " +(x%y));  
 } else {  
 throw new Exception\_isNonNumeric();  
 }  
 } catch (Exception\_isNonNumeric e) {  
 System.out.println(e);  
 }  
 }  
}

Output:

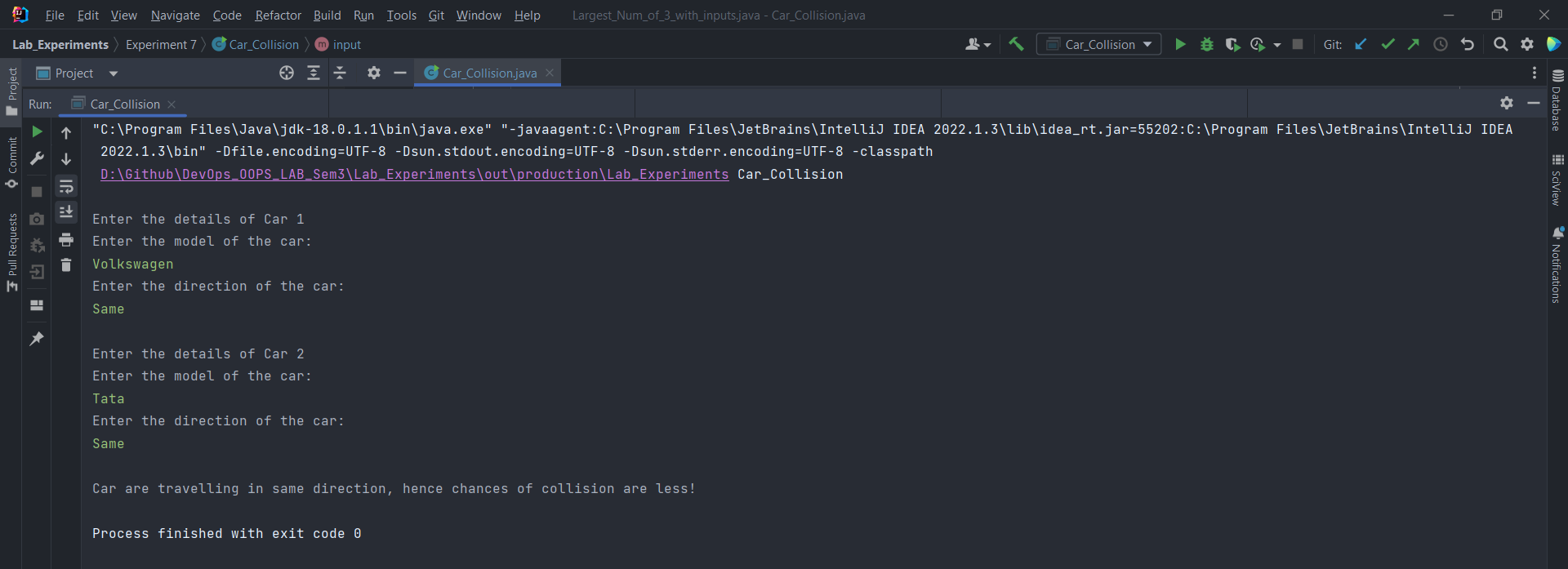


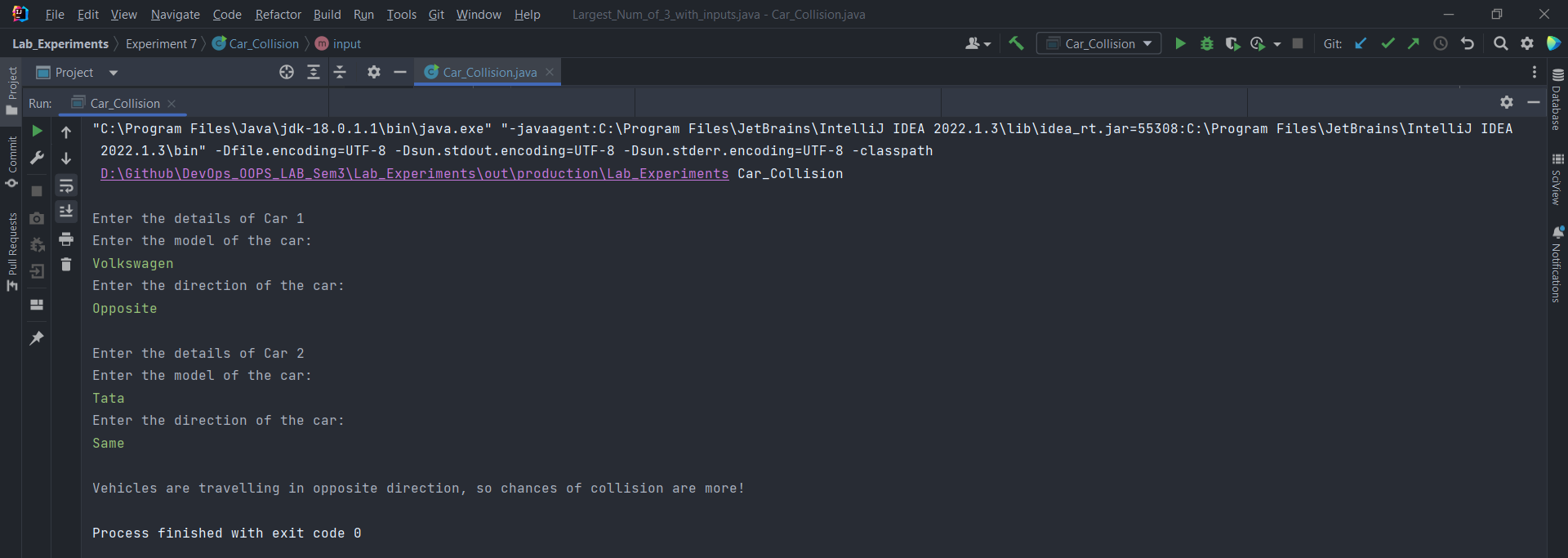
1. On a single track two vehicles are running. As vehicles are going in same direction there is no problem. If the vehicles are running in different direction there is a chance of collision. To avoid collisions, write a Java program using exception handling. You are free to make necessary assumptions.

Code:

import java.util.Scanner;  
public class Car\_Collision {  
  
 public void compute(String direction1,String direction2) throws Exception {  
 try{  
 if (!direction1.equals(direction2)) {  
 throw new Exception();  
 }  
 System.out.println("\nCar are travelling in same direction, hence chances of collision are less!");  
 } catch (Exception e) {  
 System.out.println("\nVehicles are travelling in opposite direction, so chances of collision are more!");  
 }  
 }  
 public void input() throws Exception {  
 String model1,direction1,model2,direction2;  
 Scanner sc = new Scanner(System.in);  
 System.out.println("\nEnter the details of Car 1");  
 System.out.println("Enter the model of the car: ");  
 model1 = sc.nextLine();  
 System.out.println("Enter the direction of the car: ");  
 direction1 = sc.nextLine();  
 System.out.println("\nEnter the details of Car 2");  
 System.out.println("Enter the model of the car: ");  
 model2 = sc.nextLine();  
 System.out.println("Enter the direction of the car: ");  
 direction2 = sc.nextLine();  
 compute(direction1,direction2);  
 }  
 public static void main(String[] args) throws Exception {  
 Car\_Collision obx = new Car\_Collision();  
 obx.input();  
 }  
}

Output:





1. Write a java program to throw an exception for an employee detail.

• If an employee name is a number, a name exception must be thrown.

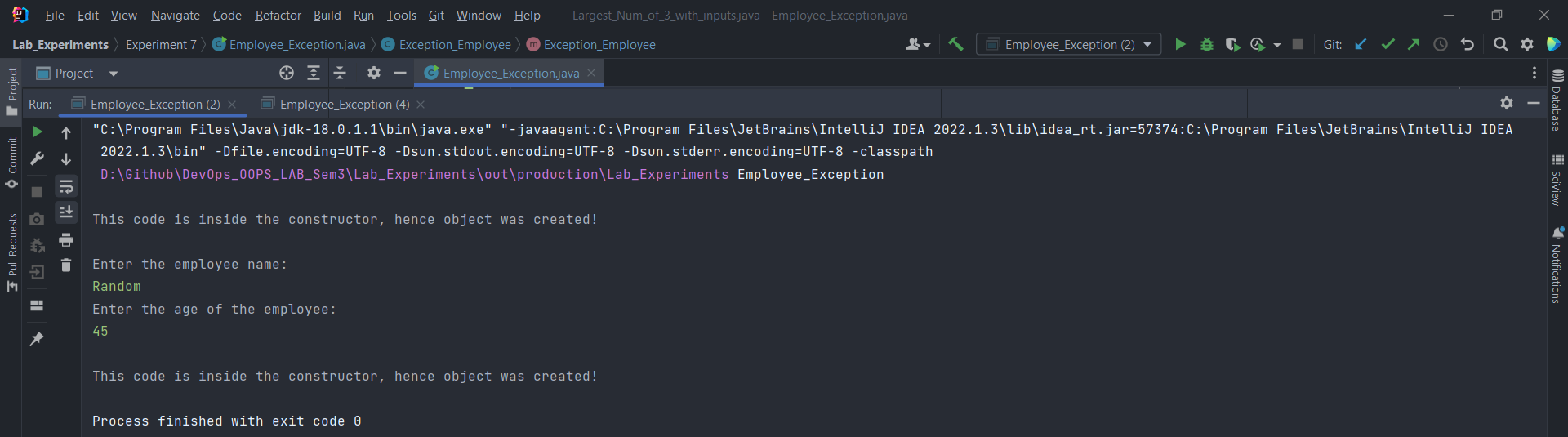
• If an employee age is greater than 50, an age exception must be thrown.

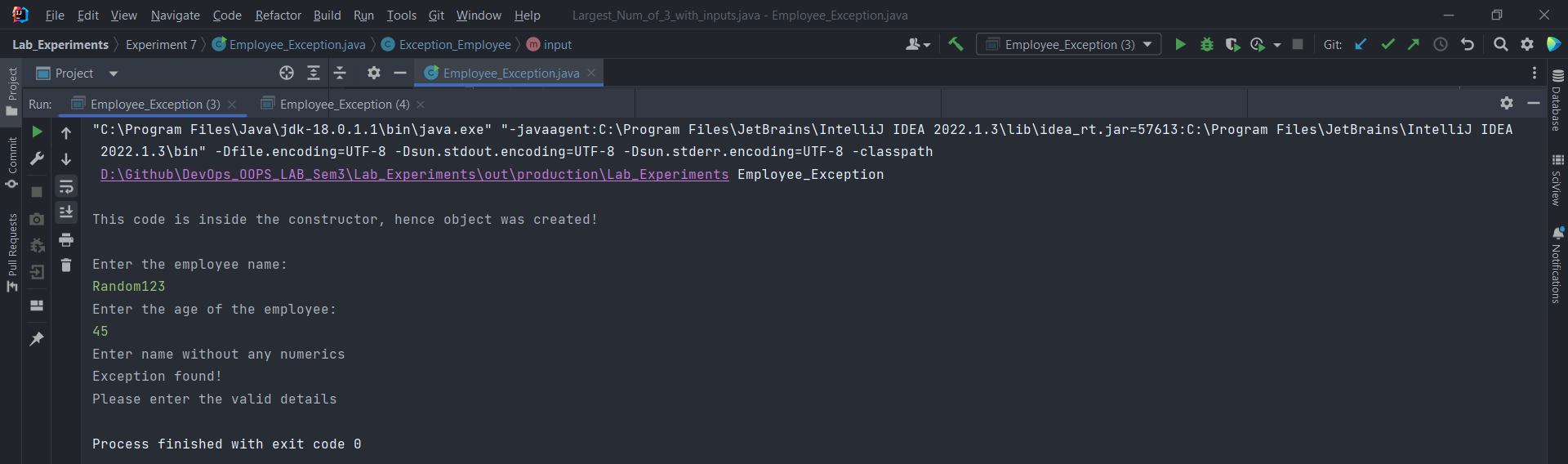
• Or else an object must be created for the entered employee details.

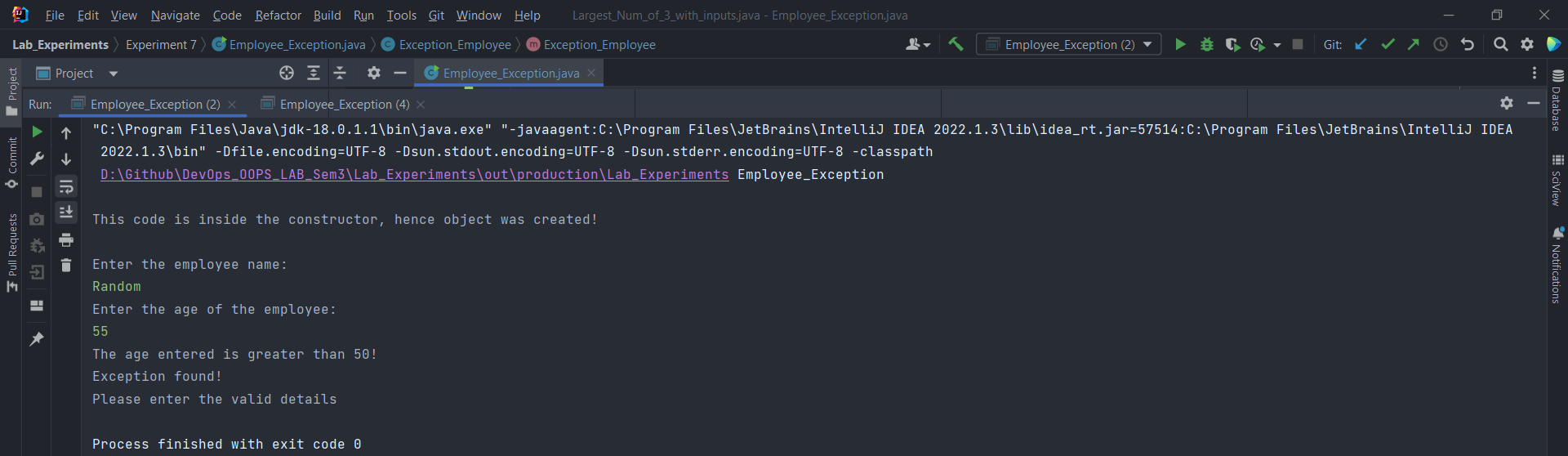
Code:

import java.io.IOException;  
import java.util.Scanner;  
class Exception\_Employee {  
 Exception\_Employee() {  
 /\*  
 if this line executes at the starting of the program, then it is meant for main method of Class Employee\_Exception  
 to call the function "input()" of Class Exception\_Employee for further computation  
 \*/  
 System.out.println("\nThis code is inside the constructor, hence object was created!");  
 }  
 public void input() throws Exception {  
 Scanner sc = new Scanner(System.in);  
 String name;  
 int age;  
 try{  
 System.out.println("\nEnter the employee name: ");  
 name = sc.nextLine();  
 System.out.println("Enter the age of the employee: ");  
 age = sc.nextInt();  
 if(!name.matches("[a-zA-z]+")) {  
 System.out.println("Enter name without any numerics");  
 throw new IOException();  
 }  
 if (age > 50) {  
 System.out.println("The age entered is greater than 50!");  
 throw new Exception();  
 }  
 Exception\_Employee obj = new Exception\_Employee();  
// System.out.println("Object is created!");  
 }  
 catch(Exception e) {  
 System.out.println("Exception found!");  
 System.out.println("Please enter the valid details");  
 }  
 }  
}  
public class Employee\_Exception extends Exception\_Employee {  
 public static void main(String[] args) throws Exception {  
 Employee\_Exception obx = new Employee\_Exception();  
 obx.input();  
 }  
}

Output:







Experiment 8

**Date of performance: 19 Oct 2022**

**Date of Submission: -**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

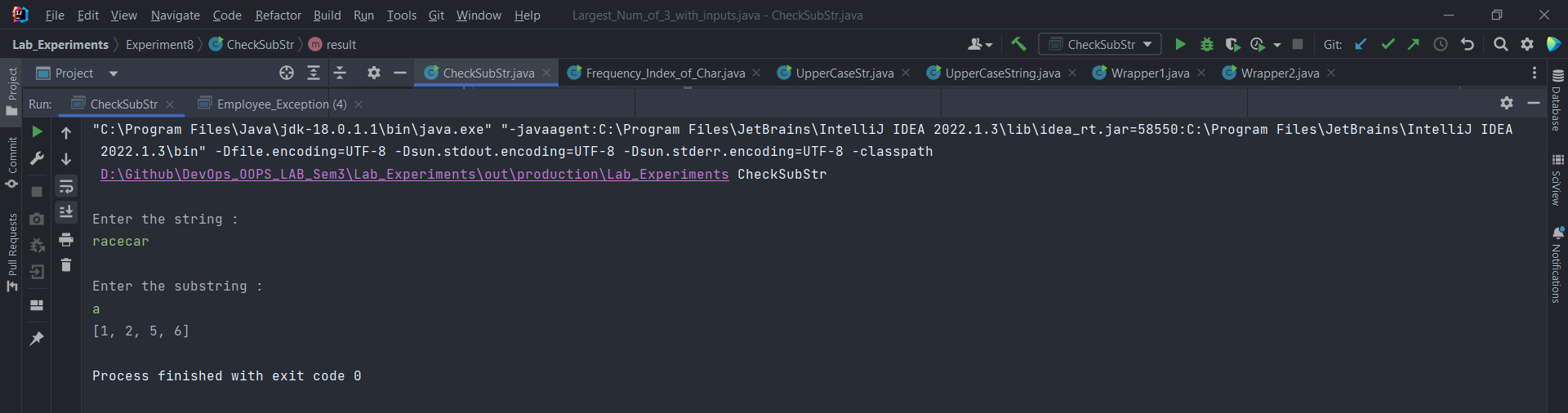
Title: String Handling

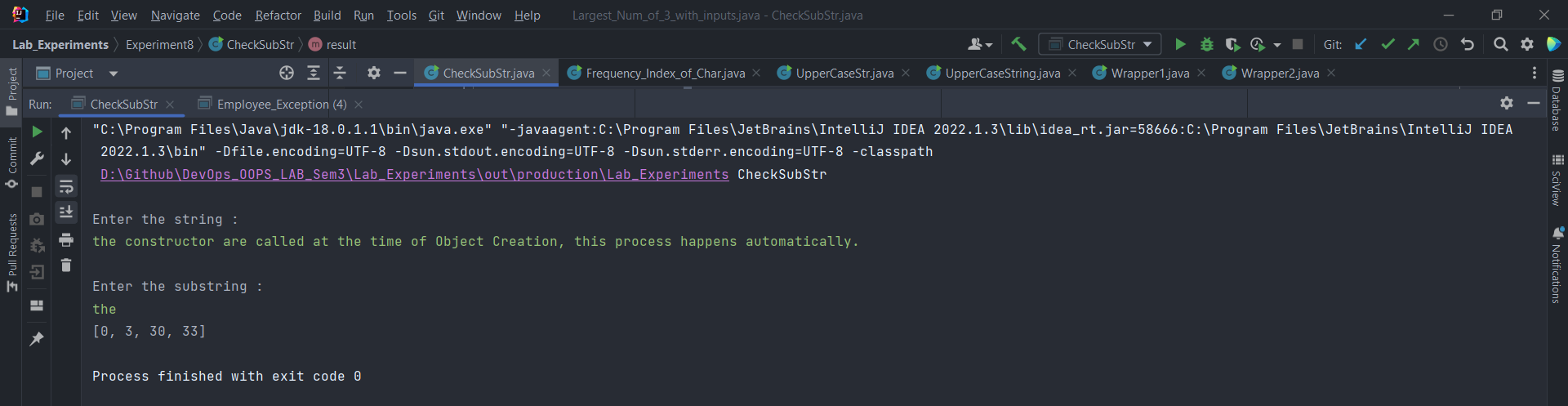
1. Write a program for searching strings for the first occurrence of a character or substring and for the last occurrence of a character or substring.

Code:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Scanner;  
  
public class CheckSubStr {  
 public List<Integer> CheckingSubstr(){  
 Scanner sc = new Scanner(System.in);  
 int k = 0;  
 System.out.println("\nEnter the string : ");  
 String str = sc.nextLine();  
 System.out.println("\nEnter the substring : ");  
 String substr = sc.nextLine();  
 int len = str.length();  
 int sub\_len = substr.length();  
 ArrayList<Integer> index = new ArrayList<Integer>();  
 for (int i = 0; i < len; i++ ) {  
 int j = sub\_len + i;  
 if (j >= len + 1 ) {  
 break;  
 }  
 if (str.substring(i, j).equals(substr)) {  
 index.add(i);  
 index.add(j);  
 break;  
 }  
 }  
 for (int i = len ; i >= 0 ; i-- ) {  
 int j = i - sub\_len;  
 if (j <= -1 ) {  
 break;  
 }  
 if (str.substring(j,i).equals(substr)) {  
 index.add(j);  
 index.add(i);  
 break;  
 }  
 }  
 return index;  
 }  
 public void result() {  
 ArrayList<Integer> ans = (ArrayList<Integer>) CheckingSubstr();  
 System.out.println(ans);  
 }  
 public static void main(String[] args) {  
 CheckSubStr obj = new CheckSubStr();  
 obj.result();  
 }  
}

Output:



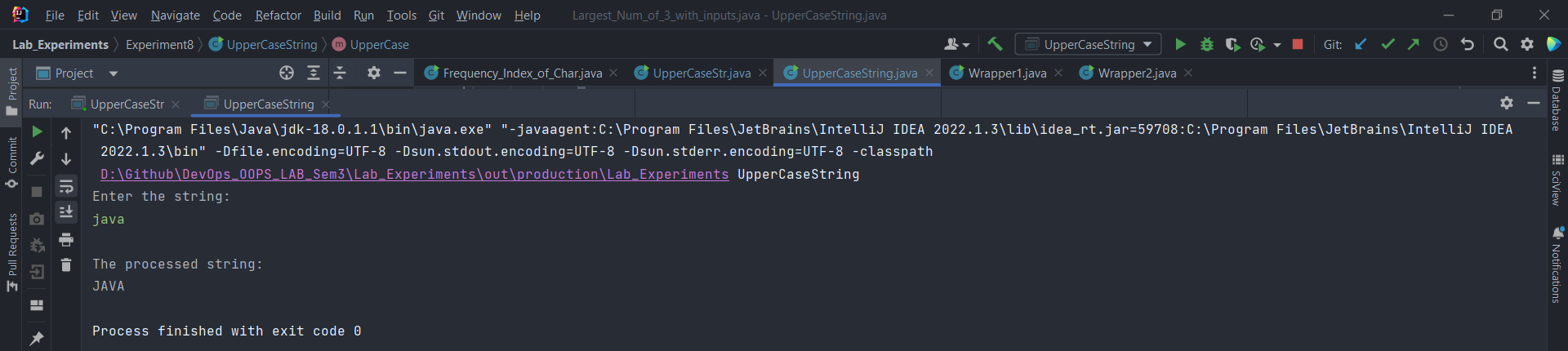


1. Write a program that converts all characters of a string in capital letters. (Use String Buffer to store a string). Don’t use inbuilt function.

Code:

import java.util.Scanner;  
  
public class UpperCaseString {  
 public void UpperCase(){  
 Scanner sc = new Scanner(System.in);  
 System.out.println("Enter the string: ");  
 String str = sc.nextLine();  
 StringBuffer sb = new StringBuffer();  
 for (int i = 0; i < str.length(); i++) {  
 char ch = str.charAt(i);  
 if (ch >= 97 && ch <= 122) {  
 sb.append((char)(ch - 32));  
 }else{  
 sb.append(ch);  
 }  
 }  
 System.out.println("\nThe processed string: \n" + sb);  
 }  
 public static void main(String[] args) {  
 UpperCaseString obj = new UpperCaseString();  
 obj.UpperCase();  
 }  
}

Output:

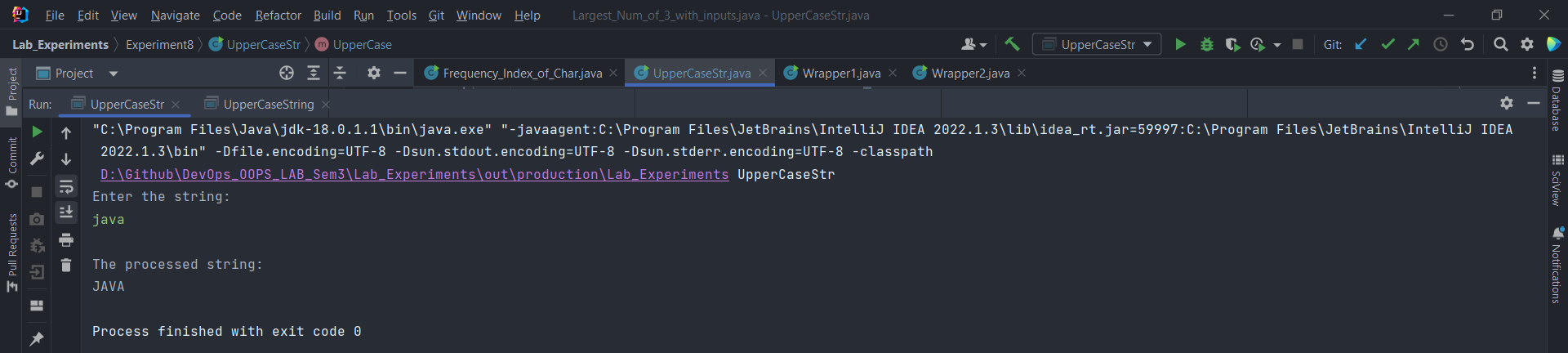


1. Write a program in Java to read a statement from console, convert it into upper case and again print on console. (Don’t use inbuilt function)

Code:

import java.util.Scanner;  
  
public class UpperCaseStr {  
 public void UpperCase(){  
 Scanner sc = new Scanner(System.in);  
 System.out.println("Enter the string: ");  
 String str = sc.nextLine();  
 StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < str.length(); i++) {  
 char ch = str.charAt(i);  
 if (ch >= 97 && ch <= 122) {  
 sb.append((char)(ch - 32));  
 }else{  
 sb.append(ch);  
 }  
 }  
 System.out.println("\nThe processed string: \n" + sb.toString());  
 }  
 public static void main(String[] args) {  
 UpperCaseStr obj = new UpperCaseStr();  
 obj.UpperCase();  
 }  
}

Output:

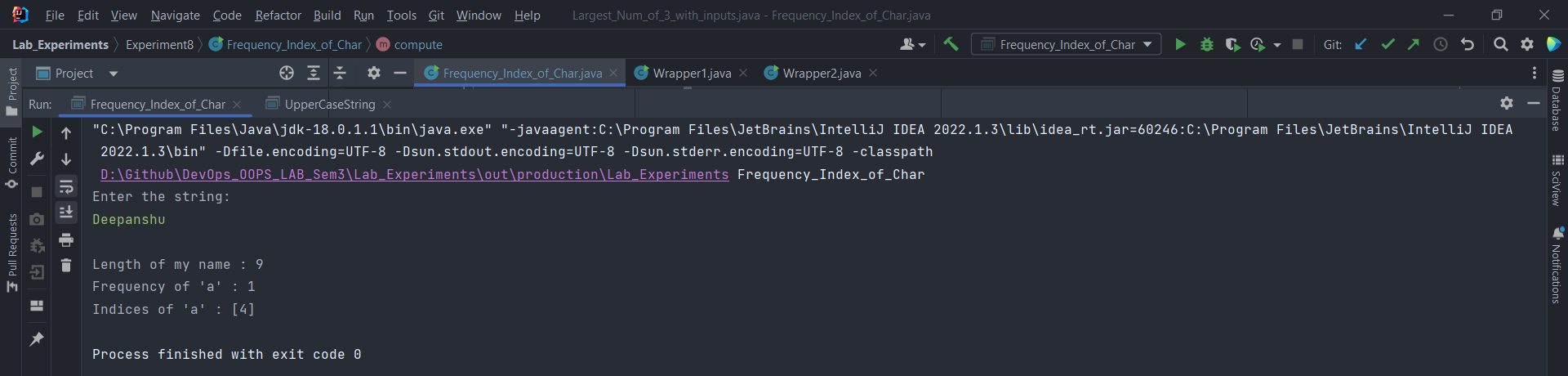


1. Write a program in Java to create a String object. Initialize this object with your name. Find the length of your name using the appropriate String method. Find whether the character ‘a’ is in your name or not; if yes find the number of times ‘a’ appears in your name. Print locations of occurrences of ‘a’. Try the same for different String objects.

Code:

import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Frequency\_Index\_of\_Char {  
 public void compute(){  
 Scanner sc = new Scanner(System.in);  
 System.out.println("Enter the string: ");  
 String str = sc.nextLine();  
 int freq = 0;  
 int len = str.length();  
 ArrayList<Integer> index = new ArrayList<>();  
 for (int i = 0; i < len ; i++) {  
 if (str.charAt(i) == 'a') {  
 freq++;  
 index.add(i);  
 }  
 }  
 System.out.println("\nLength of my name : "+len);  
 System.out.println("Frequency of 'a' : "+freq);  
 System.out.println("Indices of 'a' : "+index);  
 }  
 public static void main(String[] args) {  
 Frequency\_Index\_of\_Char obj = new Frequency\_Index\_of\_Char();  
 obj.compute();  
 }  
}

Output:



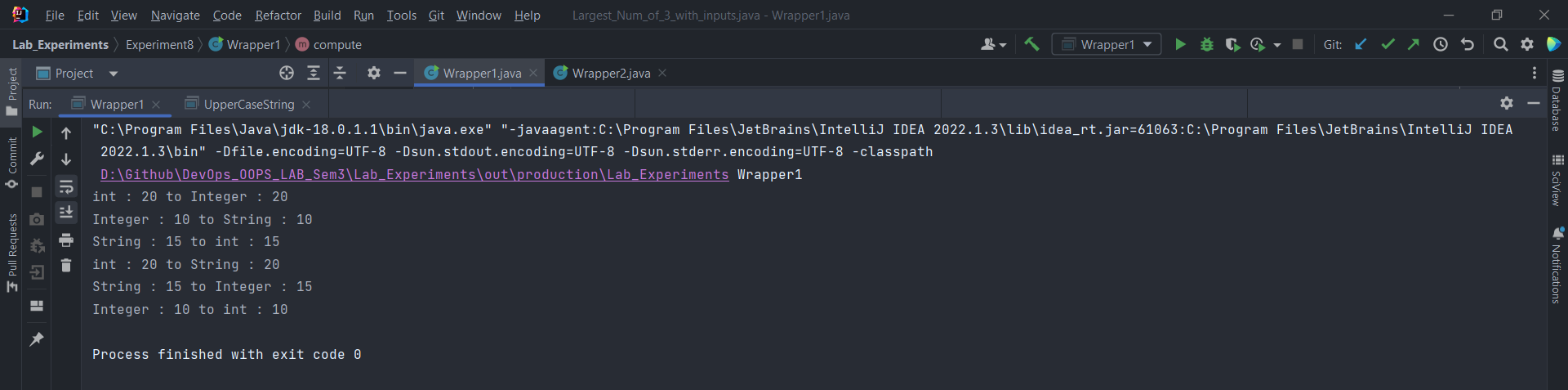
Title: Wrapper Classes

1. Write a Java code that converts int to Integer, converts Integer to String, converts String to int, converts int to String, converts String to Integer converts Integer to int.

Code:

public class Wrapper1 {  
 public void compute(){  
 int i = 20;  
 Integer ob = new Integer(10);  
 String s = "15";  
 int\_to\_Integer(i);  
 Integer\_to\_String(ob);  
 String\_to\_int(s);  
 int\_to\_String(i);  
 String\_to\_Integer(s);  
 Integer\_to\_int(ob);  
 }  
 public void int\_to\_Integer(int x) {  
 Integer I = x;  
 System.out.println("int : " + x + " to Integer : " + I);  
 }  
 public void Integer\_to\_String(Integer I) {  
 String str = I.toString();  
 System.out.println("Integer : " + I + " to String : " + str);  
 }  
 public void String\_to\_int(String str) {  
 int x = Integer.parseInt(str);  
 System.out.println("String : " + str + " to int : " + x);  
 }  
 public void int\_to\_String(int x) {  
 String str = String.valueOf(x);  
 System.out.println("int : " + x + " to String : " + str);  
 }  
 public void String\_to\_Integer(String str) {  
 Integer I = Integer.valueOf(str);  
 System.out.println("String : " + str + " to Integer : " + I);  
 }  
 public void Integer\_to\_int(Integer I) {  
 int x = I.intValue();  
 System.out.println("Integer : " + I + " to int : " + x);  
 }  
 public static void main(String[] args) {  
 Wrapper1 obj = new Wrapper1();  
 obj.compute();  
 }  
}

Output:

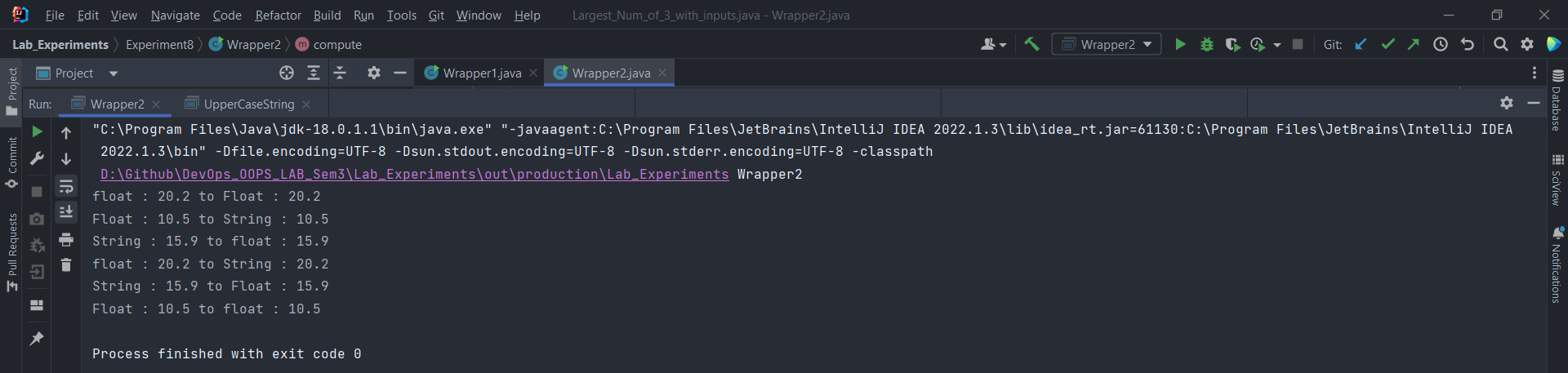


1. Write a Java code that converts float to Float converts Float to String converts String to float, converts float to String converts String to Float converts Float to float.

Code:

public class Wrapper2 {  
 public void compute(){  
 float f = 20.2f;  
 Float ob = new Float(10.5);  
 String s = "15.9";  
 float\_to\_Float(f);  
 Float\_to\_String(ob);  
 String\_to\_float(s);  
 float\_to\_String(f);  
 String\_to\_Float(s);  
 Float\_to\_float(ob);  
 }  
 public void float\_to\_Float(float f) {  
 Float F = f;  
 System.out.println("float : " + f + " to Float : " + F);  
 }  
 public void Float\_to\_String(Float F) {  
 String str = F.toString();  
 System.out.println("Float : " + F + " to String : " + str);  
 }  
 public void String\_to\_float(String str) {  
 float x = Float.parseFloat(str);  
 System.out.println("String : " + str + " to float : " + x);  
 }  
 public void float\_to\_String(float x) {  
 String str = String.valueOf(x);  
 System.out.println("float : " + x + " to String : " + str);  
 }  
 public void String\_to\_Float(String str) {  
 Float F = Float.valueOf(str);  
 System.out.println("String : " + str + " to Float : " + F);  
 }  
 public void Float\_to\_float(Float F) {  
 float x = F.floatValue();  
 System.out.println("Float : " + F + " to float : " + x);  
 }  
 public static void main(String[] args) {  
 Wrapper2 obj = new Wrapper2();  
 obj.compute();  
 }  
}

Output:



Experiment 9

**Date of performance: 19 Oct 2022**

**Date of Submission: -**

**SAP Id: 500097692**

**Roll No.: R2142211410**

**Name of the Student: Deepanshu Rawat**

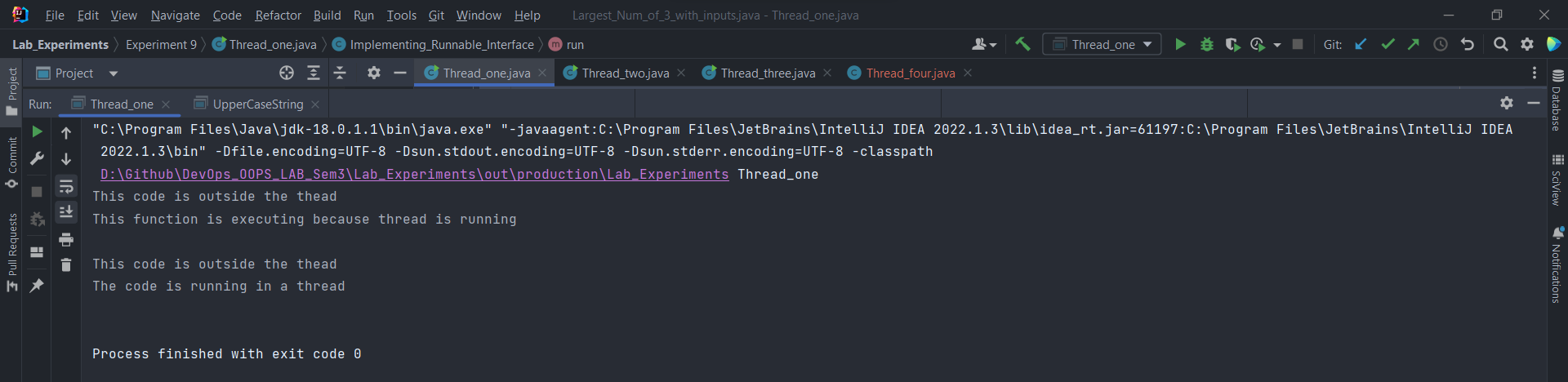
Title: Threads

1. Write a program to implement the concept of threading by extending Thread Class and Runnable interface.

Code:

class Extending\_Thread\_Class extends Thread {  
 public void run() {  
 System.out.println("This function is executing because thread is running");  
 }  
}  
  
class Implementing\_Runnable\_Interface implements Runnable {  
  
 @Override  
 public void run() {  
 System.out.println("The code is running in a thread\n");  
 }  
}  
  
public class Thread\_one extends Thread {  
 public void run() {  
 System.out.println("This function is executing because thread is running");  
 }  
 public static void main(String[] args) {  
 Extending\_Thread\_Class obj = new Extending\_Thread\_Class();  
 obj.start();  
 System.out.println("This code is outside the thead");  
  
  
 Implementing\_Runnable\_Interface obx = new Implementing\_Runnable\_Interface();  
 Thread thread = new Thread(obx);  
 thread.start();  
 System.out.println("\nThis code is outside the thead");  
  
 }  
}

Output:

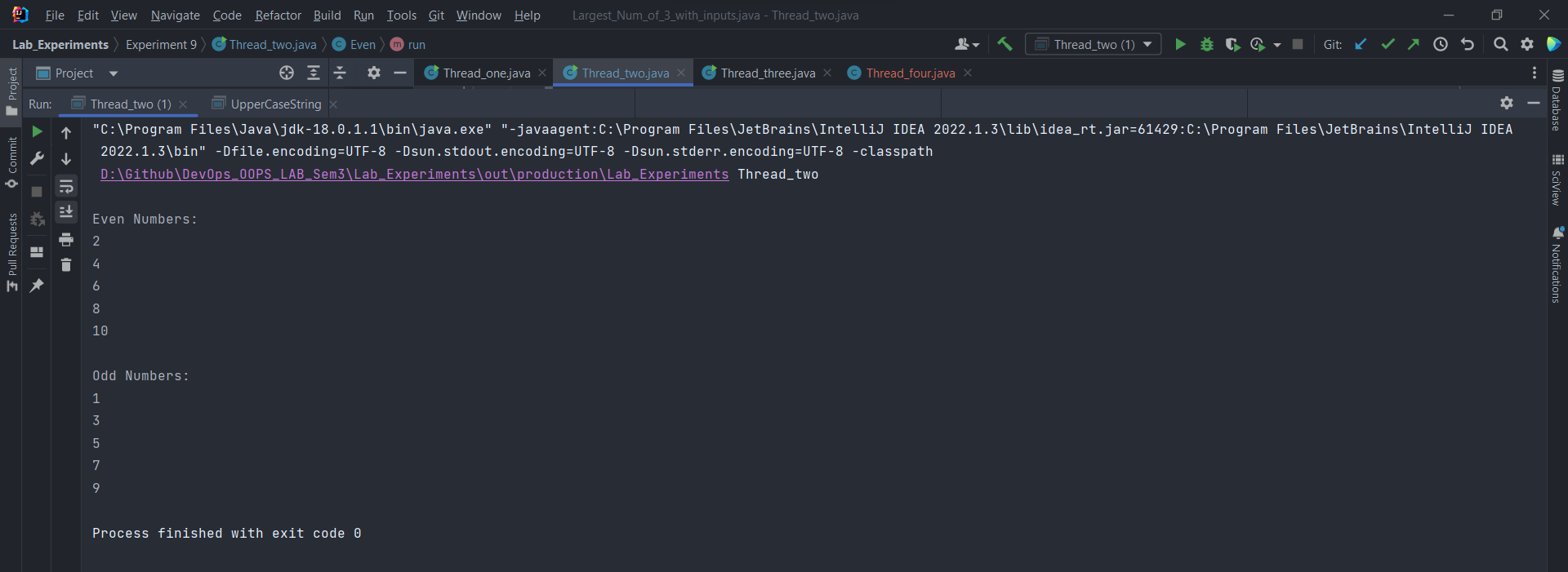


1. Write a program for generating 2 threads, one for printing even numbers and the other for printing odd numbers.

Code:

class Even extends Thread {  
// Thread t;  
 Even () {  
// t = new Thread(this);  
 System.out.println("\nEven Numbers:");  
 start();  
 }  
 @Override  
 public void run() {  
 for (int i = 1; i <= 10; i++) {  
 if ((i&1) != 1) {  
 System.out.println(i);  
 }  
 }  
 }  
}  
  
class Odd extends Thread {  
// Thread t;  
 Odd () {  
// t = new Thread(this);  
 System.out.println("\nOdd Numbers:");  
 start();  
 }  
 @Override  
 public void run() {  
 for (int i = 1; i <= 10; i++) {  
 if ((i&1) == 1) {  
 System.out.println(i);  
 }  
 }  
 }  
}  
  
  
public class Thread\_two {  
 public static void main(String[] args) {  
 Even ob1 = new Even();  
 Odd ob2 = new Odd();  
 }  
}

Output:

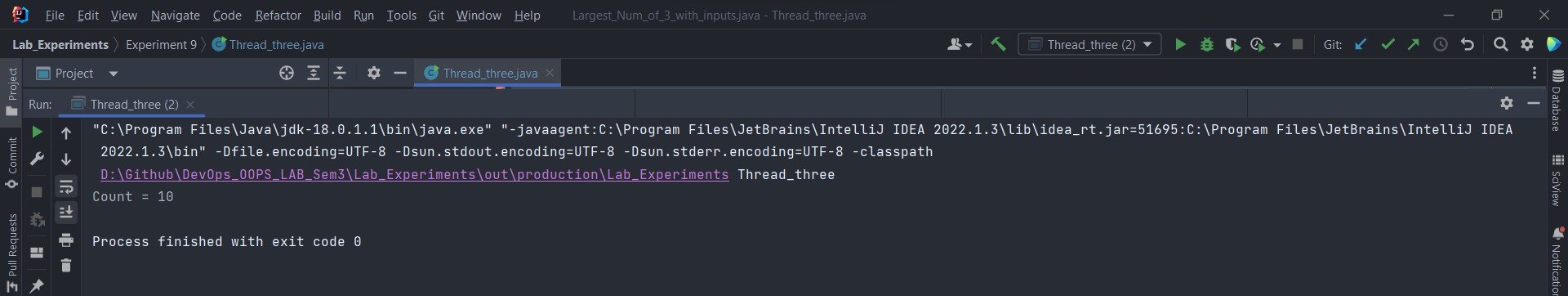


1. Write a program to launch 10 threads. Each thread increments a counter variable. Run the program with synchronization.

Code:

class Counter1{  
 static int count = 0;  
 public void incr(){  
 count++;  
 }  
}  
public class Thread\_three {  
 public static void main(String[] args)throws InterruptedException {  
  
 Counter1 c = new Counter1();  
 Thread t1 = new Thread(new Runnable(){  
 public void run(){  
 for(int i=0; i<10; i++)  
 c.incr();  
 }  
 });  
 t1.start();  
 t1.join();  
 System.out.println("Count = " + Counter1.count);  
 }  
}

Output:

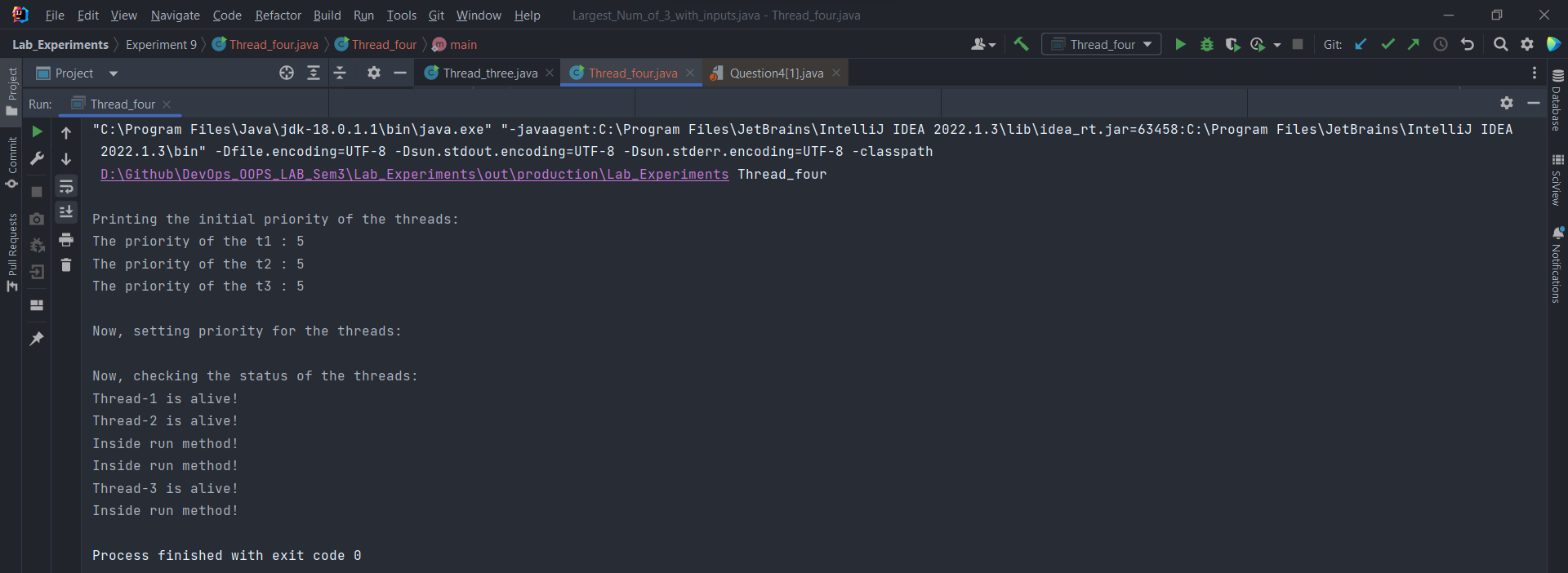


1. Write a Java program to create five threads with different priorities. Send two threads of the highest priority to sleep state. Check the aliveness of the threads and mark which thread is long lasting

Code:

class MyThread extends Thread  
{  
 public void run()  
 {  
 System.out.println("Inside run method!");  
 }  
}  
  
public class Thread\_four {  
 public static void main(String[] args) throws InterruptedException {  
 MyThread t1 = new MyThread();  
 MyThread t2 = new MyThread();  
 MyThread t3 = new MyThread();  
 System.out.println("\nPrinting the initial priority of the threads:");  
 System.out.println("The priority of the t1 : " + t1.getPriority());  
 System.out.println("The priority of the t2 : " + t2.getPriority());  
 System.out.println("The priority of the t3 : " + t3.getPriority());  
  
 System.out.println("\nNow, setting priority for the threads:");  
 t1.setPriority(6);  
 t2.setPriority(3);  
 t3.setPriority(10);  
  
 System.out.println("\nNow, checking the status of the threads:");  
 t1.start();  
 if(t1.isAlive()) {  
 System.out.println("Thread-1 is alive!");  
 } else {  
 System.out.println("Thread-1 is not alive!");  
 }  
 t2.start();  
 if(t2.isAlive()) {  
 System.out.println("Thread-2 is alive!");  
 } else {  
 System.out.println("Thread-2 is not alive!");  
 }  
 Thread.sleep(1000);  
 t3.start();  
 if(t3.isAlive()) {  
 System.out.println("Thread-3 is alive!");  
 } else {  
 System.out.println("Thread-3 is not alive!");  
 }  
 }  
}

Output:



Title: Collections

1. Write a program for the following:

Read all elements from ArrayList by using Iterator.

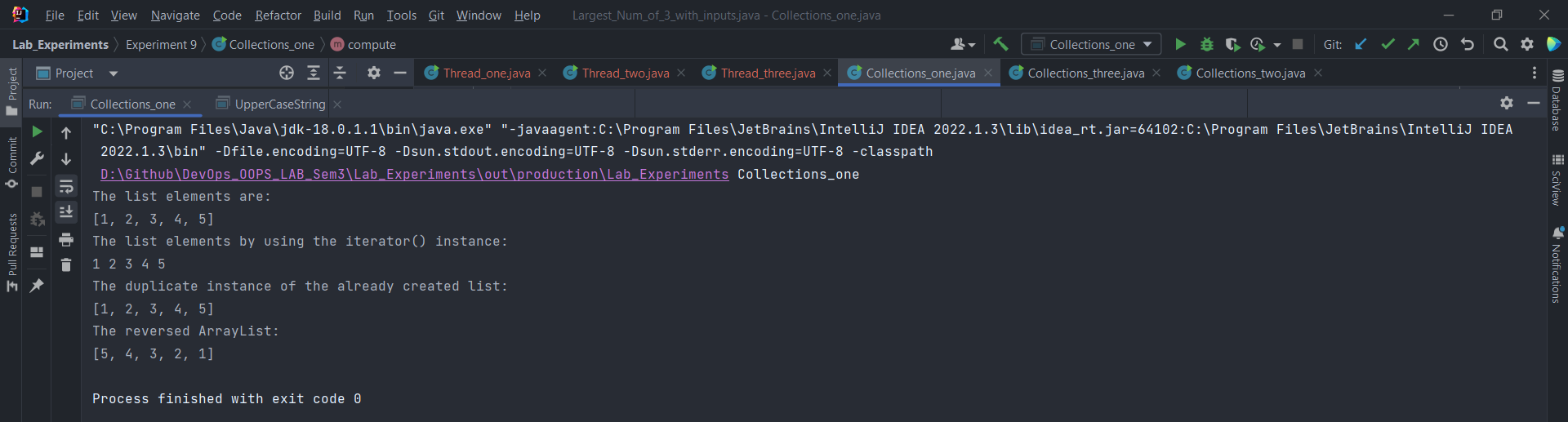
Create duplicate object of an ArrayList instance.

Reverse ArrayList content.

Code:

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Iterator;  
  
public class Collections\_one {  
 public void compute() {  
 ArrayList<Integer> list = new ArrayList<>();  
// adding elements into the list  
 list.add(1);  
 list.add(2);  
 list.add(3);  
 list.add(4);  
 list.add(5);  
// Printing the list  
 System.out.println("The list elements are:");  
 System.out.println(list);  
// Creating iterator instance of the list  
 Iterator<Integer> iter = list.iterator();  
// now displaying the list elements by using iterator  
 System.out.println("The list elements by using the iterator() instance:");  
 while (iter.hasNext()) {  
 System.out.print(iter.next() +" ");  
 }  
  
 ArrayList<?> duplicate = new ArrayList<>();  
 duplicate = (ArrayList<?>)list.clone();  
  
 System.out.println("\nThe duplicate instance of the already created list:");  
 // printing the cloned list  
 System.out.println(duplicate);  
  
// reversing the cloned ArrayList  
 Collections.reverse(duplicate);  
 System.out.println("The reversed ArrayList:");  
// printing the reversed ArrayList  
 System.out.println(duplicate);  
 }  
 public static void main(String[] args) {  
 Collections\_one obx = new Collections\_one();  
 obx.compute();  
 }  
}

Output:



1. Write a program for the following HashMap:

find whether specified key exists or not.

Find whether specified value exists or not

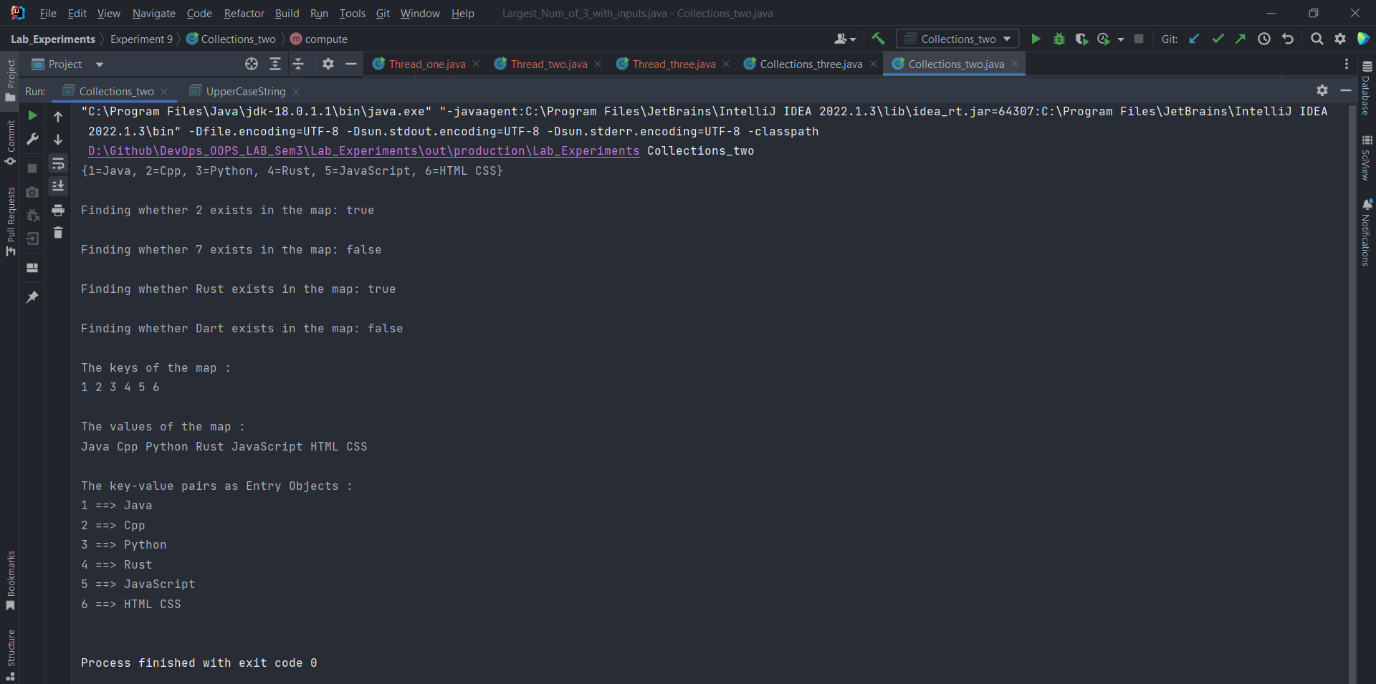
get all keys from the given HashMap

get all key-value pair as Entry objects.

Code:

import java.util.\*;  
  
public class Collections\_two {  
 public void compute() {  
 HashMap<Integer,String> map = new HashMap<>();  
 map.put(1,"Java");  
 map.put(2,"Cpp");  
 map.put(3,"Python");  
 map.put(4,"Rust");  
 map.put(5,"JavaScript");  
 map.put(6,"HTML CSS");  
// printing the entries of the HashMap  
 System.out.println(map);  
// finding whether this Key is present in the HashMap  
 System.out.println("\nFinding whether 2 exists in the map: " + map.containsKey(2));  
 System.out.println("\nFinding whether 7 exists in the map: " + map.containsKey(7));  
// finding whether this Value is present in the HashMap  
 System.out.println("\nFinding whether Rust exists in the map: " + map.containsValue("Rust"));  
 System.out.println("\nFinding whether Dart exists in the map: " + map.containsValue("Dart"));  
  
// printing the keys of the map  
 System.out.println("\nThe keys of the map : ");  
 Set<Integer> keys = map.keySet();  
 for (Integer key : keys) {  
 System.out.print(key + " ");  
 }  
 System.out.println();  
  
// printing the values of the map  
 System.out.println("\nThe values of the map :");  
 Collection<String> values = map.values();  
 for(String value : values) {  
 System.out.print(value + " ");  
 }  
 System.out.println();  
  
// printing the key-value pairs of the map  
 System.out.println("\nThe key-value pairs as Entry Objects :");  
 Set<Map.Entry<Integer,String>> entries = map.entrySet();  
 for (Map.Entry<Integer,String> ent : entries) {  
 System.out.println(ent.getKey() + " ==> " + ent.getValue());  
 }  
 System.out.println();  
 }  
 public static void main(String[] args) {  
 Collections\_two obx = new Collections\_two();  
 obx.compute();  
 }  
}

Output:



1. Write a program for the following HashSet:

copy another collection object to HashSet object.

Delete all entries at one call from HashSet

search user defined objects from HashSet

Code:

import java.util.HashSet;  
  
class Custom\_Class {  
 Custom\_Class() {  
 }  
}  
  
public class Collections\_three {  
 public void compute() {  
 HashSet<Integer> set = new HashSet<>();  
 set.add(1);  
 set.add(2);  
 set.add(3);  
 HashSet<Integer> set2 = new HashSet<>();  
 set2.add(4);  
 set2.add(5);  
 System.out.println("Copying another HashSet object to HashSet object");  
// copying another HashSet object to HashSet object  
 set.addAll(set2);  
// printing the set  
 System.out.println(set);  
  
 System.out.println("\nDeleting all entries from the HashSet object at once");  
// deleting all entries from the HashSet object at once  
 set2.clear();  
 System.out.println("set2 after clearing all entries:" + set2);  
  
  
 System.out.println("\nSearch for user defined objects in hashset");  
// creating a HashSet for Random objects  
 HashSet<Custom\_Class> hashSet = new HashSet<>();  
 Custom\_Class ob1 = new Custom\_Class();  
 Custom\_Class ob2 = new Custom\_Class();  
 Custom\_Class ob3 = new Custom\_Class();  
 hashSet.add(ob1);  
 hashSet.add(ob2);  
  
// search for user defined objects in hashset  
 if (hashSet.contains(ob1)){  
 System.out.println("Ob1 is present in the hashSet");  
 } else {  
 System.out.println("Ob1 is not present in the hashSet");  
 }  
   
 if (hashSet.contains(ob3)) {  
 System.out.println("Ob3 is present in the hashSet");  
 } else {  
 System.out.println("Ob3 is not present in the hashSet");  
 }  
 }  
 public static void main(String[] args) {  
 Collections\_three obx = new Collections\_three();  
 obx.compute();  
 }  
}

Output:

