

# BCSL-146

## Java Programming Lab



**Java Application**

**JDBC**

**Oracle DataBase**

**POLY MORPHISM in JAVA programming**

Types of Polymorphism

- compile time polymorphism (static or early binding)
- run-time polymorphism (dynamic or late binding)

**Multithreading for Beginner**

PROCESS

Thread 1, Thread 2, Thread n

**try catch finally throw throws**

```
gender = g;
birthDay = b;
pref = p;

public void setName (String n) {
    this.name = n;
}

public String getName () {
    return name;
}

public void setGender (char g) {
    this.gender = g;
}

public char getGender () {
    return gender;
}

public void setBirthDay (Date b) {
    this.birthDay = b;
}

public Date getBirthDay () {
    return birthDay;
}

public void setPref (Preference p) {
```

**JavaFX**

## **Object Oriented Programming using Java Lab**

### **LAB MANUAL**

---

#### **SECTION 1      JAVA PROGRAMMING LAB**

---

---

## PROGRAMME DESIGN COMMITTEE

---

Prof. (Retd.) S.K. Gupta, IIT, Delhi  
 Prof. T.V. Vijay Kumar JNU, New Delhi  
 Prof. Ela Kumar, IGDTUW, Delhi  
 Prof. Gayatri Dhingra, GVMITM, Sonipat  
 Mr. Milind Mahajani, Impressico Business Solutions, New Delhi

Sh. Shashi Bhushan, Associate Prof. SOCIS, IGNOU, New Delhi  
 Dr. Akshay Kumar, Associate Prof. SOCIS, IGNOU, New Delhi  
 Prof. P.V. Suresh, SOCIS, IGNOU, New Delhi  
 Prof. V.V. Subrahmanyam, SOCIS, IGNOU, New Delhi  
 Dr. M.P. Mishra, Associate Prof. SOCIS, IGNOU, New Delhi  
 Dr. Sudhansh Sharma, Assistant Prof. SOCIS, IGNOU

---

## COURSE DESIGN COMMITTEE

---

Prof. S. Balasundaram, JNU, New Delhi  
 Prof. Sonajahria Minz, JNU, New Delhi  
 Dr. Aditi Sharan, JNU, New Delhi  
 Dr. Sanjay Kumar Dubey, Amity University Noida  
 Dr. Rahul Johri, GGSIPU, New Delhi  
 Dr. Gurmeet Kaur, School of Law, IGNOU  
 Dr. Anand Gupta, School of Law, IGNOU

Sh. Shashi Bhushan, Associate Prof. SOCIS, IGNOU, New Delhi  
 Dr. Akshay Kumar, Associate Prof. SOCIS, IGNOU, New Delhi  
 Prof. P.V. Suresh, SOCIS, IGNOU, New Delhi  
 Prof. V.V. Subrahmanyam, SOCIS, IGNOU, New Delhi  
 Dr. M.P. Mishra, Associate Prof. SOCIS, IGNOU, New Delhi  
 Dr. Sudhansh Sharma, Assistant Prof. SOCIS, IGNOU

---

## FACULTY OF SOCIS

---

Prof. Sandeep Singh Rawat, Director, SOCIS, IGNOU  
 Prof. P.V. Suresh, SOCIS, IGNOU  
 Prof. V.V. Subrahmanyam, SOCIS, IGNOU  
 Prof. M.P. Mishra, Associate Professor, SOCIS, IGNOU  
 Prof. Akshay Kumar, Associate Professor, SOCIS, IGNOU  
 Prof. Divakar Yadav, SOCIS, IGNOU  
 Dr. Sudhansh Sharma, Associate Professor, SOCIS, IGNOU

---

## COURSE PREPARATION TEAM

---

Adapted from MCSL-210 (Section 2) and vetted by Prof. M.P. Mishra, SOCIS, IGNOU, New Delhi

---

### Course Coordinators

---

Prof. M. P. Mishra, SOCIS, IGNOU

---

## PRINT PRODUCTION

---

Registrar (Publication)

December, 2025

©Indira Gandhi National Open University, 2025

ISBN-

*All rights reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from the Indira Gandhi National Open University.*

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110068.

Printed and published on behalf of the Indira Gandhi National Open University, New Delhi by MPDD, IGNOU.

---

---

## COURSE INTRODUCTION

---

This laboratory course is based on courses MCS-206: Object Oriented Programming Using Java. There is a section in this course based on lab exercises for the MCS-206 course. This lab section explains about setting up a “Development Environment for Java programming”. This section also explains the Installation of Java Development Kit (JDK) and the Installation of Netbeans IDE. Subsequently, you learn about the setting of an important environment variable called the CLASSPATH. Further, some Java programs are given so that you can practice these using the programming environment. Towards the end, ten sessions are given; each session has a list of problems for which you have to write/develop Java Program. You should complete all the exercises in these sections to obtain better practical skills. You are advised to develop solution/ write code before each lab counselling sessions and get your doubt cleared during the session.



ignou  
THE PEOPLE'S  
UNIVERSITY



---

## Section-1: Java Programming Lab

---

Structure

Page No.

1.0 Introduction

1.1 Objective

1.2 Introduction to Java Programming

1.3 Setup Development Environment

1.3.1 Installation of Java Development Kit (JDK)

1.3.2 Installation of NetBeans IDE

1.3.3 Installing Apache NetBeans In Windows 10

1.3.4 Setting CLASSPATH

1.4 Some Example Programs

1.4.1 Core Java Programs

1.4.2 JDBC Programs

1.5 Lab Sessions

1.6 References/Further Readings

---

### 1.0 INTRODUCTION

---

For writing a program, you need to use IDE (Integrated Development Environment) for better and smooth programming experiences. Section-2 of this lab manual is for Java programming, which is based on MCS-206: Java Programming course. At the end of this section, you will get a list of lab sessions consisting of lab exercises that cover almost all the topics and concepts you will be learning in course MCS-206. These ten lab sessions consist of several programming exercises which you have to implement/write code and execute using NetBeans IDE. Some popular Java IDEs include NetBeans, Eclipse, and IntelliJ IDEA. In this lab manual, we will use NetBeans, an open-source IDEs, to write our programs.

In this section, we will talk about the use of the development environment of NetBeans, and all the example programs has been executed using this IDE. To use IDE for Java programming, you need to install Java Development Kit(JDK).

---

### 1.1 OBJECTIVE

---

The objectives of this section are:

- To install different IDEs for Java/J2EE programming,
- Develop simple applications using IDEs, and
- Use the given list of exercises for your assigned lab sessions.

## 1.2 INTRODUCTION TO JAVA PROGRAMMING

Java is a high-level language, and the language that a computer can directly understand is called **machine language**. The **compiler** job is the translation of a program from a high-level language(c/c++/Java etc.) to a low-level language. So you have to run your program written in a high-level language, using the compiler. The machine-language program that is the output of the compiler is often called the **object program** or **object code**. When you do this, you say that you have **compiled** your program. An interesting thing to note is that the Java compiler does not translate your program into the machine language that can run on your computer; it translates your Java program into a code called **Bytecode**. The Bytecode is not the machine language for any particular operating system(computer). Instead, it is a machine language for a hypothetical computer that is known as a **virtual machine**. The program that translates source code into Bytecode is a kind of interpreter called the **Java Virtual Machine( JVM)**.

To run your Java program on your computer, first, you have to use the compiler to translate your Java program into Bytecode. Then you have to use the particular JVM for your computer system for translating each bytecode instruction into machine language and run it.

### Java Bytecode

When you compile a Java program, the Java compiler does not translate your program into the machine language for your particular computer. Instead, your Java program is translated to **Bytecode**. Remember that Java Bytecode is not the machine language for any particular computer. Instead, it is a machine language for a hypothetical computer known as a **virtual machine**. Translating Java bytecode into a machine-language program for an actual computer is performed in **JVM**. Also, the JVM runs the Java bytecode.

To run your Java program, you first have to compile it into Bytecode using a compiler; then you can use a specific JVM to translate each bytecode instruction into machine language and run it. Figures 1.1 and 1.2 show the entire process of Java program execution.

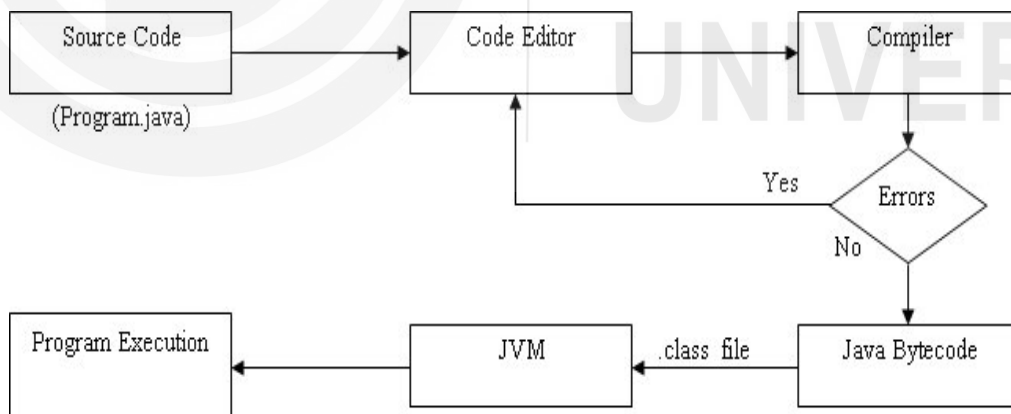


Figure: 1.1: Java Program Writing and Execution

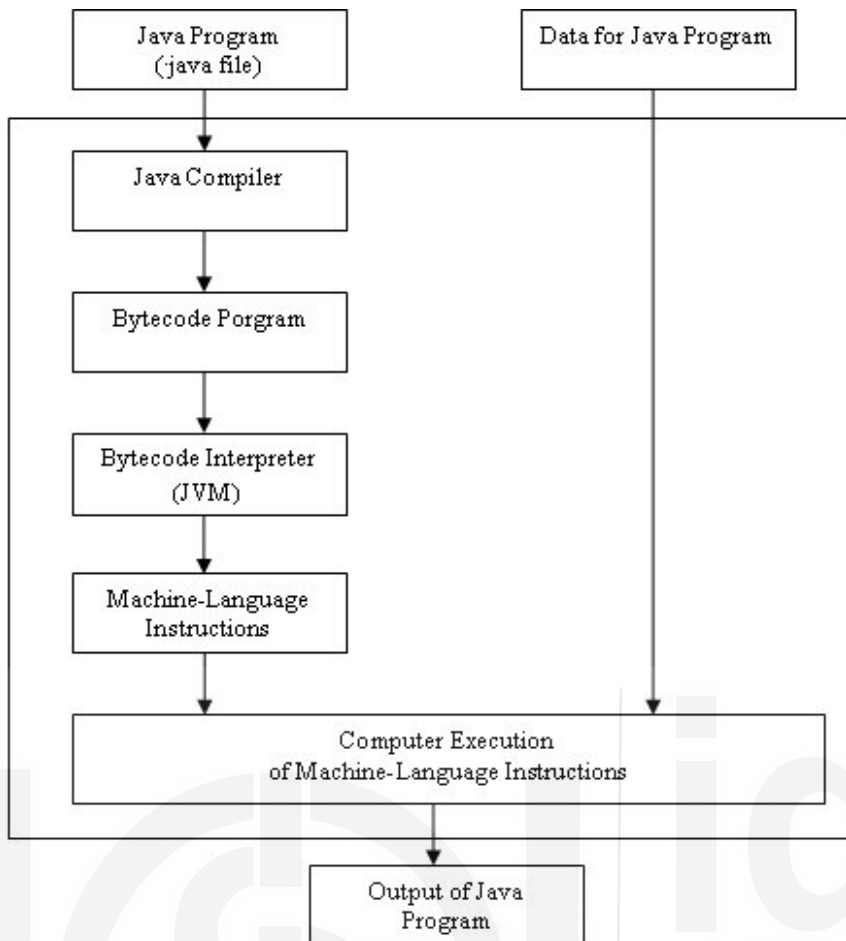


Figure 1.2: Compiling and Running a Java Program

### Writing, Compiling, and Running a Java Program

When writing Java programs, you are advised to use an IDE, as it will make your programming easier. Also, you can write a Java program in a simple text editor, for example, Notepad, in a Windows environment. Generally, in a complex program, there may be more than one class. In that case, the name of that file must be the name of the class in which the main method is defined with .java added to the end.

Signature of main method :

```
public static void main(String[] args)
```

If you are writing a Java program using a simple editor, then to compile that Java class, you have to use a free Java system distributed by Oracle for Windows, Linux, or some other Operating System; you have to use the command **javac** followed by program file with extension .java. For example, to compile a java program that has a class named **MyClass** that is in a program file named **MyClass.java**, you give the following command to the operating system:

```
javac MyClass.java
```

Thus, to compile the class in your program, you would give the following command:

```
javac YourProgram.java
```

When you compile a Java class, it is translated into the class file-its bytecode and is placed in a file whose name is the name of the class followed by **.class**. That means after executing **javac**

YourProgram.java you will get YourProgram.class file as a result. Now you can run YourProgram.class file

To run a Java program by giving the **command java**, followed by the name of the class you think of as the program. For example, to run the program in YourProgram.class, you would give the following one-line command:

```
java YourProgram
```

Now let us try to write a simple Java program using Notepad. The program to display **Welcome to Java Programming** is written as shown in figure 1.3



```
File Edit Format View Help
public class HelloJava
{
    public static void main(String args[])
    {
        System.out.println("Welcome to Java Programming");
    }
}
```

Figure 1.3: Java Program Written in Notepad

Now we can save and compile this program using command prompt. This program is saved in C:\MyJava directory. The Java program file name is HelloJava.java. Before you compile this program, you have to set the path for the location where your **javac** and **java** files are saved. Here we have set the path to the bin directory of your JDK folder:

```
set path = C:\Program Files\Java\jdk-10.0.2\bin
```

Details about Classpath you will learn later in this section.

You can see in figure 1.4, which is a screenshot of the compilation and execution of this program.



```
C:\MyJava>javac HelloJava.java

C:\MyJava>java HelloJava
Welcome to Java Programming

C:\MyJava>_
```

Figure 1.4: Screenshot of the Command Prompt

Remember that when you run a Java program, actually, you are running the Java bytecode interpreter on the compiled version of your program, and hence you are not writing .class after the file name.

Java programs can be written and executed easily using an **integrated development environment(IDE)**. An IDE provides you with a text editor with menu commands for compiling and running the Java programs. An IDE provides many useful GUI tools which are useful for developing applications. Some of the common features of IDEs are:

- **Code Editor:** It is a space for writing and editing the source code.
- **Compiler:** It is used to transform the source code of the program written by you into a form that is executable by a computer.

- **Debugger:** It is used during the testing of the application programs. It helps to debug application programs during development.
- **Builder:** This automates common tasks of programmers, such as building the application to run/execute the application.
- **Browsers:**
  - **Class Browser** is used to examine and reference the properties of an object-oriented class hierarchy.
  - **Object Browser** is used to examine the objects instantiated in a running application program.
- **Class hierarchy Diagram:** It allows the programmer to see the project structure of programming code, including different classes in the project.

This lab section is based on the MCS-206 course in which you will be learning about basic constructs of Java programming, including data types, operators, basic OOP concepts and other Java programming concepts. You need to learn Java programming in the MS-206 course and try to implement the programming exercises given in that course along with the exercises given in this section of the lab manual.

Now let us see how to set up the development environment for Java programming using NetBeans IDE.

## 1.3 SETUP DEVELOPMENT ENVIRONMENT

### 1.3.1 Installation of Java Development Kit (JDK)

The best thing about IDE is that its intelligence provides code completion hints during coding in IDE. It also provides easy code, file and folder navigation. For writing java program using IDE, we need to set up the development environment and set up a compiler, builder, and editor (mainly JDK and IDE). The Java Development Kit (JDK) is a software development environment that is used to develop java applications. Before installation of any IDE ( in our case NetBeans), **JDK installation is required**. That is why we need to install the latest version of JDK in our system. In this section, first, we will see how to install JDK and NetBeans IDE and then to use this IDE, we will write and execute our java programs.

For the Setup development environment, we need to install a compiler, a builder, and an editor (mainly JDK and an IDE). That is why we need to install the latest version of JDK in our system. The best thing about an IDE is that its intelligence provides code-completion hints while coding. It also provides easy code, file and folder navigation. In this unit Windows 10 based installation and environment setup are shown.

- First, check whether Java is already installed by running the command “java –version” in the terminal or command prompt, as shown in Figure 1.5.

```

20-10-2021 02.07 PM 50,016 verify.dll
20-10-2021 02.07 PM 24,928 w2k_lsa_auth.dll
20-10-2021 02.07 PM 145,248 windowsaccessbridge-64.dll
20-10-2021 02.07 PM 442,208 wsdetect.dll
20-10-2021 02.07 PM 17,760 wsgen.exe
20-10-2021 02.07 PM 18,272 wsimport.exe
20-10-2021 02.07 PM 17,760 xjc.exe
20-10-2021 02.07 PM 77,152 zip.dll
      159 File(s)      68,604,872 bytes
       5 Dir(s)  156,014,473,216 bytes free

C:\Program Files\Java\jdk-10.0.2\bin>java -version
java version "10.0.2" 2018-07-17
Java(TM) SE Runtime Environment 18.3 (build 10.0.2+13)
Java HotSpot(TM) 64-Bit Server VM 18.3 (build 10.0.2+13, mixed mode)

```

Figure 1.5: Checking Java Version

- If Java is not installed on your system, first download the latest version of JDK from its official website (<https://www.oracle.com/in/java/technologies/javase-downloads.html>). You can choose any latest version of JDK for development practice, but it is strongly recommended to choose the LTS (Long Term Support) version only for the production environment. Choose the JDK package as per your system/platform.
- Install JDK in your system and recheck the installed version.
- Remember that you need Java Runtime Environment(JRE) to run a Java program. A JRE provides you with a Java virtual machine (JVM), Java class libraries, and the Java class loader. You do not have to download JRE separately as a JRE is part of a Java development kit (JDK).

### 1.3.2 Downloading and Installation of Java Development Kit (JDK)

- If Java is not installed in your system, then you have to download the latest version of JDK from its official website ([java.com/download/ie\\_manual.jsp](http://java.com/download/ie_manual.jsp)). For practicals of this course, you can choose the latest version of JDK, but it is advised that you choose the LTS (Long Term Support) version only for the project development/ production environment.
- Install JDK in your system, and then you may recheck the installed version.



Figure 1.6: Downloading JDK



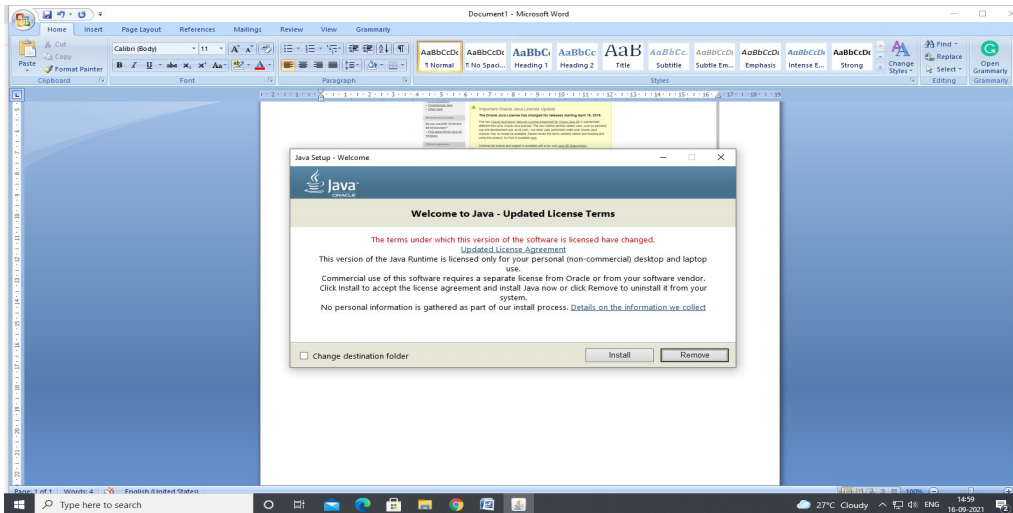


Figure 1.7: Accepting Licencing Terms for Downloading JDK



Figure 1.8: Installing JDK

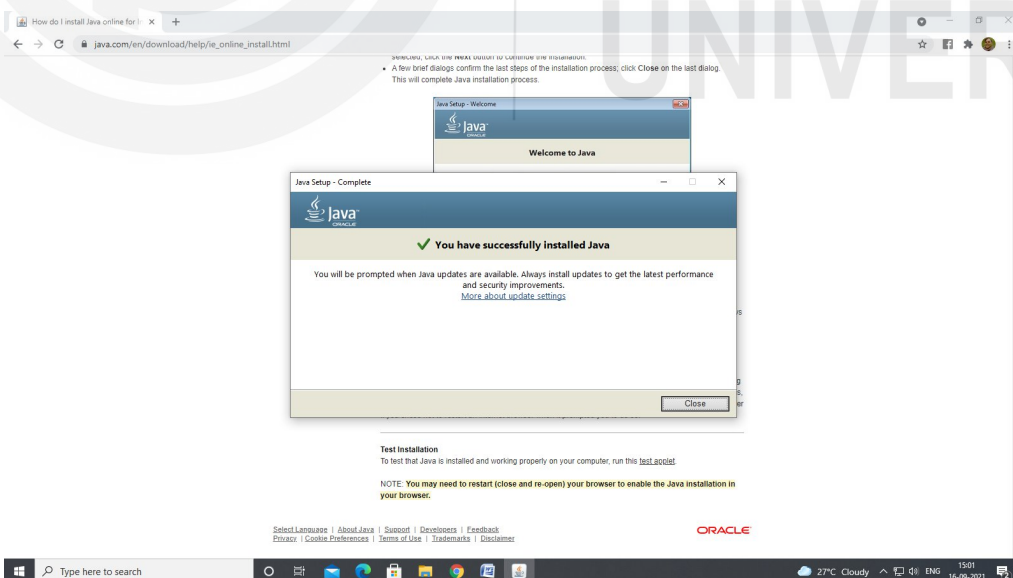


Figure 1.9: JDK Installed

Once JDK is successfully installed, then you can download and install NetBeans IDE. The steps for installing NetBeans IDE for Windows 10 is given below.

### 1.3.3 Installing Apache NetBeans -In Windows 10

NetBeans is an open-source Java IDE. It is one of the biggest and most popular IDE. It is supported by a variety of operating systems like Linux, Windows, macOS, Solaris, etc., along with feature-limited OS-independent versions.

Using NetBeans, it is very easy to create custom software applications. NetBeans highlights Java code syntactically as well as semantically. Also, there are many tools in NetBeans that help in writing bug-free code. NetBeans is primarily a Java IDE, but it also has extensions for working in other programming languages, including C, C++, PHP, HTML5, JavaScript, etc.

To download Apache NetBeans Idea, visit the official site of Apache NetBeans-  
<https://netbeans.apache.org/download/index.html>

Download the package as per your platform(Linux/Windows/macOS). In this section, we are using Windows 10 as a platform for installing NetBeans IDE.

For this lab manual, Apache NetBeans IDE 12.5 is used. To install it, locate the Apache-NetBeans-12.5-bin-windows-x64.exe setup file in your system, where it is saved after download and run as administrator. The User access control wizard will be displayed. Click on Yes. Setup will configure the installer. If you want to customize the installation during the installation process, click on the Customize button and choose the features as per your need. Follow the instructions of installation Wizard, and finally, click on Install to install the software; the installation process will begin; wait till the process finishes. Now, click on the Finish Button to close the wizard. After installation, you are ready to use the NetBeans IDE.

Go to the start program, navigate to Apache NetBeans, click on Apache NetBeans IDE 12.5 link to open the IDE, or double click on the Apache NetBeans IDE 12.5 icon created in Desktop as well to open the IDE.

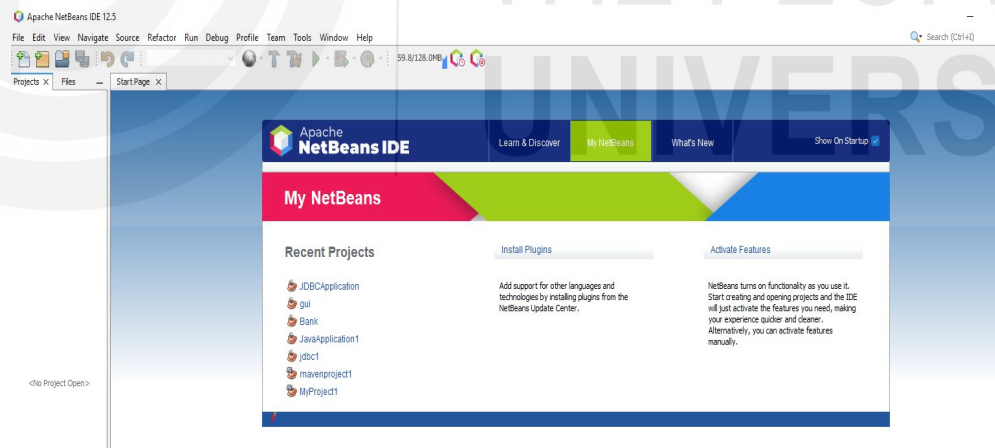


Figure 1.10: NetBeans IDE

Welcome to the First look of Apache NetBeans IDE 12.5 as shown in figure – 1.10.

### 1.3.4 Setting CLASSPATH

The PATH and CLASSPATH are the two very important environment variables of the Java environment. JDK uses these variables **to find the JDK binaries used to compile and run Java class files** which are compiled Java bytecodes. For example, you can add the bin directory path of JDK or

JRE in CLASSPATH, so that any binaries under the directory can be accessed directly without specifying the absolute path. As a programmer, you need to set the CLASSPATH when there is a need to load a class that is not present in the current directory or any sub-directories. The Classpath **tells JDK tools and applications where to find third-party and user-defined classes**, i.e classes that are not Java extensions or part of the Java platform. You can use Classpath as a parameter in the JVM or the Java compiler that specifies the location of user-defined classes and packages. Also, the CLASSPATH is used to define the path and to find third-party and user-defined classes. Now let us see how to set CLASSPATH .

PATH is the environment variable where we specify the locations of binaries. The main difference between PATH and CLASSPATH is that path is set for java tools in java programs like Java and javac, which are used to compile your code. CLASSPATH is used by System or Application class loader to locate and load compile Java bytecodes stored in the .class file.

You need to set the CLASSPATH in the situation :

- When there is a need to load a class that is not present in the current directory or any sub-directories.

The CLASSPATH will have what you are setting as the CLASSPATH variable. In the CLASSPATH you will have a directory name or file name at the end. The following points list what should be the end of the CLASSPATH.

- If you have a JAR or zip file, which contains class files, the CLASSPATH ends with the name of that zip or JAR file.
- If class files are placed in an unnamed package, then the CLASSPATH will end with the class file's directory.
- If class files are placed in a named package, the CLASSPATH will end with the directory containing the root package in the full package name.

If you have to set multiple classpaths, then you need to separate each CLASSPATH by a semicolon (;).

There are two ways for setting the Path in Java:

1. Temporary setting
2. Permanent setting

You can set the temporary Path of JDK by performing the following steps:

- Open the command prompt, then
- Copy the path of the JDK/bin directory, and finally
- Write/paste the JDK/bin in the command prompt

```
set path = C:\Program Files\Java\jdk-10.0.2\bin
```

Let us see it in figure 1.11 given below:

```
C:\MyJava> set path = C:\Program Files\Java\jdk-10.0.2\bin

C:\MyJava>
```

Figure 1.11: Setting Path of JDK/bin directory

### Set Permanent Path of JDK in Windows

To set the permanent Path of JDK, you have to perform the following steps:

1. Go to MyComputer properties -> Select the advanced tab
2. Select environment variables -> new tab of the user variable
3. Write the Path in variable name -> write Path of JDK/ bin folder in variable value
4. Select ok three times as shown in figures 1.12 to figures 1.14
  - First, click on environment variables.

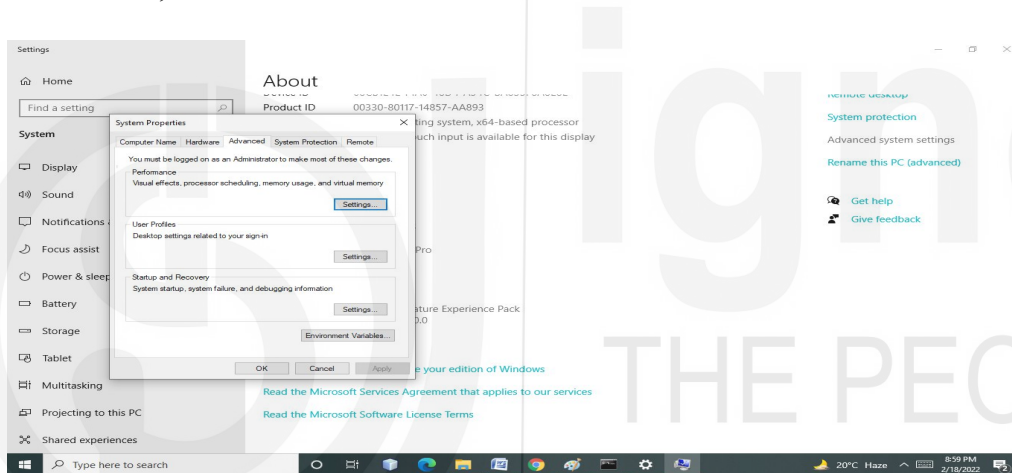


Figure 1.12: Setting Environment Variable

- Then click on the new tab of user variables.

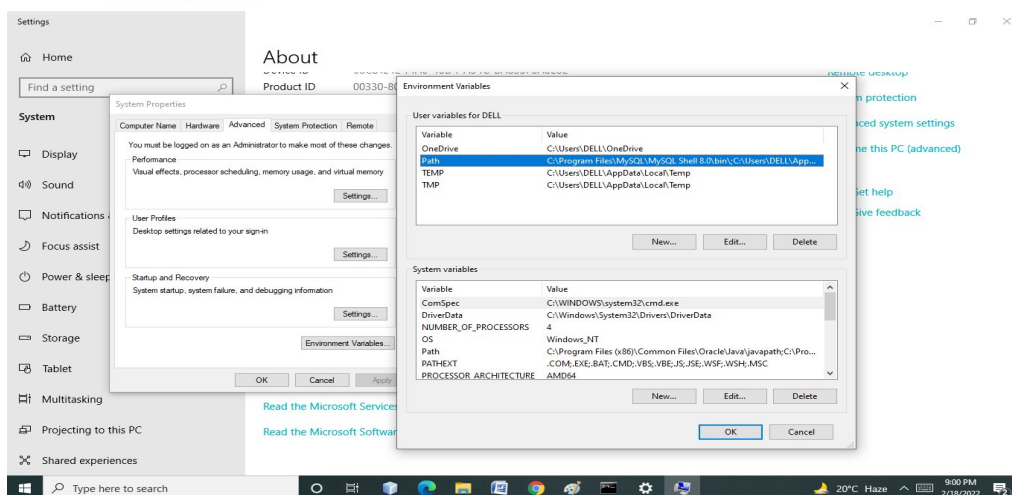


Figure 1.13: Add New User Variable

- Finally, write the path in the variable name.

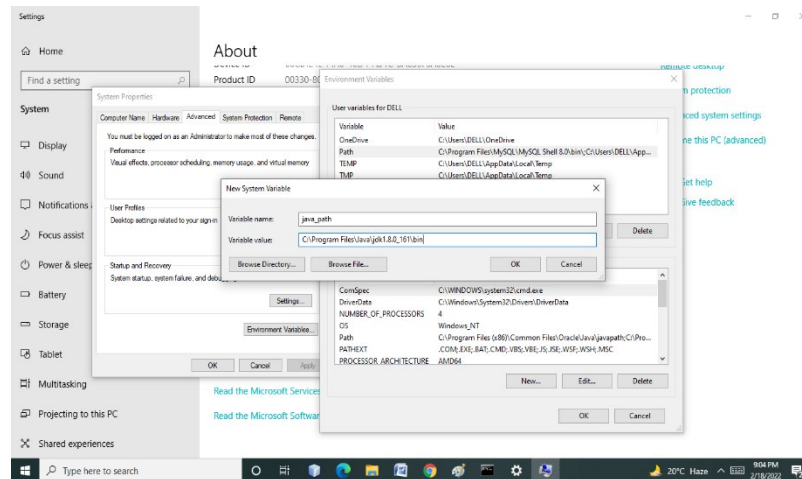


Figure: 1.14: Writing Path

Before you get ready to write java programs, you are advised to read at least the first two units of the MCS-206 course. There you will learn basic constructs for java programming. You must know the basic structure of the java program, data types, variables declarations and conventions used for naming variables, java keywords, programming comments etc. Also, you are advised to write the program code given in your course MCS-206 and execute them for better learning and understanding of concepts.

## 2.4 SOME EXAMPLE PROGRAMS

### 2.4.1 Core Java Programs

Example 1: First Java Program which displays a welcome message

Step-1: Open NetBeans IDE -> New Project

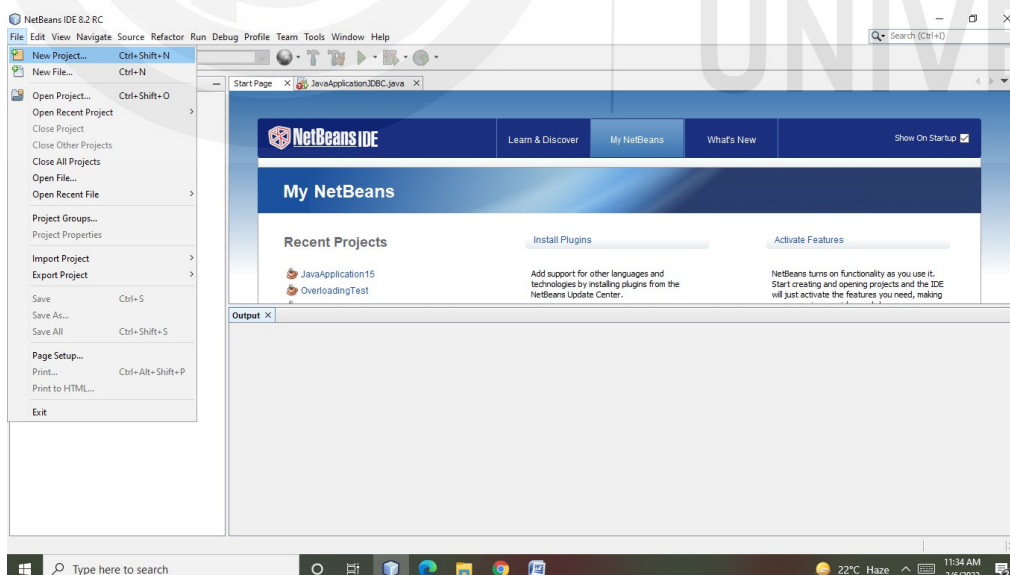


Figure 1.15: Opening NetBeans IDE

Step-2: Select Java as a category, then select Java Applications as categories.

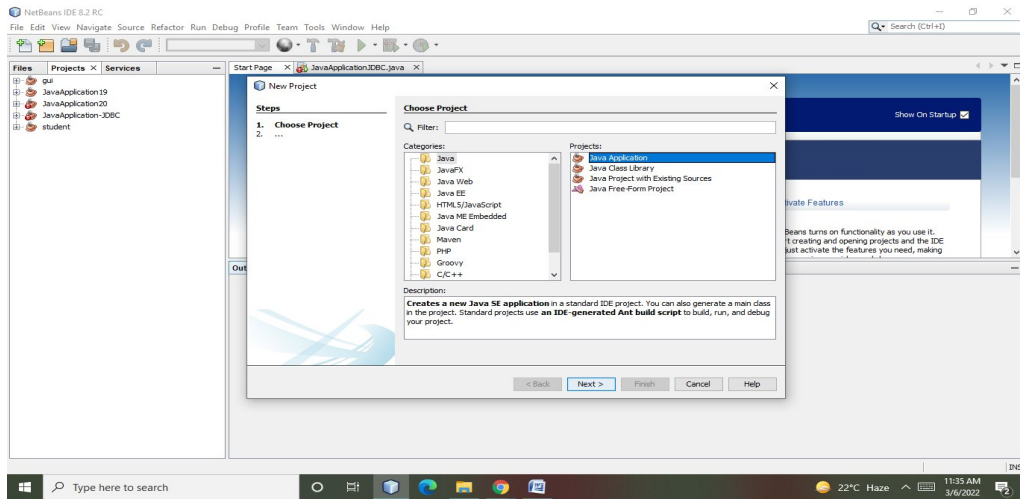


Figure 1.16: Selecting Project (Application) category

Step-3: Rename your project name as FirstApplication, then proceed further

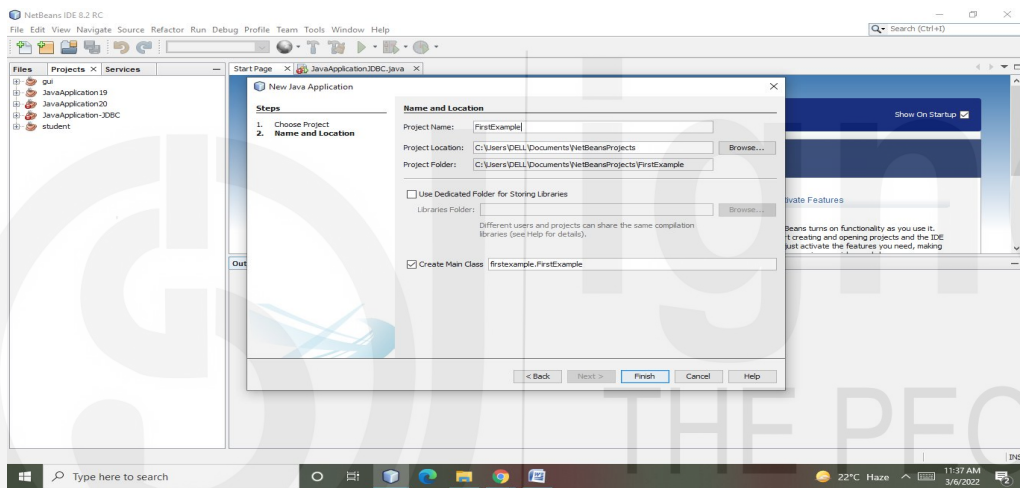


Figure 1.16: Giving a Name to Project

You will get the following screen. This environment is your starting point to write your program.

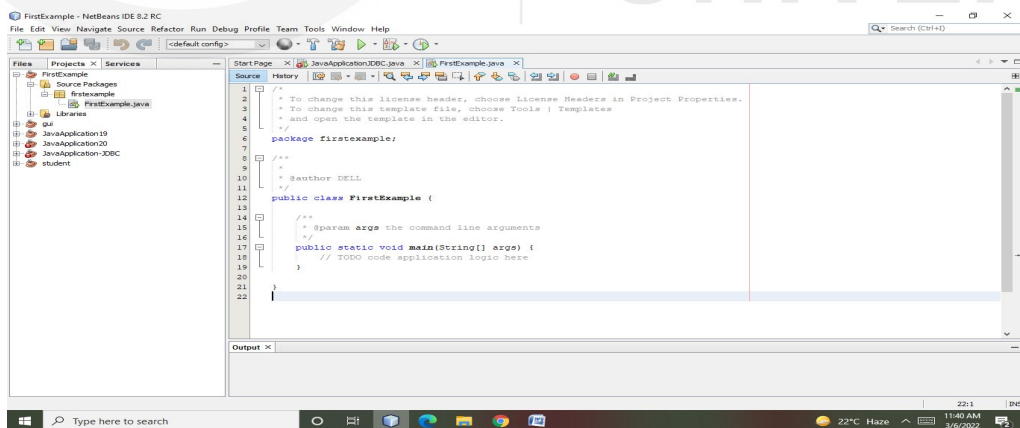


Figure 1.17: Using IDE for Writing Program

Now let us write our first java program using NetBeans IDE.

As you can see in figure 1.18 the first statement written is - `System.out.println("Welcome to Java Programming");`



Now save and compile this program.

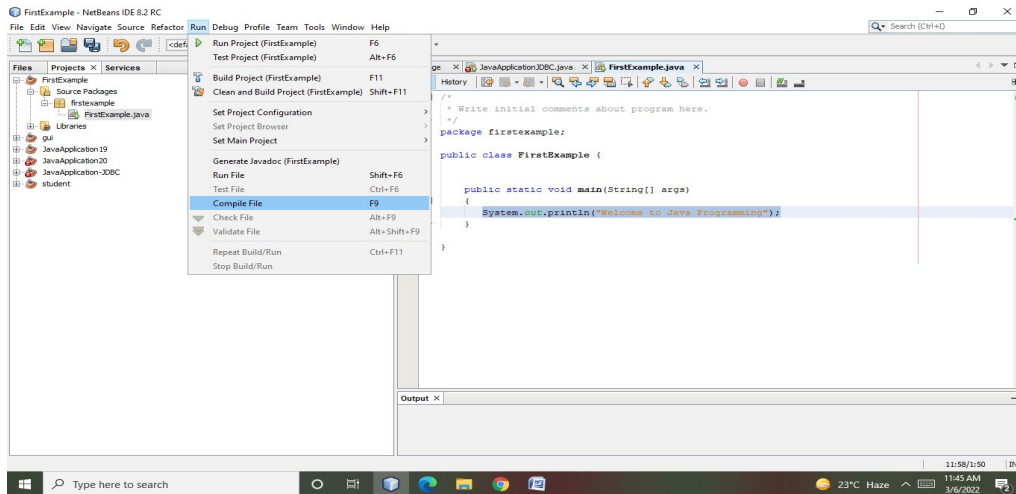


Figure 1.18: Java Program

Once your program is compiled successfully, you will get the following message in the output window.

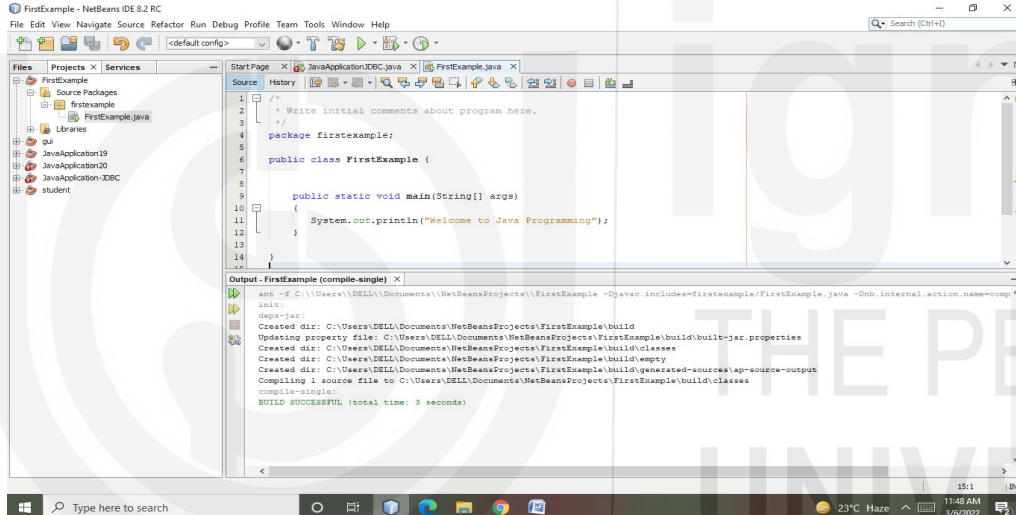


Figure 1.19 : Compiling Java Program

For final output, you need to run the project.

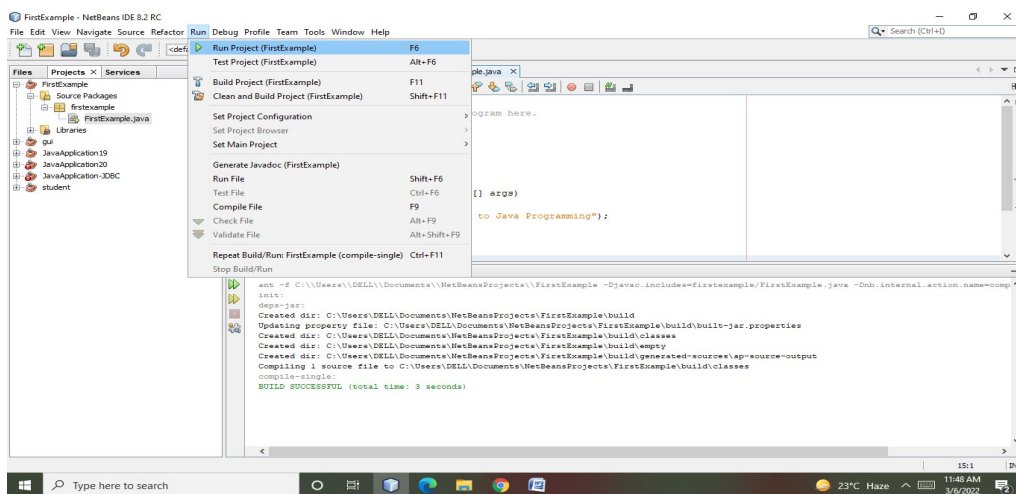


Figure 1.20: Running Application

Finally, you will get the result displayed in the output window.

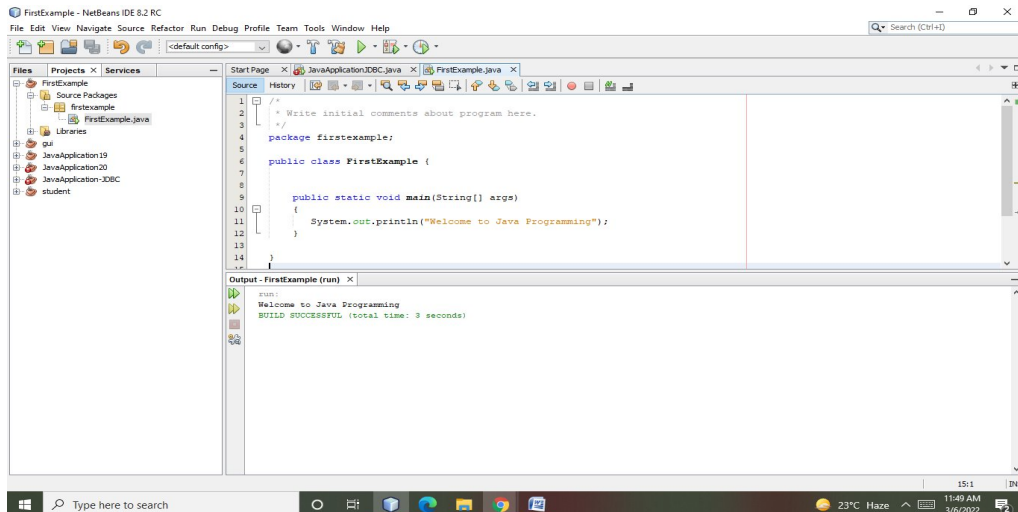


Figure 1.21: Output of Application

Now you can write and run ( execute) your program using the steps mentioned in Example -1 above. You need to remember that when you write and compile some complex programs, there is the possibility that your program will not be successfully compiled in the first attempt. You may have some errors, including syntax errors. You have to address(remove) those errors then you have to compile your program again.

Further, we will have some more example programs to help you learn about writing java programs.

Example-2: Java Program to show the use of Scanner class

```
/*
 * Use of Scanner Class
 */
package useofscannerclass;
import java.util.Scanner; //Scanner class

public class UseofScannerClass
{
    public static void main(String[] args)
    {
        Scanner myObj1 = new Scanner(System.in); // Scanner object
        System.out.println("Please Enter Your Name:-");
        String name = myObj1.nextLine();
        Scanner myObj2 = new Scanner(System.in);
        System.out.println("Please Enter Your Address:-");
        String add = myObj2.nextLine();
        System.out.println("Name: " + name);
        System.out.println("Address: " + add);
    }
}
```

Output:

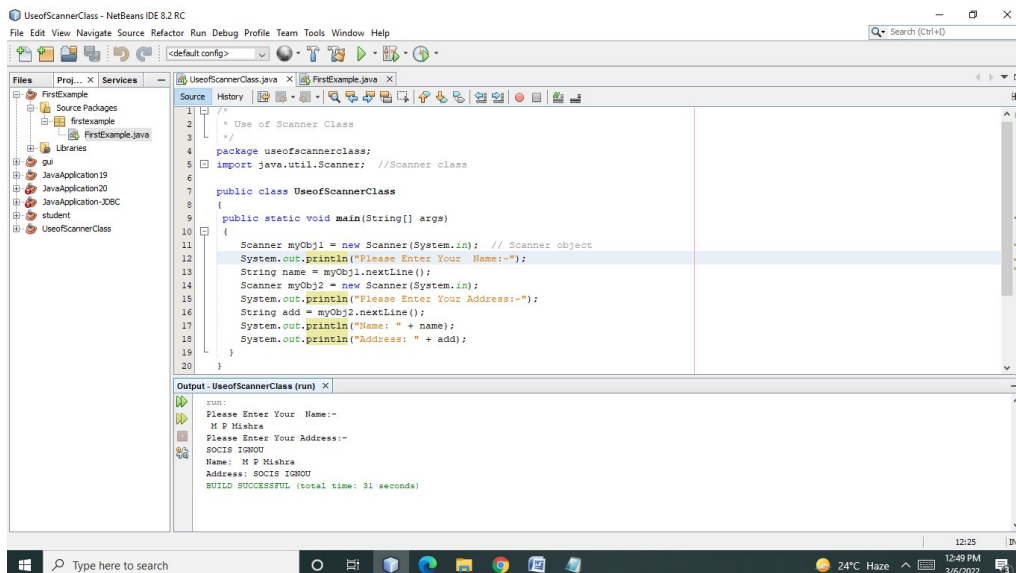


Figure 1.22: Compiling and Running Program

Example-3: Java program to create BankAccount class.

```
package bankaccount;
import java.util.*;
public class BankAccount {
    Scanner s=new Scanner(System.in);
    int accno;
    int am1;
    int amount;
    int num1;
    void deposit(){
        System.out.println("Please enter account number:");
        accno=s.nextInt();
        System.out.println("Please enter the amount:");
        am1=s.nextInt();
        amount += am1;
        System.out.println("You have sucessfully added the the amount:"+am1);
        menu();
    }
    void withdraw(){
        System.out.println("Please enter the amount:");
        am1=s.nextInt();
        amount -= am1;
        System.out.println("You have sucessfully withrawed the the amount:"+am1);
        menu();
    }
    void balance(){
        System.out.println("Your available balance is: "+amount);
        menu();
    }
    void menu(){
        System.out.println("MENU:")
        System.out.println("1. To deposit");
        System.out.println("2. To widraw the amount");
        System.out.println("3 To check balance");
    }
}
```

```

        System.out.println("Please choose the option: ");
        num1 = s.nextInt();
        switch(num1)
        {
            case 1: deposit();
                break;
            case 2: withdraw();
                break;
            case 3: balance();
                break;
            default:
                menu();
        }
    }
    public static void main(String[] args)
    {
        BankAccount MyAccount = new BankAccount();
        MyAccount.menu();
    }
}

```

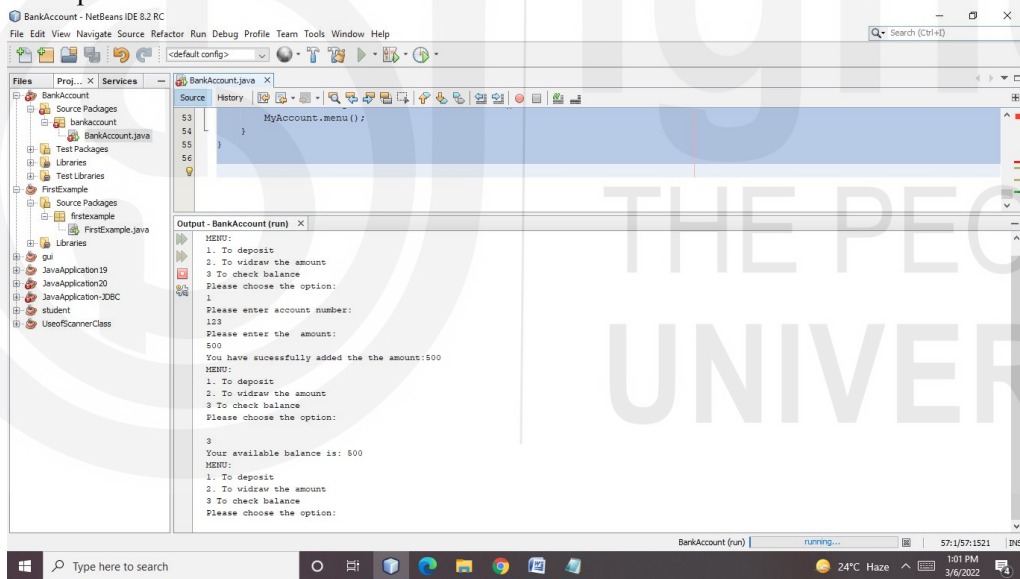
**Output:**

Figure 1.23: Compiling and Running a Program

**Example-4:** Java program to demonstrate the use of the Thread class methods- setPriority and getPriority.

```

package multithreadingdemo;
import java.lang.*;
public class MultiThreadingDemo extends Thread
{
    public void run()
    {
        System.out.println("Inside run method");
    }
}

```

```

public static void main(String[] args)
{
    MultiThreadingDemo t1 = new MultiThreadingDemo();
    MultiThreadingDemo t2 = new MultiThreadingDemo();
    MultiThreadingDemo t3 = new MultiThreadingDemo();
    System.out.println("t1 thread priority : "+ t1.getPriority());
    System.out.println("t2 thread priority : "+ t2.getPriority());
    System.out.println("t3 thread priority : "+ t3.getPriority());
    t1.setPriority(4);
    t2.setPriority(5);
    t3.setPriority(8);
    System.out.println("t1 thread priority : "+ t1.getPriority());
    System.out.println("t2 thread priority : "+ t2.getPriority());
    System.out.println("t3 thread priority : "+ t3.getPriority());
    System.out.println("Currently Executing Thread : "
        + Thread.currentThread().getName());
    System.out.println("Main thread priority : "
        + Thread.currentThread().getPriority());
    Thread.currentThread().setPriority(10);
    System.out.println("Main thread priority : "
        + Thread.currentThread().getPriority());
}
}

```

Output:

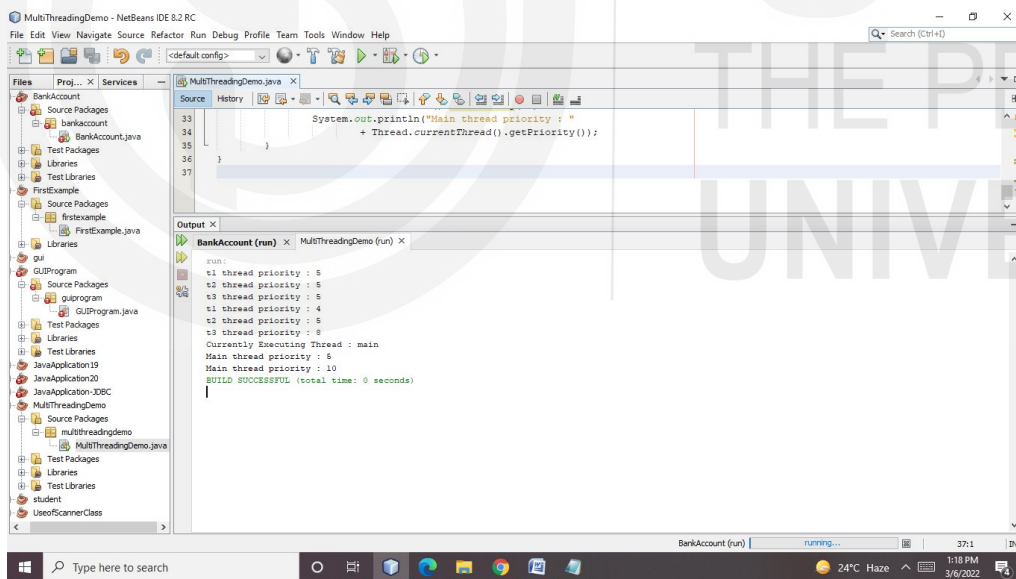


Figure 1.23: Compiling and Running a Program

## 2.4.2 JDBC Programs

For managing a database, you need to use a database server. Here we are using MySQL database for our program writing. You have to download and install MySQL from:

<https://www.mysql.com/downloads/>

Suppose you have to compile the following JDBC program :

Example 5: JDBC Program to manage PGDCA Student's records.

```

/*JDBC Program */
package jdbcapplication;
import java.sql.*;
public class JDBCApplication
{
    public static void main(String args[])
    {
        try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con=DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/jdbc2","root","mpmishra");
            //Here jdbc2 is database name, root is username and password is mpmishra
            System.out.println("Welcome to JDBC Programming");
            Statement stmt=con.createStatement();
            String sql = "CREATE TABLE pgdcastudent( RollNumber int(10),Name
            varchar(40), ProgrammeCode char(8))";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO pgdcastudent VALUES (111102, 'Ravi', 'PGDCA')";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO pgdcastudent VALUES (111103, 'Rajeev','GDCA')";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO pgdcastudent VALUES (111104,'Rahul', 'GDCA')";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO pgdcastudent VALUES (111105,'Rajesh', 'PGDCA')";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO pgdcastudent VALUES (111101,'Mohan', 'PGDCA')";
            stmt.executeUpdate(sql);
            ResultSet rs=stmt.executeQuery("select * from pgdcastudent");
            while(rs.next())
            {
                System.out.println("RollNumber:"+rs.getInt(1)+" Name:"+rs.getString(2)+"
                ProgrammeCode:"+rs.getString(3));
            }
            con.close();
        }
        catch(Exception e)
        {
            System.out.println("My Exception:"+e.toString());
        }
    }
}

```

To compile and run jedbc program, you need to provide a connector jar file. This file you have to download from: <https://dev.mysql.com/downloads/connector/j/>



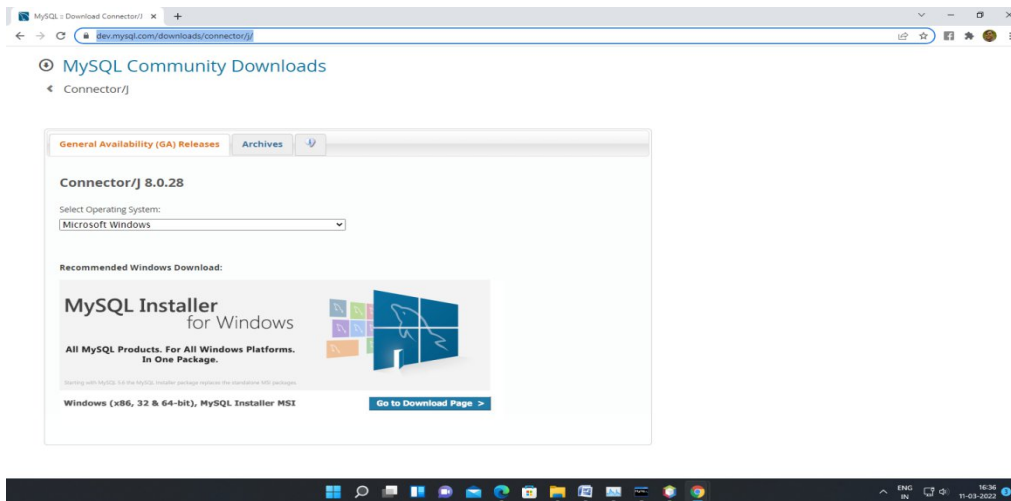


Figure 1.24: Downloading MySQL-Java Connector file

The JAR file - `mysql-connector-java-8.0.28.jar` is downloaded at "`C:\Users\ignou\Desktop\mysql-connector-java-8.0.28\mysql-connector-java-8.0.28.jar`".

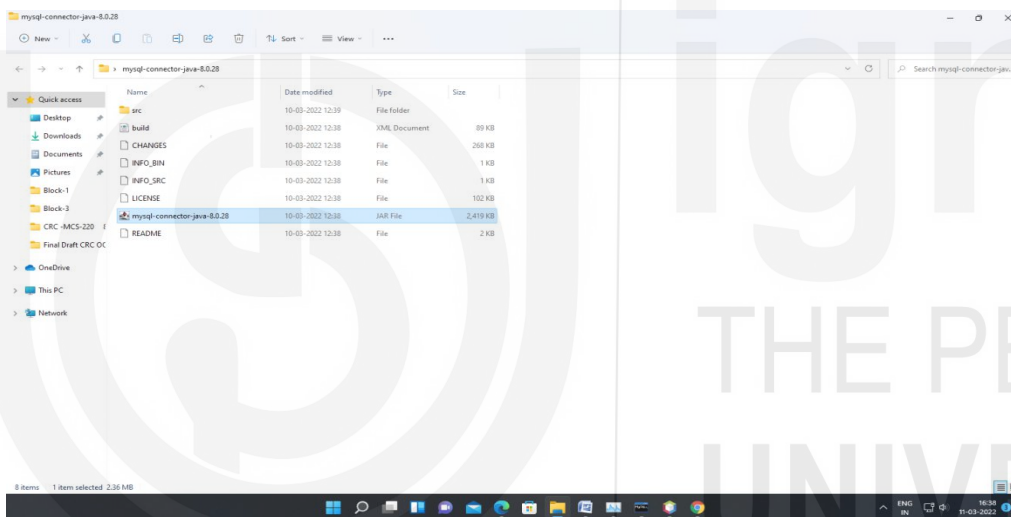


Figure 1.25: Download Connector File

This file must be included in the library of your JDBC Program, as shown in figure 1.26 below. To add this jar file, you need to do the following:

1. Right click on Application and select properties.

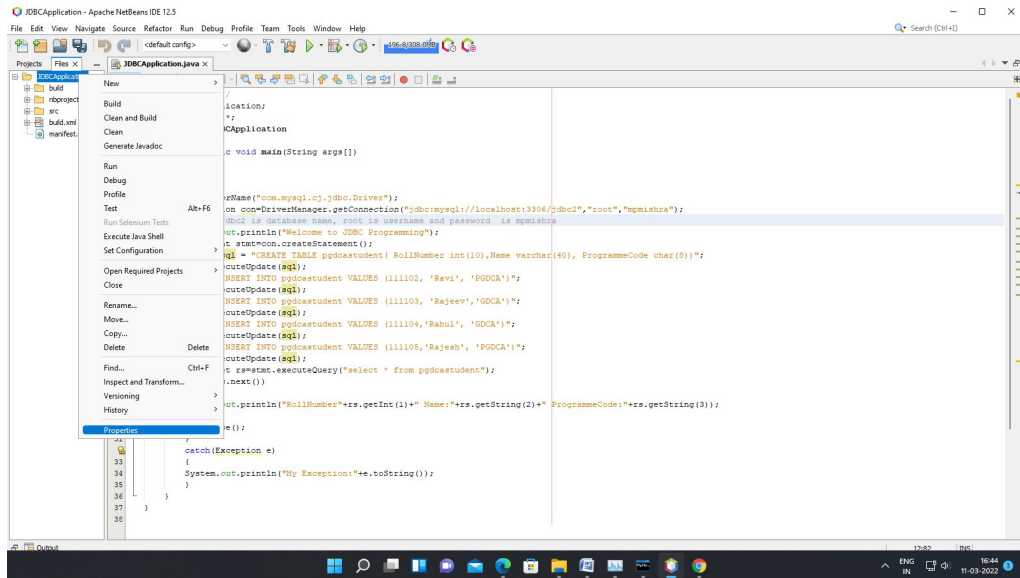


Figure 1.26 : Screenshot of Selecting Application Properties

2. Select Libraries, then add the file: "**C:\Users\ignou\Desktop\mysql-connector-java-8.0.28\mysql-connector-java-8.0.28.jar**" in Classpath (Remember that you have downloaded Java – MYSQL connector and added it in the Classpath of JDBC program/application library). Now you can compile and run this JDBC program.

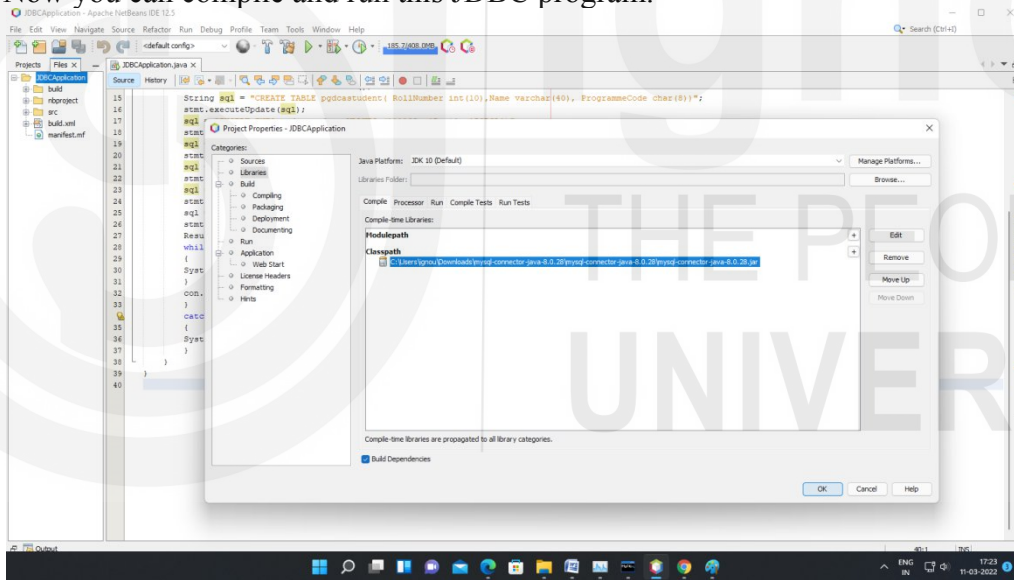


Figure 1.27: Adding MYSQL JDBC connector JAR file in Application Library

Output:

As you can see in figure 1.28, after the successful execution of the program given in example -5 above, you will get a database table named 'pgdca' created and five rows inserted in it.

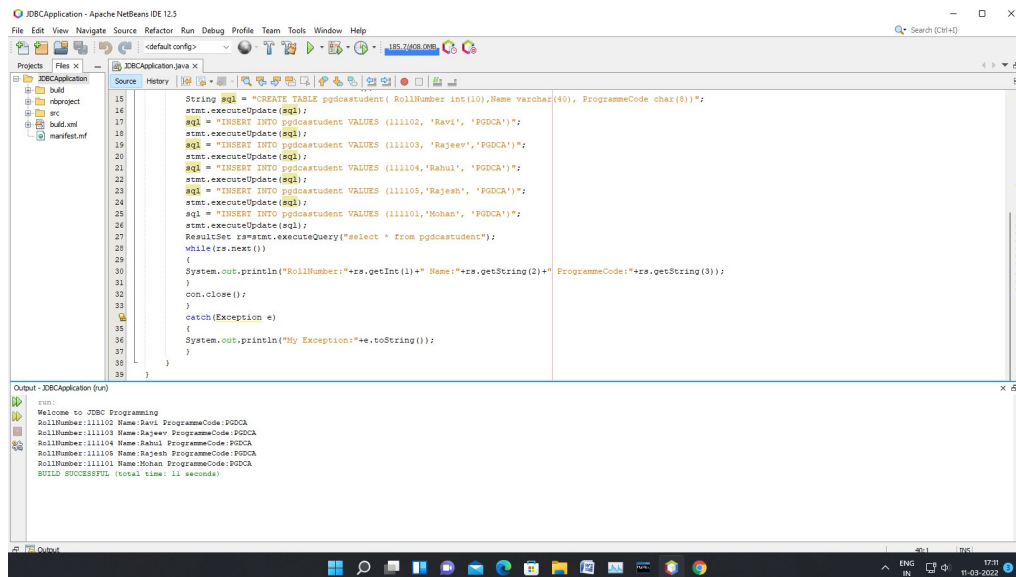


Figure 1.28: Output of JDBC Program

Also, when you verify the table creation and records in table pgdcastudent in jdbc2 database of MySQL you will get:

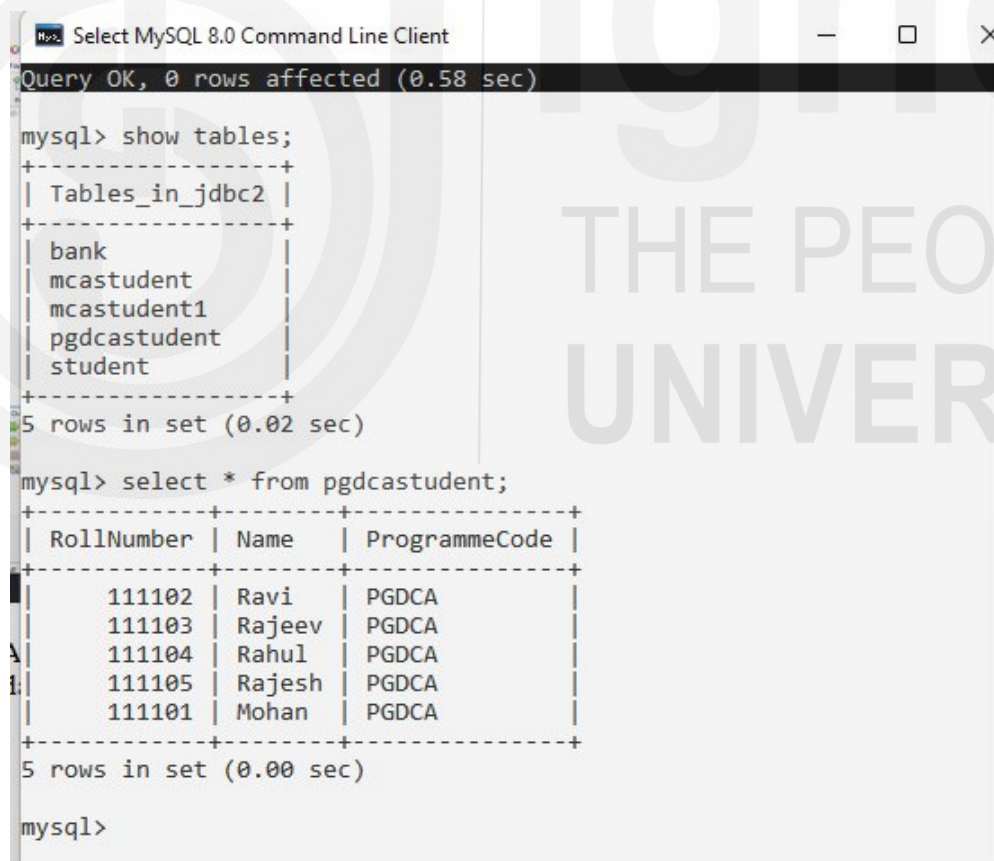


Figure 1.29: Screenshot of SQL commands execution on jdbc2 database

## 2.5 LIST OF LAB ASSIGNMENTS ( SESSION WISE)

In this section, session-wise problems/exercises are given for your lab.

### SESSION 1: JAVA BASICS- VARIABLES, EXPRESSIONS, OPERATORS, LOOPS

1. Write a Java program that displays Java primitive data types' default values.
2. Write a Java program that takes any two numbers as input and displays:
  - a. Average of the numbers
  - b. Largest number
  - c. Smallest number
3. Write a Java program that takes your name, address, enrolment number, and mobile number as input and displays it.
4. Write a Java program that takes a number between 1 and 20 as input and displays its multiplication table.
5. Write a Java program to calculate the roots of a given quadratic equation.
6. Write a Java program to display whether a given number is prime or not.

### SESSION 2: LOOPS AND ARRAYS

1. Write a Java program that demonstrates the use of the following:
  - a. for loop
  - b. while loop
2. Write a Java program that reverses a given array.
3. Write a Java program for performing the following operations on two 4X4 matrices.
  - a. Addition
  - b. Multiplication
4. Write a program in Java to find the average of marks obtained by a student in five papers as given below:

Paper:	Paper 1	Paper 2	Paper 3	Paper 4	Paper 5
Marks:	50	70	65	80	56

5. Write a Java program to find the "missing number" in a given integer array that contains numbers from 1 to 100 (e.g., the array has 99 elements and is missing one).
6. Write a Java program to find the largest and smallest elements in an integer array.
7. Write a Java program to transpose a 3x3 matrix (swap rows and columns).

### SESSION 3: CLASSES, OBJECTS , METHODS, ACCESS SPECIFIERS

1. Write a Java program to create a class Factorial with a method *display* that displays a given number's factorial.

2. Write a program to create a room class, the attributes of this class is Room\_no, Room\_type, Room\_AC\_Status, Room\_Rent. In this class, implement the member functions *setdata* and *displaydata* to insert data and display data respectively.
3. Write a Java program to calculate an employee's salary. Make necessary assumptions.
4. Write a Java program to create a class Rectangle with the data fields width, length, area and color. Define the methods *Set\_Length ()*, *Set\_Width ()*, and *Find\_Area ()*. Create two Rectangle objects and compare their areas.
5. Create a class Account to represent a Savings Bank account. Define two overloaded constructors, in which the first constructor is to be used for initialising the name of the account holder, the account number and the initial balance in the account. The second constructor is to be used to initialise the name of the account holder, the account number, the addresses, the mobile number, and the account balance. Define methods *Deposit ()*, *Withdraw ()*, and *Get\_Balance()* for performing respective operations on accounts. Make necessary assumptions as required. Create objects of the Account class and use them to demonstrate the working of the methods defined in the class.

#### SESSION 4: INHERITANCE, POLYMORPHISM

1. Write a program in Java to create a class Player. Inherit classes Cricket\_Player, Football\_Player and Hockey\_Player from class Player. Make necessary assumptions.
2. Write a Java program to create an abstract class Shape that has two data members of integer types to store the value of two sides of the shape and an empty method *FindArea()*. Derive three classes, Rectangle, Triangle and Circle, from the Shape class. Each of these classes contains only the method *printArea()* that prints the area of that shape. Make necessary assumptions.
3. Write a Java program to demonstrate the concept of method overriding. Use Online Shopping as an example.
4. Provide a code example of an abstract class Shape with an abstract method *draw()* and a concrete method *display()*.

#### SESSION 5: INTERFACE, PACKAGE AND EXCEPTIONS HANDLING

1. Create your own package, which provides a service to find whether a given number is odd or even and import this package in another class.
2. Create a package `com.utils.string` with a class `StringHelper` that has a static method `isPalindrome(String s)`. Import and use this method in another class.
3. Create an Interface having three methods: multiplication, division and addition. Create a class that overrides these methods. Make necessary assumptions.
4. Write a program in Java that implements the interface Student, which has two methods: *Display\_Grade* and *ShowAttendance* for *PG\_Students* and *UG\_Students*, where *PG\_Students* and *UG\_Students* are two different classes for Post Graduate and Under Graduate students, respectively.

5. Write a program in Java to display the name and roll number of students. Initialise respective array variables for ten students. Handle `ArrayIndexOutOfBoundsException`, to avoid illegal termination of the program.
6. Write a program to illustrate subclass exception precedence over base class.
7. Write a Java program to create custom exceptions in the process of validating IGNOU PGDCA student records. Make necessary assumptions.
8. Assume that two cars are running on the single track of a road. When both vehicles are going in the same direction, there is no problem. But when the cars are running in the opposite direction, there is a chance of a collision. To avoid collisions, write a Java program using exception handling. You are free to make necessary assumptions.

## SESSION 6: MULTITHREADING AND NUMBER & STRINGS

1. Write Java program to show the use of the `String` class and its operations:
  - i) Find string length.
  - ii) Reverse the contents of a string given on the console and convert the resultant string to upper case.
  - iii) Read a string from the console and append it to the given string.
2. Write a Java program to illustrate how the thread priorities are implemented. Imagine that the first thread has just begun to run, even before it has a chance to do anything. Now comes the higher-priority thread, which also wants to run. Note that the higher-priority thread must complete its work before the first thread starts.
3. Write a java program that creates five threads `T1`, `T2`, `T3`, `T4`, and `T5` with different priorities. Send two threads that have the highest priority in the sleep state. Check the thread's aliveness and mark which thread is long-lasting.
4. Write a Java program with two threads, one printing odd numbers and one printing even numbers, in the correct alternating sequence (1, 2, 3, 4, 5...) using `wait()` and `notify()`.
5. Write a Java program that properly implements the producer-consumer problem using the concept of inter-thread communication.
6. Write a Java Program to demonstrate the use of `String` and `StringBuilder`.
7. Write a program to convert an integer to a `String` using `Integer.toString()` and a `String` to an integer using `Integer.parseInt()`.

## SESSION 7: I/O AND JAVA API

1. Write a Java program that takes the name of a file as input from a user, reads the file's contents, and displays it on the console.
2. Write a Java program to copy a file into another file.
3. Write a Java program to count characters, words and lines of a given file.
4. Write a Java program to list all the files in a directory, including those present in all its subdirectories.



5. Write a program to demonstrate the `Iterator` interface by iterating over a `List` of integers and removing all even numbers.
6. Write a Java program to demonstrate the use of the `Map` API, specifically `HashMap`. The program should store and retrieve employee IDs and names.
7. Write a program to sort an `ArrayList` of custom `Employee` objects (with id, name, salary) based on their salary using the `Comparable` interface.

## SESSION 8: GUI PROGRAMMING

1. Write a Java program to simulate a traffic light. The program lets the user select one of the three lights: red, yellow or green. On choosing a button, an appropriate message with "Stop", "Ready" or "Go" should appear above the button's selected colour (note that initially, no message is shown).
2. Write a Java program using `Swing` to display an image (.jpg or .png) selected by the user via a `JFileChooser`.
3. Create a dummy page for email accounts using proper GUI components.
4. Write a Java program to create a simple calculator using proper GUI components. Handle any possible exceptions, including division by zero, in your program.

## SESSION 9: JDBC PROGRAMMING

1. Write a Java program to demonstrate connecting to a database. Use an appropriate database created in `MySQL` for demonstration.
2. Write a Java program to create a database table named `Account`, which has fields of a Saving Bank Account, and perform the following on this table:
  - a. insert records,
  - b. update records,
  - c. select records, and
  - c. delete records.
3. Write a program that iterates through a `ResultSet` from the `Account` table and prints the `account_holder_name` and `balance` for each
4. Define an interface `Database` with methods `connect()`, `disconnect()`, and `executeQuery(String query)`. Create `MySqlDatabase` and `OracleDatabase` classes that implement this interface.

## SESSION 10: JDBC PROGRAMMING

1. Write a Java program to manage student's records using `JDBC`. Use appropriate GUI components to create the user interface. Demonstrate uses of `commit` and `rollback` in your program. Make necessary assumptions.
2. Write a Java program to execute a batch of `SQL` statements.
3. Write a `JDBC` program that properly uses a `try-catch-finally` block to catch `SQLException` and ensure the database `Connection` is *always* closed.

---

## 2.6 REFERENCES/FURTHER READINGS

---

- Walter Savitch, “Java: An Introduction to Problem Solving and Programming”, Pearson Education, 2017.
- Download JDK: <https://www.oracle.com/in/java/technologies/javase-downloads.html>
- JDK Installation Document for Multiple Platforms:  
<https://docs.oracle.com/en/java/javase/16/install/overview-jdk-installation.html#GUID-8677A77F-231A-40F7-98B9-1FD0B48C346A>
- Java Environment PATH AND CLASSPATH Setup for the various platforms:  
<https://docs.oracle.com/javase/tutorial/essential/environment/paths.html>
- Download NetBeans IDE: <https://netbeans.apache.org/download/index.html>
- <https://www.javatpoint.com/how-to-set-classpath-in-java>
- Download MySQL: <https://www.mysql.com/downloads/>
- Download Java-MySQL Connector: <https://dev.mysql.com/downloads/connector/j/>

