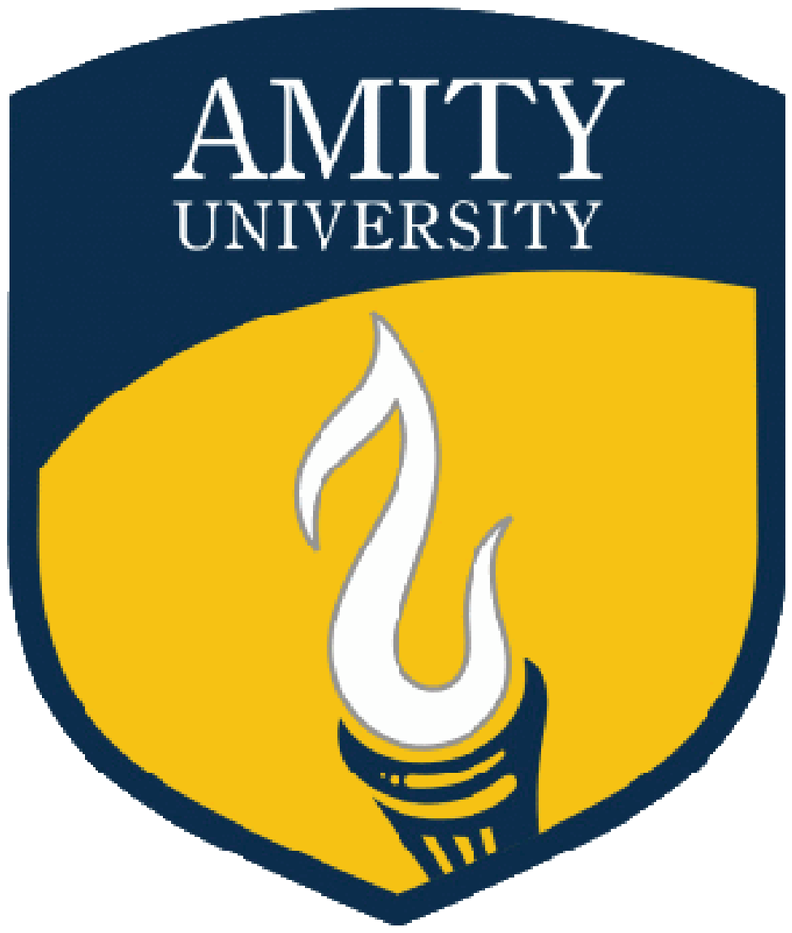
AMITY INSTITUTE OF INFORMATION TECHNOLOGY



Batch Year 2021-24

COURSE: B.SC (IT)

PRACTICAL FILE OF JAVA PROGRAMMING

Submitted To:

Dr Surjeet Dalal

Submitted By:

Rahul Raj Singh

A50504921014

5th Semester

PRACTICAL NO: -1

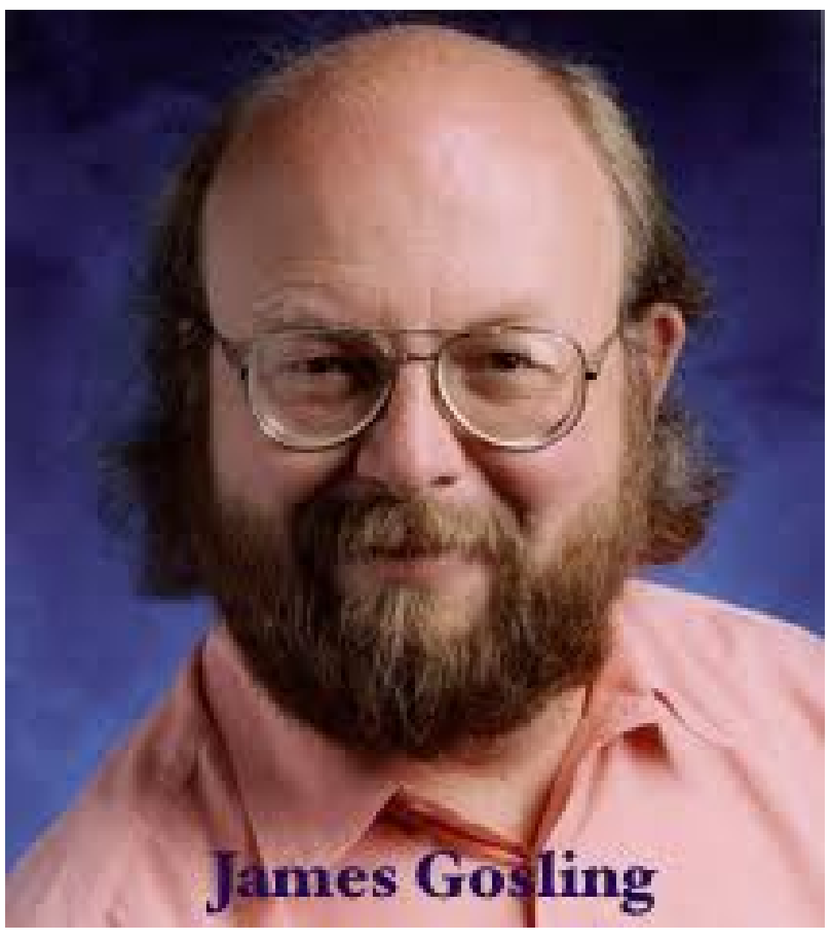
AIM: -

To Study the History of JAVA PROGRAMMING

JAVA PROGRAMMING

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was best suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.



Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. Following are given significant points that describe the history of Java.

1) **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

2) Initially it was designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called "**Greentalk**" by James Gosling, and the file extension was .gt.

4) After that, it was called **Oak** and was developed as a part of the Green project.

5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

6) In 1995, Oak was renamed as "**Java**" because it was already a trademark by Oak Technologies.

PRACTICAL NO: -2

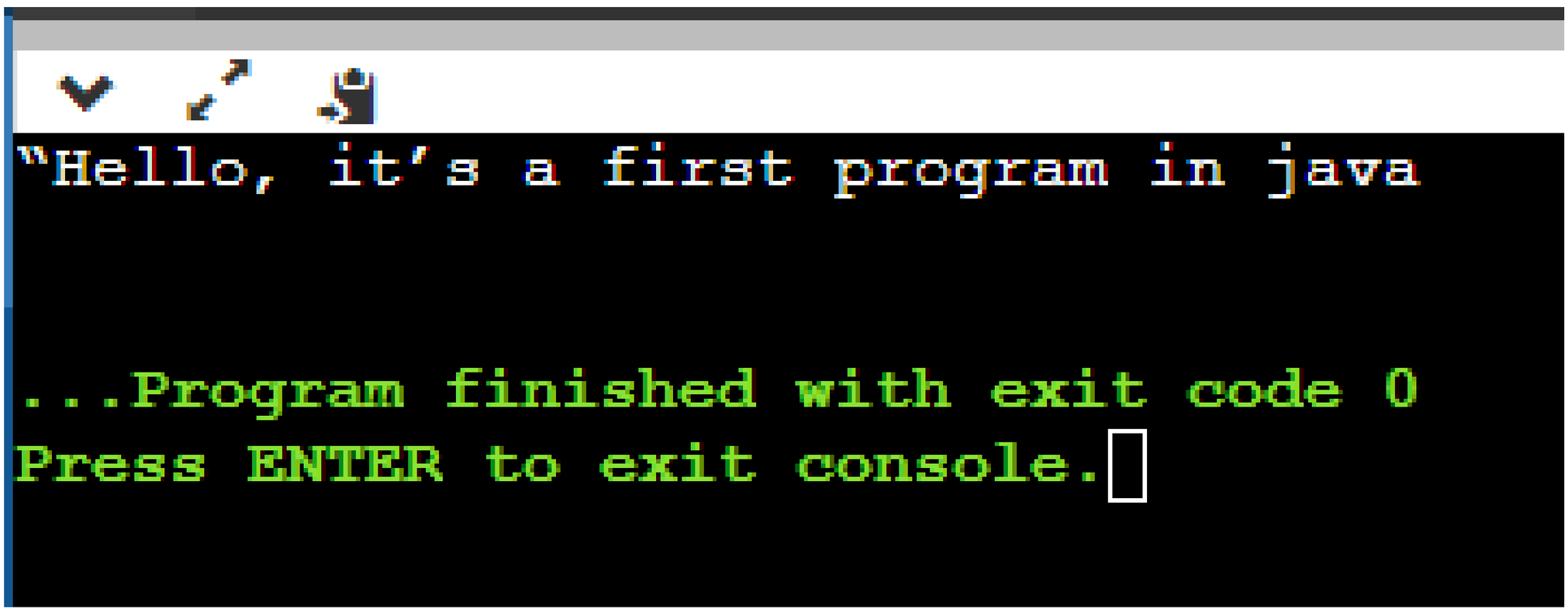
AIM: -

WAP to display “ Hello, it’ s a first program in java” .

CODE: -

```
public class Main
{
        public static void main(String[] args) {
                System.out.println(""“ Hello, it’ s a first program in java ");
        }
}
```

OUTPUT: -



PRACTICAL NO: -3

AIM: -

WAP to find sum of two integers taken as input from user at runtime.

CODE: -

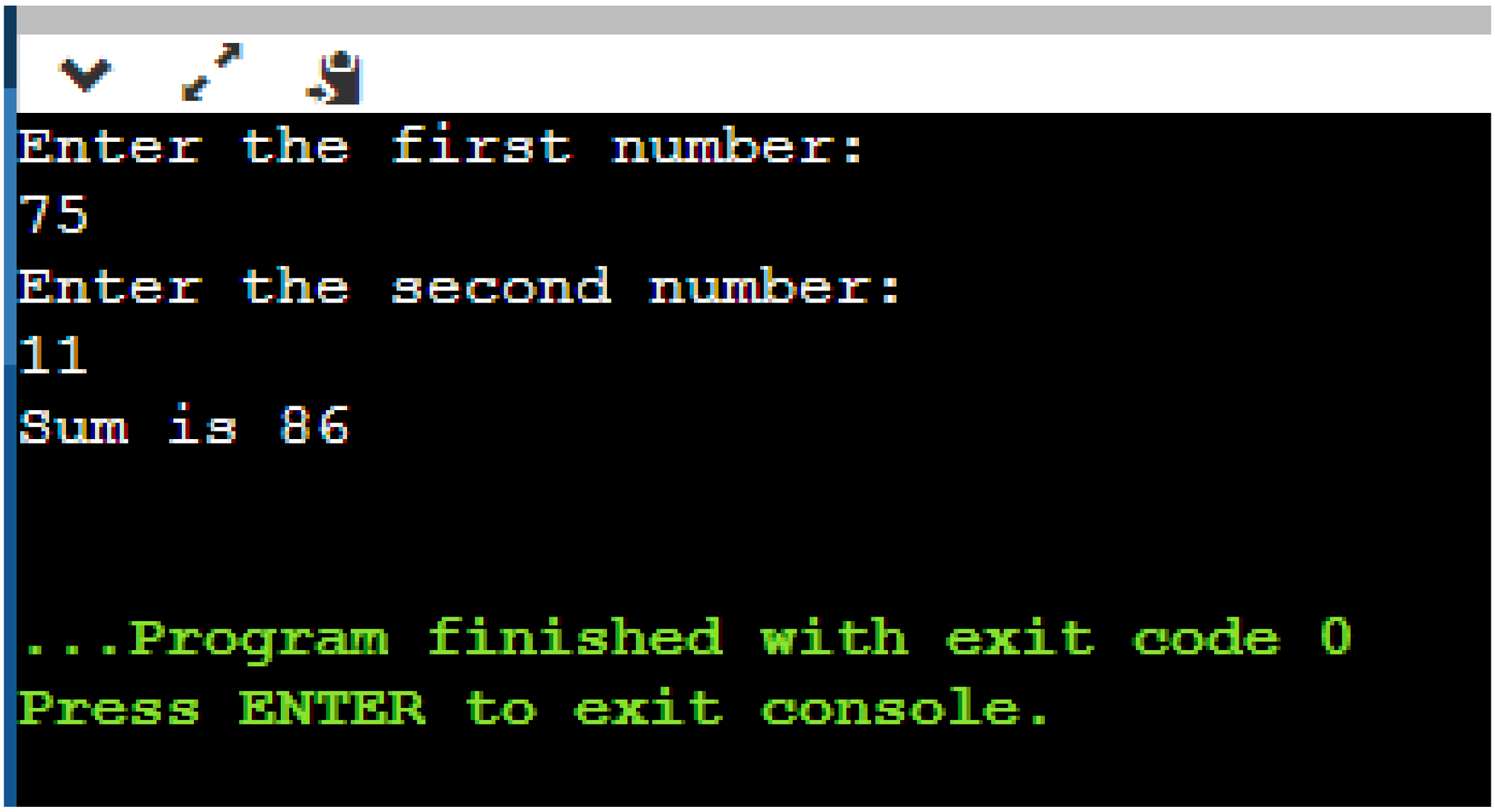
```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner scn=new Scanner(System.in);

        System.out.println("Enter the first number:");
        int x = scn.nextInt();

        System.out.println("Enter the second number:");
        int y = scn.nextInt();

        int sum = x + y;
        System.out.println("Sum is " + sum);
    }
}
```

OUTPUT: -



AIM: -

WAP to find sum of two float numbers taken as command line arguments.

CODE: -

```
import java.util.Scanner;
public class AddTwoNumbers2 {
    public static void main(String[] args) {
        float num1, num2, sum;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter First Number: ");
        num1 = sc.nextFloat();

        System.out.println("Enter Second Number: ");
        num2 = sc.nextFloat();

        sc.close();
        sum = num1 + num2;
```

```
    System.out.println("Sum of these numbers: "+sum);
  }}
```

OUTPUT: -

```
Output

java -cp /tmp/utYXCCRB3K AddTwoNumbers2
Enter First Number: 74
Enter Second Number:
52
Sum of these numbers: 126.0
```

PRACTICAL NO: -5

AIM: -

WAP to find changed case of entered character.

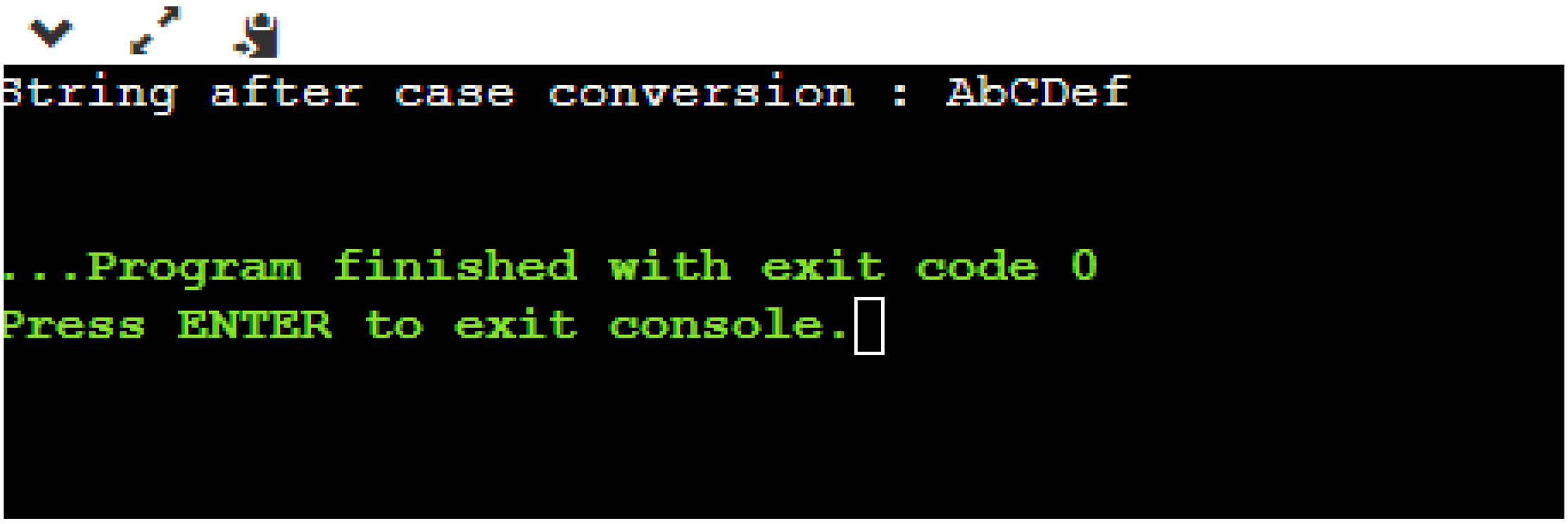
CODE: -

```
public class Main {
   public static void main(String[] args) {
      String str1="aBcdEF";
      StringBuffer newStr=new StringBuffer(str1);
      for(int i = 0; i < str1.length(); i++) {
         if(Character.isLowerCase(str1.charAt(i)
            newStr.setCharAt(i, Character.toUpperCase(str1.charAt(i)));
         }
```

```
        else if(Character.isUpperCase(str1.charAt(i))) {

            newStr.setCharAt(i, Character.toLowerCase(str1.charAt(i)));
        }
      }
      System.out.println("String after case conversion : " + newStr);
   }
}
```

OUTPUT: -



PRACTICAL NO: -6

AIM: -

WAP to find maximum of 3 integer numbers taken as input from user at runtime.

CODE: -

```
import java.util.Scanner;
public class Main {
public static void main(String[] args)   {
int a, b, c, largest, temp;
```

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter the first number:");
a = sc.nextInt();
System.out.println("Enter the second number:");
b = sc.nextInt();
System.out.println("Enter the third number:");
c = sc.nextInt();
temp=a>b?a:b;
largest=c>temp?c:temp;
System.out.println("The largest number is: "+largest);
}
}
```

OUTPUT: -

```
Enter the first number:
50
Enter the second number:
69
Enter the third number:
10
The largest number is: 69


...Program finished with exit code 0
Press ENTER to exit console.
```

PRACTICAL NO: -7

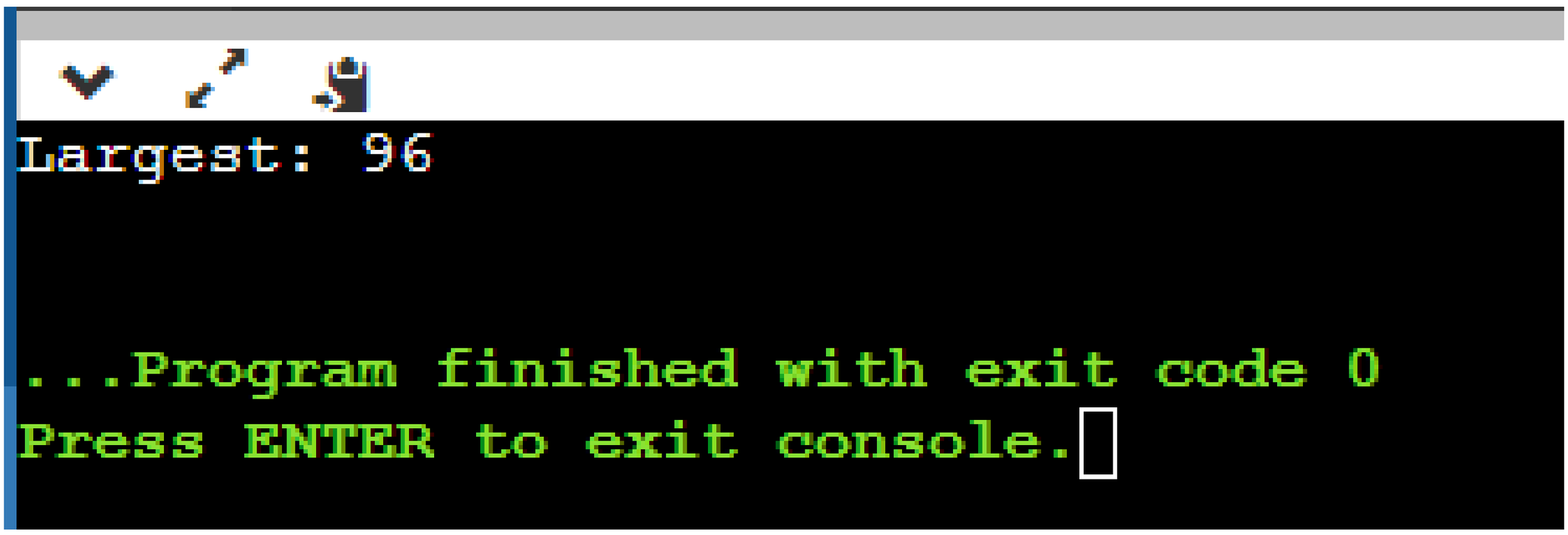
AIM: -

WAP in java to find out the greatest out of ten numbers stored using arrays.

CODE: -

```
public class Main{
public static int getLargest(int[] a, int total){
int temp;
for (int i = 0; i < total; i++)
        {
            for (int j = i + 1; j < total; j++)
            {
                if (a[i] > a[j])
                {
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                }
            }
        }
        return a[total-1];  }
public static void main(String args[]){
int a[]={1,2,5,6,3,2,11,74,96,85};
System.out.println("Largest: "+getLargest(a,10));
}}
```

OUTPUT: -



PRACTICAL NO: -8

AIM: -

WAP to create class with “ name” as String and “ age” as integer data members. The class should have two methods to take input from user and display the data.

CODE: -

```
import java.util.Scanner;
class Person {
    String name;
    int age;
    void inputDetails() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter name: ");
        name = scanner.nextLine();
        System.out.print("Enter age: ");
        age = scanner.nextInt();
    }   void displayDetails() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }  public static void main(String[] args) {
        Person person = new Person();
        person.inputDetails();
        person.displayDetails();
    }}
```

OUTPUT: -

Output

```
java -cp /tmp/lq6zpQPCHD Person
Enter name: Rahul
Enter age: 20
Name: RahulAge: 20
```

PRACTICAL NO: -9

AIM: -

WAP to find factorial of a number using class and object.

CODE: -

```
class FactorialExample{
 public static void main(String args[]){
  int i,fact=1;
  int number=7;
  for(i=1;i<=number;i++){
      fact=fact*i;
  }
  System.out.println("Factorial of "+number+" is: "+fact);
 }
}
```

OUTPUT: -

Output

```
java -cp /tmp/h7sstbfZ4Y FactorialExample
Factorial of 7 is: 5040
```

PRACTICAL NO: -10

AIM: -

WAP in java to illustrate the concept of interfaces.

CODE: -

```
interface Polygon {
  void getArea(int length, int breadth);
}
class Rectangle implements Polygon {
  public void getArea(int length, int breadth) {
    System.out.println("The area of the rectangle is " + (length * breadth));
  }
}
class Main {
  public static void main(String[] args) {
```

```
    Rectangle r1 = new Rectangle();
    r1.getArea(5, 6);
  }
}
```

OUTPUT: -

```
Output

java -cp /tmp/oaOgONMTme Main
The area of the rectangle is 30
```

PRACTICAL NO: -11

AIM: -

WAP to create a package as MyPack having a class with three methods: max, fact and show. Use it in other folder with setting classpath and without setting class path.

CODE: -

```
package MyPack;

public class MyPackClass {
    public int max(int a, int b) {
        return (a > b) ? a : b;
    }
```

```
    public int fact(int n) {
        if (n == 0 || n == 1) {
            return 1;
        } else {
            return n * fact(n - 1);
        }
    }
    public void show() {
        System.out.println("Inside the show method of MyPackClass");
import MyPack.MyPackClass;

public class MainClass {
    public static void main(String[] args) {
        MyPackClass myPackObj = new MyPackClass();

        int resultMax = myPackObj.max(5, 8);
        System.out.println("Max value: " + resultMax);

        int resultFact = myPackObj.fact(5);
        System.out.println("Factorial: " + resultFact);

        myPackObj.show();
    }
}
```

OUTPUT: -

```
Max value: 8
Factorial: 120
Inside the show method of MyPackClass
```

PRACTICAL NO: -12

AIM: -

WAP in java to showcase uses of super keyword.

CODE: -

```
class Superclass
{
  int num = 100;
```

```
}
class Subclass extends Superclass
{
    int num = 110;
    void printNumber(){
        System.out.println(num);
    }
    public static void main(String args[]){
        Subclass obj= new Subclass();
        obj.printNumber();
    }
}
```

OUTPUT: -

Output

```
java -cp /tmp/VtTz7f232i Subclass
110
```

AIM: -

WAP to demonstrate the use of nesting of try-catch block.

CODE: -

```
class Nest{
  public static void main(String args[]){
    try{
       try{
          System.out.println("Inside block1");
          int b =45/0;
          System.out.println(b);
       }
       catch(ArithmeticException e1){
          System.out.println("Exception: e1");
       }
       try{
          System.out.println("Inside block2");
          int b =45/0;
          System.out.println(b);
       }
       catch(ArrayIndexOutOfBoundsException e2){
          System.out.println("Exception: e2");
       }
      System.out.println("Just other statement");
    }
    catch(ArithmeticException e3){
           System.out.println("Arithmetic Exception");
        System.out.println("Inside parent try catch block");
    }
    catch(ArrayIndexOutOfBoundsException e4){
           System.out.println("ArrayIndexOutOfBoundsException");
        System.out.println("Inside parent try catch block");
    }
    catch(Exception e5){
           System.out.println("Exception");
        System.out.println("Inside parent try catch block");
    }
    System.out.println("Next statement..");
  }}
```

OUTPUT: -

Output

```
java -cp /tmp/VtTz7f232i Nest
Inside block1
Exception: e1
Inside block2
Arithmetic Exception
Inside parent try catch block
Next statement..
```

PRACTICAL NO: -14

AIM: -

WAP in java to illustrate the concept of using multiple catch clauses to handle different types of exceptions.

CODE: -

```
public class MultipleCatchBlock1 {

    public static void main(String[] args) {

        try{
            int a[]=new int[5];
            a[5]=30/0;
           }
           catch(ArithmeticException e)
             {
              System.out.println("Arithmetic Exception occurs");
             }
           catch(ArrayIndexOutOfBoundsException e)
             {
              System.out.println("ArrayIndexOutOfBounds Exception occurs");
             }
           catch(Exception e)
             {
              System.out.println("Parent Exception occurs");
             }
           System.out.println("rest of the code");
    }
}
```

OUTPUT: -

Output

```
java -cp /tmp/VtTz7f232i MultipleCatchBlock1
Arithmetic Exception occurs
rest of the code
```

PRACTICAL NO: -15

AIM: -

WAP in java to create a user defined Exception and throw it explicitly.

CODE: -

```
class SampleException{
public static void main(String args[]){
try{
throw new UserException(400);
}
catch(UserException e){
System.out.println(e) ;
}
}
}
class UserException extends Exception{
int num1;
UserException(int num2) {
num1=num2;
}
public String toString(){
return ("Status code = "+num1) ;
}
}
```

OUTPUT: -

Output

```
java -cp /tmp/VtTz7f232i SampleException
Status code = 400
```

PRACTICAL NO: -16

AIM: -

Demonstrate thread using Thread class and Runnable interface.

CODE: -

Java Thread by extending Thread class:-

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

```
Output

java -cp /tmp/VtTz7f232i Multi
thread is running...
```

Java Thread by implementing Runnable interface:-

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

```
Output

java -cp /tmp/VtTz7f232i Multi
thread is running...
```

PRACTICAL NO: -17

AIM: -

Demonstrate various thread methods using a program.

CODE: -

```
public class AnimalMultiThreadDemo {
        public static void main(String[] args) throws Exception{
                // Make object of  Runnable
                AnimalRunnable anr = new AnimalRunnable();
                Thread cat = new Thread(anr);
                cat.setName("Cat");
```

```
                Thread dog = new Thread(anr);
                dog.setName("Dog");
                Thread cow = new Thread(anr);
                cow.setName("Cow");
                System.out.println("Thread State of Cat before calling start:
"+cat.getState());
                cat.start();
                dog.start();
                cow.start();
                System.out.println("Thread State of Cat in Main method before Sleep: " +
cat.getState());
                System.out.println("Thread State of Dog in Main method before Sleep: " +
dog.getState());
                System.out.println("Thread State of Cow in Main method before Sleep: " +
cow.getState());
                Thread.sleep(10000);
                System.out.println("Thread State of Cat in Main method after sleep: " +
cat.getState());
                System.out.println("Thread State of Dog in Main method after sleep: " +
dog.getState());
                System.out.println("Thread State of Cow in Main method after sleep: " +
cow.getState());
        }
}class AnimalRunnable implements Runnable {
        @Override
        public void run() {
                for (int x = 1; x < 4; x++) {
                        System.out.println("Run by " + Thread.currentThread().getName());
                        try {
                                Thread.sleep(1000);
                        } catch (InterruptedException ex) {
                                ex.printStackTrace();
                        }
                }
                System.out.println("Thread State of: "+ Thread.currentThread().getName()+ "
- "+Thread.currentThread().getState());
                System.out.println("Exit of Thread: "
                                + Thread.currentThread().getName());}}
```

OUTPUT: -

Output

```
java -cp /tmp/VtTz7f232i AnimalMultiThreadDemo
Thread State of Cat before calling start: NEW
Thread State of Cat in Main method before Sleep: RUNNABLE
Run by CowThread State of Dog in Main method before Sleep: RUNNABLE
Run by Cat
Run by Dog
Thread State of Cow in Main method before Sleep: RUNNABLE
Run by Cow
Run by Cat
Run by Dog
Run by Cow
Run by CatRun by Dog
Thread State of: Dog - RUNNABLE
Thread State of: Cow - RUNNABLE
Thread State of: Cat - RUNNABLE
Exit of Thread: Dog
Exit of Thread: Cow
Exit of Thread: Cat
Thread State of Cat in Main method after sleep: TERMINATED
Thread State of Dog in Main method after sleep: TERMINATED
Thread State of Cow in Main method after sleep: TERMINATED
```