Declaration

Mahendra Singh Mahara
BCA-III Semester

I, Mahendra Singh Mahara, hereby declare that the lab report titled "Java-Lab Report" is my original work and has been prepared in fulfillment of the requirements of the academic course "OOP in JAVA" (BCA third semester) at Triton International College, as outlined in the course curriculum. This report is based on practical knowledge gained from the "Object Oriented Programming in Java" textbook prescribed by Tribhuvan University.

I further declare that:

1. All the data and information used in this report are derived from authentic sources and are accurately represented.
2. Any external sources of information, such as books, research papers, or websites, have been properly cited and referenced in accordance with the academic guidelines provided by the subject teacher.
3. The VScode (Visual Studio Code) software was used for practical work, and the results presented in this report are a true representation of the statistical analysis conducted.
4. I have not engaged in any form of academic dishonesty, including plagiarism, unauthorized collaboration, or any other activities that would compromise the integrity of this report.

I understand that academic honesty and integrity are of utmost importance, and I take full responsibility for the content of this report. Any violation of academic ethics would be subject to appropriate disciplinary action.

I hereby affix my signature to this declaration.

Signature: [illegible]
Date: 1 September 2023
Thank You

INTRODUCTION:

Java, a versatile and widely used programming language, is celebrated for its platform independence, object-oriented approach, robustness, and security features. It was developed by Sun Microsystems, which is now under the ownership of Oracle Corporation. These distinct features contribute to Java's popularity and make it suitable for a wide range of applications.

Java's platform independence is one of its standout features. Unlike many other programming languages, Java is compiled into an intermediate form called bytecode. This bytecode can run on any platform equipped with a Java Virtual Machine (JVM). The JVM serves as an intermediary between the bytecode and the native machine code of the host system. This portability is a significant advantage for developers, as it allows them to write code once and run it anywhere, reducing compatibility issues.

Another core feature of Java is its adherence to object-oriented principles. Java encourages the use of classes and objects, fostering modularity and code reusability. This object-oriented approach simplifies software design, maintenance, and testing.

Java's robustness is evident through its strong type-checking and comprehensive exception handling mechanisms. This contributes to the development of reliable and secure applications, which is especially crucial for mission-critical systems and web applications.

Furthermore, Java supports multithreading, allowing multiple threads to execute concurrently. This capability enhances application responsiveness and performance, making Java suitable for developing complex, high-performance software.

Java's built-in security features, including class loaders and bytecode verification, protect against malicious code execution, making it a preferred choice for web applications and services that require stringent security measures.

In summary, Java's platform independence, object-oriented nature, robustness, multithreading support, and security features make it a compelling choice for a wide range of software development projects.

Define the terms: JDK, JRE, JVM, Byte Code, Token, Keyword, Literal, Operator

- JDK (Java Development Kit): The JDK is a comprehensive software package that includes everything necessary for developing, compiling, and running Java applications. It contains the Java compiler (javac), libraries, documentation, and tools like the debugger and profiler.

- JRE (Java Runtime Environment): The JRE is a subset of the JDK. It consists of the essential runtime components required to execute Java applications, including the Java Virtual Machine (JVM) and the Java class libraries.

- JVM (Java Virtual Machine): The JVM is a crucial component of the Java runtime environment. It interprets and executes Java bytecode, which is a platform-independent intermediate representation of Java source code. The JVM converts bytecode into native machine code, allowing Java applications to run on various platforms.

- Bytecode: Bytecode is an intermediary representation of Java source code. When you compile a Java program, the compiler generates bytecode instead of native machine code. Bytecode is executed by the JVM, making Java platform-independent. It ensures that Java programs can run on any system with a compatible JVM.

- Token : In Java, a token is the smallest individual unit in a program. Tokens are the building blocks of Java code and include keywords, identifiers, literals, operators, and punctuation symbols.

- Keyword : Keywords are reserved words in Java that have predefined meanings and cannot be used as identifiers. Examples of Java keywords include "public," "class," "if," "while," and "return."

- Literal : Literals are constant values used directly in Java code. They can be of various types, such as numeric literals (e.g., 42, 3.14), character literals (e.g., 'A', '\n'), string literals (e.g., "Hello, World!"), and boolean literals (e.g., true, false).

- Operator : Operators are symbols or special characters that perform operations on operands. Java supports a variety of operators, including arithmetic operators (e.g., +, -, *, /), relational operators (e.g., ==, !=, <, >), logical operators (e.g., &&, ||, !), and assignment operators (e.g., =, +=, -=).

Understanding these fundamental terms is crucial for grasping the basics of Java programming, as they form the foundation of the language's syntax and semantics. They enable developers to construct meaningful Java programs by combining these elements effectively.

Write a Java Program to Demonstrate Different Operators

```
public class OperatorDemo {
    public static void main(String[] args) {
        // Arithmetic Operators
        int a = 10, b = 5;
        int sum = a + b;           // Addition
        int difference = a - b;    // Subtraction
        int product = a * b;       // Multiplication
        int quotient = a / b;      // Division
```

```java
        int remainder = a % b;      // Modulus

        // Relational Operators
        boolean isEqual = (a == b);         // Equal to
        boolean isNotEqual = (a != b);      // Not equal to
        boolean isGreaterThan = (a > b);   // Greater than
        boolean isLessThan = (a < b);      // Less than

        // Logical Operators
        boolean andResult = (true && false);    // Logical AND
        boolean orResult = (true || false);      // Logical OR
        boolean notResult = !true;              // Logical NOT

        // Conditional Structure (if-else)
        if (a > b) {
            System.out.println("a is greater than b");
        } else {
            System.out.println("b is greater than or equal to a");
        }
    }
}
```

1)Write a program to calculate salary of n employees using concept of classes with constructors and methods.

```
import java.lang.*;
import java.io.*;
class Employee
{
intemp_id;
String emp_name;
float basic_salary;
Employee(int id, String name, float sal)
{
        emp_id=id;
        emp_name=name;
        basic_salary=sal;
}
void display()
{
        float da=basic_salary*15/100;
        float hra=basic_salary*10/100;
        float gross_sal=basic_salary+da+hra;
        System.out.println ("Employee Id= "+emp_id);
        System.out.println ("Emplyee Name= "+emp_name);
        System.out.println ("Hra= "+hra);
        System.out.println ("DA= "+da);
        System.out.println ("Gross Salary= "+gross_sal);
}
}
class Employeedemo
{
public static void main(String args[]) throws IOException
{
        BufferedReaderbr = new BufferedReader(new InputStreamReader(System.in));
        System.out.println ("Enter number of Employees");
        int n = Integer.parseInt(br.readLine());
        for(inti=0;i<n;i++){
        System.out.println ("\n\n\nEnter Employee id");
        int id = Integer.parseInt(br.readLine());
        System.out.println ("Enter Employee Name");
        String name = br.readLine();
        System.out.println ("Enter Basic Salary");
        Float sal = Float.parseFloat(br.readLine());
        Employee e = new Employee(id, name, sal);
        e.display();
        }
}
}
```

2) Write a program to demonstrate various arithmetic calculations using packages.

```
package arithmetic;
public class MyMath
{
            public intadd(intx,int y)
            {
            System.out.println ("\t"+(x+y));
            }
            public intsub(intx,int y)
            {
            System.out.println ("\t"+(x-y));
            }
public intmul(intx,int y)
            {
            System.out.println ("\t"+(x*y));
            }
            public double div(intx,int y)
            {
            System.out.println ("\t"+(x/y));
            }
            public intmod(intx,int y)
            {
            System.out.println ("\t"+(x%y));
            }
}
-----------------------------------------------------------------------------------------------------------------------
import arithmetic.*;
class Test
{
            public static void main(String as[])
            {
            MyMath m=new MyMath();
            m.add(13.42);
            m.sub(16,5);
            m.mul(8,4);
            m.div(24,9);
            m.mod(19,6);
            }
}
```

3) Write a program to demonstrate client-server environment using multithreading.

```
import java.net.*;
import java.io.*;

public class GreetingClient {

   public static void main(String [] args) {
      String serverName = args[0];
int port = Integer.parseInt(args[1]);
      try {
System.out.println("Connecting to " + serverName + " on port " + port);
         Socket client = new Socket(serverName, port);

System.out.println("Just connected to " + client.getRemoteSocketAddress());
OutputStreamoutToServer = client.getOutputStream();
DataOutputStream out = new DataOutputStream(outToServer);

out.writeUTF("Hello from " + client.getLocalSocketAddress());
InputStreaminFromServer = client.getInputStream();
DataInputStream in = new DataInputStream(inFromServer);

System.out.println("Server says " + in.readUTF());
client.close();
      } catch (IOException e) {
e.printStackTrace();
      }
   }
}
-------------------------------------------------------------------

// File Name GreetingServer.java
import java.net.*;
import java.io.*;

public class GreetingServer extends Thread {
   private ServerSocketserverSocket;

   public GreetingServer(int port) throws IOException {
serverSocket = new ServerSocket(port);
serverSocket.setSoTimeout(10000);
   }

   public void run() {
      while(true) {
         try {
System.out.println("Waiting for client on port " +
serverSocket.getLocalPort() + "...");
            Socket server = serverSocket.accept();

System.out.println("Just connected to " + server.getRemoteSocketAddress());
DataInputStream in = new DataInputStream(server.getInputStream());
System.out.println(in.readUTF());
DataOutputStream out = new DataOutputStream(server.getOutputStream());
out.writeUTF("Thank you for connecting to " + server.getLocalSocketAddress()
          + "\nGoodbye!");
server.close();

      } catch (SocketTimeoutException s) {
System.out.println("Socket timed out!");
         break;
      } catch (IOException e) {
e.printStackTrace();
         break;
      }
    }
  }

  public static void main(String [] args) {
int port = Integer.parseInt(args[0]);
    try {
      Thread t = new GreetingServer(port);
t.start();
    } catch (IOException e) {
e.printStackTrace();
    }
  }
}
```

4) Write a program to demonstrate mutual exclusion using thread synchronization.

```
class PrintDemo {
  public void printCount() {
    try {
for(inti = 5; i> 0; i--) {
System.out.println("Counter   ---   "  +i );
      }
    } catch (Exception e) {
System.out.println("Thread  interrupted.");
    }
  }
}
class ThreadDemo extends Thread {
  private Thread t;
  private String threadName;
PrintDemo  PD;
ThreadDemo( String name,  PrintDemopd) {
threadName = name;
    PD = pd;
  }
  public void run() {
PD.printCount();
System.out.println("Thread " +  threadName + " exiting.");
  }
  public void start () {
System.out.println("Starting " +  threadName );
    if (t == null) {
      t = new Thread (this, threadName);
t.start ();
    }
  }
}
public class TestThread {
  public static void main(String args[]) {
PrintDemo PD = new PrintDemo();
ThreadDemo T1 = new ThreadDemo( "Thread - 1 ", PD );
ThreadDemo T2 = new ThreadDemo( "Thread - 2 ", PD );
    T1.start();
    T2.start();
    // wait for threads to end
      try {
      T1.join();
      T2.join();
    } catch ( Exception e) {
System.out.println("Interrupted");
    }
  }
}
```

5) Write a program to demonstrate Linked list class.

```
import java.util.*;
class LinkedListDemo {
public static void main(String args[]) {
LinkedListll = new LinkedList();
ll.add("F");
ll.add("B");
ll.add("D");
ll.add("E");
ll.add("C");
ll.addLast("Z");
ll.addFirst("A");
ll.add(1, "A2");
System.out.println("Original contents of ll: " + ll);
ll.remove("F");
ll.remove(2);
System.out.println("Contents of ll after deletion: "
+ ll);
ll.removeFirst();
ll.removeLast();
System.out.println("ll after deleting first and last: "
+ ll);
Object val = ll.get(2);
ll.set(2, (String) val + " Changed");
System.out.println("ll after change: " + ll);
}
}
```

6) Write a program to demonstrate Hash set and Iterator classes.
Without iterator:-

```
import java.util.HashSet;
public class HashSetExample {
   public static void main(String args[]) {
HashSet<String>hset = new HashSet<String>();
hset.add("Apple");
hset.add("Mango");
hset.add("Grapes");
hset.add("Orange");
hset.add("Fig");
      //Addition of duplicate elements
hset.add("Apple");
hset.add("Mango");
hset.add(null);
hset.add(null);
 //Displaying HashSet elements
System.out.println(hset);
    }
}
```

With iterator:-

```
import java.util.HashSet;
import java.util.Iterator;

class IterateHashSet{
  public static void main(String[] args) {
     // Create a HashSet
HashSet<String>hset = new HashSet<String>();

     //add elements to HashSet
hset.add("Chaitanya");
hset.add("Rahul");
hset.add("Tim");
hset.add("Rick");
hset.add("Harry");

     Iterator<String> it = hset.iterator();
     while(it.hasNext()){
System.out.println(it.next());
     }
  }
}
```

7) Write a program to demonstrate Enumeration and Comparator interfaces.

```
import java.util.Enumeration;
import java.util.Vector;

class EnumerationTest
{
   public static void main(String arg[])
   {
      Enumeration<String> days;
      Vector<String>dayNames = new Vector<String>();
      dayNames.add("Monday");
      dayNames.add("Tuesday");
      dayNames.add("Wednesday");
      dayNames.add("Thursday");
      dayNames.add("Friday");
      dayNames.add("Saturday");
      dayNames.add("Sunday");

      // Assigns vector elements to enumeration
      days = dayNames.elements();
      while (days.hasMoreElements()) {
         System.out.println(days.nextElement());
      }
   }
}
```

----

8) Write a program to create a registration form with different controls, menus and demonstrate event handling.

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class student extends Frame implements ActionListener
{String msg;
  Button b1=new Button("save");
  Label l11=new Label("Student details",Label.CENTER);
  Label l1=new Label("Name:",Label.LEFT);
  Label l2=new Label("age:",Label.LEFT);
  Label l3=new Label("Sex(M/F):",Label.LEFT);
  Label l4=new Label("Address:",Label.LEFT);
  Label l5=new Label("Course:",Label.LEFT);
  Label l6=new Label("Semester:",Label.LEFT);
  Label l7=new Label("",Label.RIGHT);
  TextField t1=new TextField();
  Choice c1=new Choice();
  CheckboxGroupcbg=new CheckboxGroup();
  Checkbox ck1=new Checkbox("Male",false,cbg);
  Checkbox ck2=new Checkbox("Female",false,cbg);
  TextArea t2=new TextArea("",180,90,TextArea.SCROLLBARS_VERTICAL_ONLY);
  Choice course=new Choice();
  Choice sem=new Choice();
  Choice age=new Choice();
public student()
 {addWindowListener(new myWindowAdapter());
  setBackground(Color.cyan);
  setForeground(Color.black);
  setLayout(null);
  add(l11);
  add(l1);
  add(l2);
  add(l3);
  add(l4);
  add(l5);
  add(l6);
  add(l7);
  add(t1);
  add(t2);
  add(ck1);
  add(ck2);
  add(course);
  add(sem);
  add(age);
  add(b1);
  b1.addActionListener(this);
  add(b1);
  course.add("BSc c.s");
  course.add("BSc maths");
  course.add("BSc physics");
  course.add("BA English");
  course.add("BCOM");
  sem.add("1");
  sem.add("2");
  sem.add("3");
  sem.add("4");
  sem.add("5");
  sem.add("6");
  age.add("17");
  age.add("18");
  age.add("19");
  age.add("20");
  age.add("21");
  l1.setBounds(25,65,90,20);
  l2.setBounds(25,90,90,20);
  l3.setBounds(25,120,90,20);
  l4.setBounds(25,185,90,20);
  l5.setBounds(25,260,90,20);
  l6.setBounds(25,290,90,20);
  l7.setBounds(25,260,90,20);
  l11.setBounds(10,40,280,20);
  t1.setBounds(120,65,170,20);
  t2.setBounds(120,185,170,60);
  ck1.setBounds(120,120,50,20);
  ck2.setBounds(170,120,60,20);
  course.setBounds(120,260,100,20);
  sem.setBounds(120,290,50,20);
  age.setBounds(120,90,50,20);
  b1.setBounds(120,350,50,30);
}
public void paint(Graphics g)
{g.drawString(msg,200,450);}
public void actionPerformed(ActionEvent ae)
{if(ae.getActionCommand().equals("save"))
  {msg="Student details saved!";
   setForeground(Color.red); }
}
public static void main(String g[])
{student stu=new student();
 stu.setSize(new Dimension(500,500));
 stu.setTitle("student registration");
 stu.setVisible(true);
}
}
 class myWindowAdapter extends WindowAdapter
{public void windowClosing(WindowEvent we)
 {
  System.exit(0);
 }
}
```

9) Write a program to copy data from one file to another file.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class CopyExample
{
   public static void main(String[] args)
   {
FileInputStream instream = null;
FileOutputStreamoutstream = null;

try{
   File infile =new File("C:\\MyInputFile.txt");
   File outfile =new File("C:\\MyOutputFile.txt");

   instream = new FileInputStream(infile);
outstream = new FileOutputStream(outfile);

byte[] buffer = new byte[1024];

int length;
   /*copying the contents from input stream to
    * output stream using read and write methods
    */
   while ((length = instream.read(buffer)) > 0){
          outstream.write(buffer, 0, length);
   }

   //Closing the input/output file streams
instream.close();
outstream.close();

System.out.println("File copied successfully!!");

}catch(IOExceptionioe){
          ioe.printStackTrace();
 }
   }
}
```

10) Write a program to merge contents of two files and display output on console.

```
import java.io.*;
public class FileMerge
{
   public static void main(String[] args) throws IOException
   {
PrintWriter pw = new PrintWriter("file3.txt");
BufferedReader br1 = new BufferedReader(new FileReader("file1.txt"));
BufferedReader br2 = new BufferedReader(new FileReader("file2.txt"));
       String line1 = br1.readLine();
       String line2 = br2.readLine();
       // loop to copy lines of
       // file1.txt and file2.txt
       // to  file3.txt alternatively
       while (line1 != null || line2 !=null)
       {
if(line1 != null)
          {
pw.println(line1);
             line1 = br1.readLine();
          }

if(line2 != null)
          {
pw.println(line2);
             line2 = br2.readLine();
          }
       }
pw.flush();
       // closing resources
       br1.close();
       br2.close();
pw.close();

System.out.println("Merged file1.txt and file2.txt
alternatively into file3.txt");
BufferedReader in = new BufferedReader(new FileReader("file3.txt"));
String line = in.readLine();
while(line != null)
{
System.out.println(line);
  line = in.readLine();
}
in.close();
 }
}
```

11) Write a program to illustrate Serialization.

```
import java.io.*;
public class Employee implements java.io.Serializable {
  public String name;
  public String address;
  public transient int SSN;
  public int number;

  public void mailCheck() {
System.out.println("Mailing a check to " + name + " " + address);
  }
}

public class SerializeDemo {

  public static void main(String [] args) {
    Employee e = new Employee();
    e.name = "Reyan Ali";
e.address = "PhokkaKuan, Ambehta Peer";
e.SSN = 11122333;
e.number = 101;

    try {
FileOutputStreamfileOut =
      new FileOutputStream("/tmp/employee.ser");
ObjectOutputStream out = new ObjectOutputStream(fileOut);
out.writeObject(e);
out.close();
fileOut.close();
System.out.printf("Serialized data is saved in /tmp/employee.ser");
    } catch (IOExceptioni) {
i.printStackTrace();
    }
  }
}
```

12) Write a program to retrieve web page using URL class.

```
import java.net.*;
import java.io.*;

public class URLDemo {

   public static void main(String [] args) {
      try {
         URL url = new URL("https://www.amrood.com/index.htm?language=en#j2se");

System.out.println("URL is " + url.toString());
System.out.println("protocol is " + url.getProtocol());
System.out.println("authority is " + url.getAuthority());
System.out.println("file name is " + url.getFile());
System.out.println("host is " + url.getHost());
System.out.println("path is " + url.getPath());
System.out.println("port is " + url.getPort());
System.out.println("default port is " + url.getDefaultPort());
System.out.println("query is " + url.getQuery());
System.out.println("ref is " + url.getRef());
      } catch (IOException e) {
e.printStackTrace();
      }
   }
}
```

13) Write a program to load and display image and perform gray scale.

```
import java.awt.*;
import java.awt.image.BufferedImage;

import java.io.*;

import javax.imageio.ImageIO;
import javax.swing.JFrame;

public class GrayScale {

BufferedImage  image;
int width;
int height;

   public GrayScale() {

      try {
         File input = new File("digital_image_processing.jpg");
         image = ImageIO.read(input);
         width = image.getWidth();
         height = image.getHeight();

for(inti=0; i<height; i++){

for(int j=0; j<width; j++){

    Color c = new Color(image.getRGB(j, i));
int red = (int)(c.getRed() * 0.299);
int green = (int)(c.getGreen() * 0.587);
int blue = (int)(c.getBlue() *0.114);
               Color newColor = new Color(red+green+blue,

red+green+blue,red+green+blue);

image.setRGB(j,i,newColor.getRGB());
            }
         }

         File ouptut = new File("grayscale.jpg");
ImageIO.write(image, "jpg", ouptut);

      } catch (Exception e) {}
   }

   static public void main(String args[]) throws Exception
   {
GrayScaleobj = new GrayScale();
   }
}
```