



AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING



Department of Information Technology

Lab Manual

Object Oriented Programming (ITPCC407)

Second Year

Information Technology



Department of Information Technology

Vision

To be a leader in preparing technically competent and skillful IT Graduates to address the needs of industry and society.

Mission

- curricular, extracurricular and extension activities.
- ii. To promote research and professional activities through industry involvement and professional bodies
- iii. To instil professional ethics and lifelong learning skills with concern for the society.

Programme Educational Objectives

Graduates will

- i. Excel in diverse career paths with core professional skills.
- ii. Engage in multi domain research/professional activities.
- iii. Cater to the needs of society with IT solutions/applications.

Program Specific Outcomes

Graduates will be able to

- 1. Use database, networking and programming technologies for solving real life problems.
- 2. Develop applications in the field of computing, networking, security and analytics.

Program Outcomes

Graduates will be able to

- 1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. **[Engineering knowledge]**
- 2. Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. **[Problem analysis]**
- 3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. **[Design/development of solutions]**



Department of Information Technology

4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. **[Conduct investigations of complex problems]**
5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations. **[Modern tool usage]**
6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. **[The engineer and society]**
7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. **[Environment and sustainability]**
8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. **[Ethics]**
9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. **[Individual and team work]**
10. Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. **[Communication]**
11. Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. **[Project management and finance]**
12. Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. **[Life-long learning]**



Department of Information Technology

Second Year Information Technology (2023 Course) Object Oriented Programming Lab

Course Code:	ITPCC407	Credit:	2
Contact Hours:	4 Hrs./week(P)	Type of Course:	Practical
Examination Scheme	Term-work 25 Marks	Oral 25 Marks	

Pre-requisites:

- Programming and Problem Solving I
- Programming and Problem Solving II

Course assessment methods/tools:

Sr. No.	Course assessment methods/tools	External/ Internal	Marks
1.	Term Work	Internal	25
2.	Oral	External	25

Course Objectives

1	To implement object-oriented programming concepts
2	To handle exceptions using error handling
3	To make use of collections for implementing generics
4	To use file handling for given problem

Course Outcomes: Students will be able to

407.1	Study and installation of Tools for development and execution of Java Programs
407.2	Write program for a given problem using OOP concept
407.3	Use exception handling mechanism to manage exceptions
407.4	Make use of generic framework for given problems
407.5	Solve a database-oriented problem using file handling
407.6	Demonstrate and Present solution for given problems



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

List of Experiments

Sr. No.	Problem Statement	Page No.
1	Study of different modern tools and Installations of JDK and Java IDE like NetBeans or Eclipse.	7
2	Design a class for student entity and consider relevant abstract data. Accept and display the data for 5 objects using array of objects.	30
3	Design a class 'Complex' with data members for real and imaginary part. Provide default and Parameterized constructors. Write a program to perform arithmetic operations of two complex numbers.	35
4	Identify commonalities and differences between Publication, Book and Magazine classes. Title, Price, Copies are common instance variables and saleCopy is common method. The differences are, Bookclass has author and order Copies(). Magazine Class has orderQty, Currenttissue, recievetissue(). Write a program to find how many copies of the given books are ordered and display total sale of publication.	39
5	Design and Develop inheritance for a given case study, identify objects and relations and implement inheritance wherever applicable. Employee class with Emp_name, Emp_id, Address, Mail_id, and Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all inherited classes with 97% of BP as DA, 10% of Bp as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.	42
6	Design a base class shape with two double type values and member functions to input the data and compute_area() for calculating area of figure, Derive two classes' triangle and rectangle. Make compute_area() as abstract function and redefine this function in the derived class to suit their requirements. Write a program that accepts dimensions of triangle/ rectangle and display calculated area. Implement dynamic binding for given cases.	47
7	Design and develop a context for given case study and implement an interface for Vehicles. Consider the example of vehicles like bicycle, car and bike. All Vehicles have common functionalities such as Gear Change, Speed up and apply brakes. Make an interface and put all these functionalities. Bicycle, Bike, Car classes should be implemented for all these functionalities in their own class in their own way.	51
8	Implement a program to handle Arithmetic exception, Array Index Out of Bounds. The user enters two numbers Num1 and Num2. The division of Num1 and Num2 is displayed. If Num1 and Num2 were not integers, the program would throw a Number Format Exception. If Num2 were zero, the	56



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

	program would throw an Arithmetic Exception. Display the exception.	
9	Implement a generic program using any collections class to count the number of elements in a collection that have a specific property such as even numbers, odd number, prime number and palindromes.	59
10	Implement a program for maintaining a student records database using File Handling. Student has Student_id, name, Roll_no, Class, marks and address. Display the data for five students. a) Create Databases b) Display Database c) Clear Records d) Modify Records e) Search Records	62
11	Using all concepts of Object-Oriented programming develop a solution for any application contains following operations such as a) Creation of database b) Addition of data c) Deletion of data d) Updation of data e) Display of data	69
12	Virtual Lab Experiments: a) Using constructors for creating Objects: https://java-iitd.vlabs.ac.in/exp/constructors/index.html b) Understanding Inheritance in Java: https://java-iitd.vlabs.ac.in/exp/inheritance/index.html	77
13*	Design and develop mini project using Object-Oriented Programming.	

* - Mini Project of OOP (Write writeup based on project)



Department of Information Technology

Experiment No. 1

Aim:

To understand the installation process and features of modern Java Development Kit (JDK) and Integrated Development Environments (IDEs) such as NetBeans and Eclipse for efficient Java development.

Problem Statement:

Study of different modern tools and Installations of JDK and Java IDE like NetBeans or Eclipse.

Objectives:

- Gain knowledge about JDK installation and configuration.
- Explore the features and functionalities of NetBeans and Eclipse IDEs.
- Learn to set up a Java development environment for writing, compiling, and running Java programs.

Concepts:

- Java Development Kit (JDK)
- Integrated Development Environment (IDE)
- Debugging and Testing Tools
- Code Compilation and Execution

Theory:

1. **JDK (Java Development Kit):** JDK is a software development kit used to develop Java applications. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), and other necessary tools for development.
2. **NetBeans IDE:** NetBeans is an open-source IDE that supports multiple programming languages. It provides tools for editing, compiling, debugging, and deploying Java applications.
3. **Eclipse IDE:** Eclipse is another widely used IDE for Java development. It offers rich features such as code suggestions, refactoring, version control, and plugin support.

Installation of JDK

Install JDK on Microsoft Windows

Follow the below steps to install JDK on Windows environment. The below steps works on every Windows like **Windows 7, Windows 8, Windows 8.1, Windows 10, and Windows 11**. So, whatever Windows you are running in your system just go through the step to install Java Development Kit.

Step 1: Download and Install Java Development Kit (JDK)

The very first step is to download the **Oracle Java Development Kit (JDK)** from the Official Oracle Website. For that, Head over to the

<https://www.oracle.com/java/technologies/downloads/#jdk18-windows>.



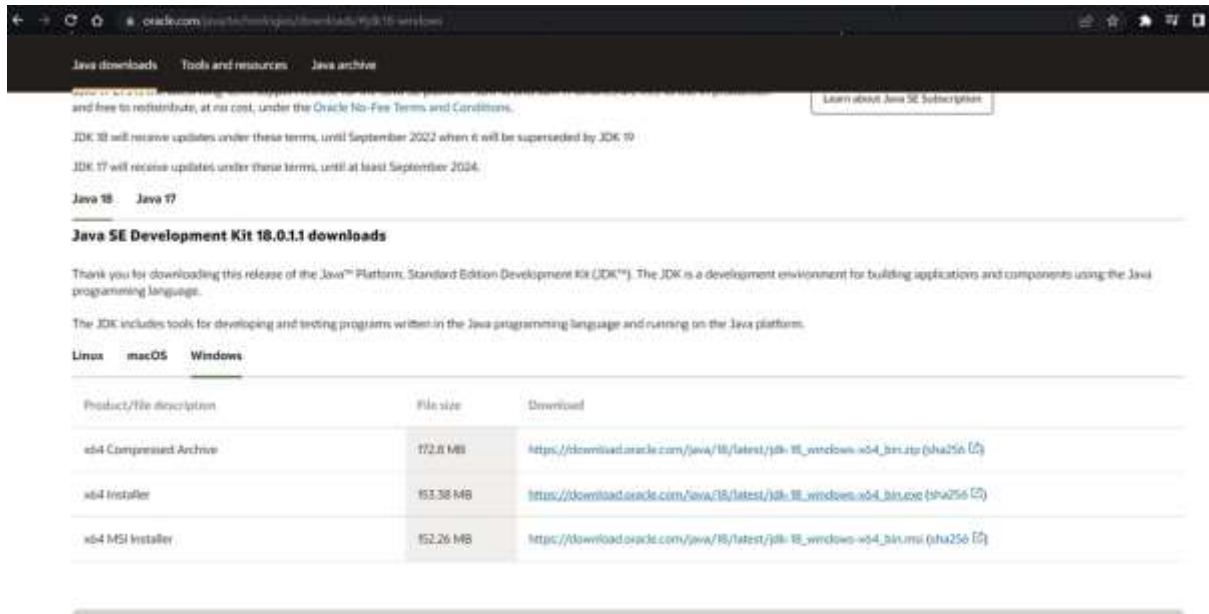
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



You need to identify your system specifications to choose the Product/file description. The website will contain the latest version for your corresponding system. For Windows, we'll be downloading the latest **x64 Installer of Java SE Development Kit 18**. After the download is complete, proceed to install the JDK by following the bootstrapped steps.



Step 2: Configure Environment Variables

After the installation is complete, we have to configure environment variables to notify the system about the directory in which JDK files are located. Proceed to **C:\Program Files\Java\jdk-{YOUR_JDK_VERSION}\bin** (replace {-} with your JDK version)



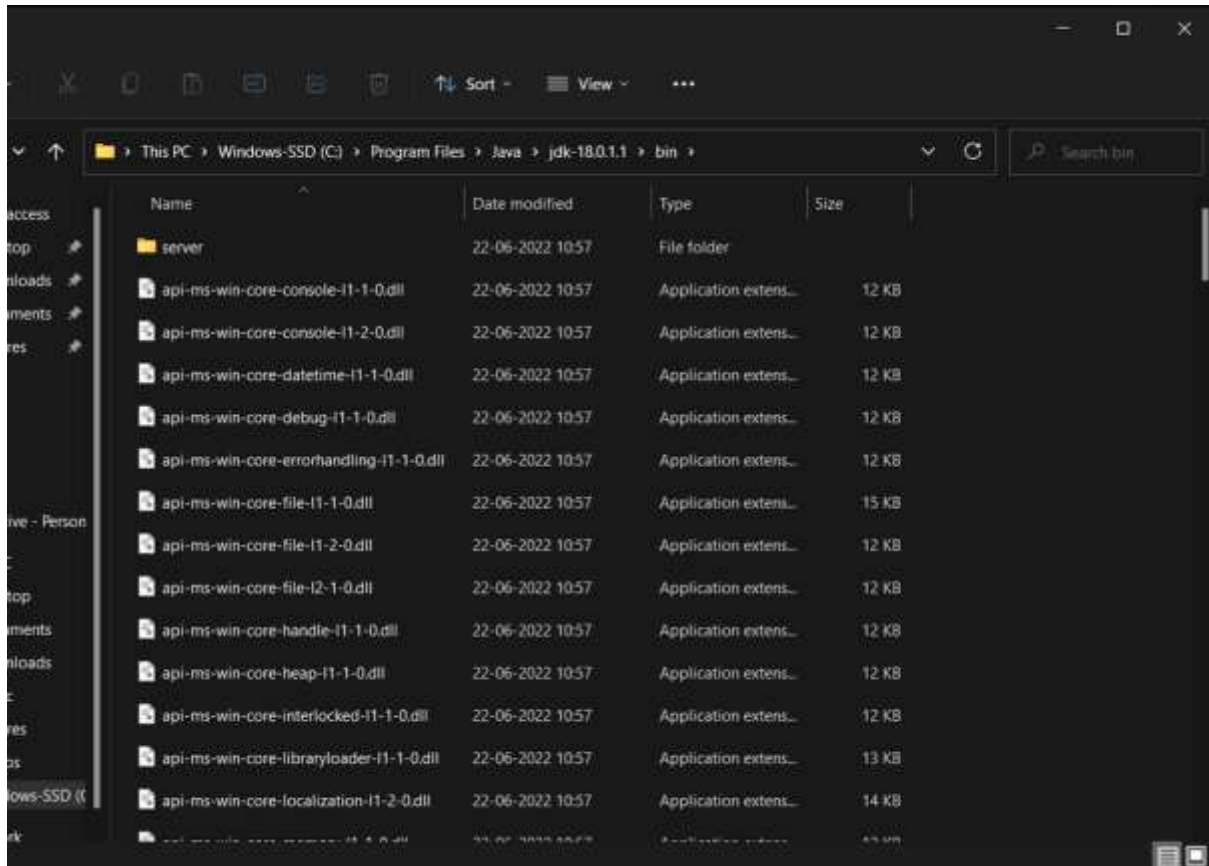
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



To set the Environment Variables, you need to search Environment Variables in the Task Bar and click on **“Edit the system environment variables”**.



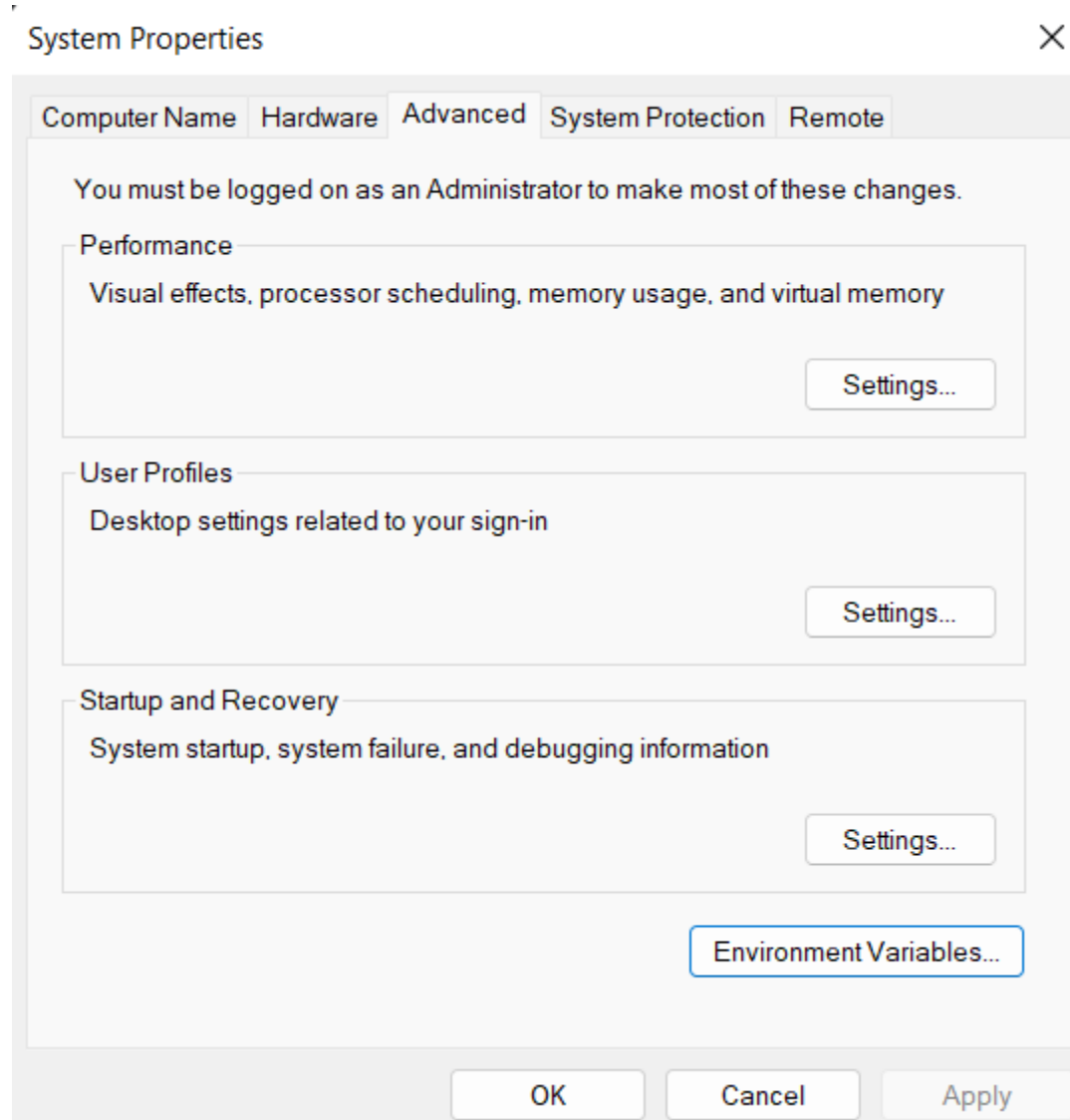
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



Under the **Advanced** section, Click on “**Environment Variables**”.



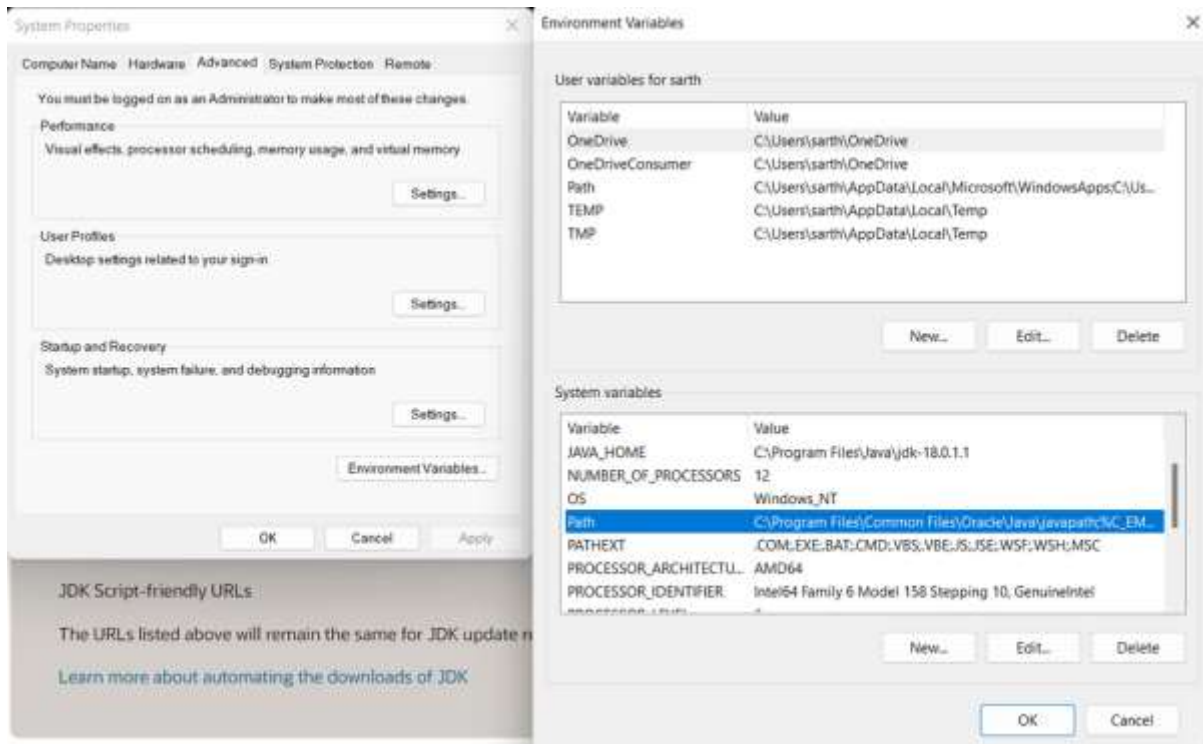
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

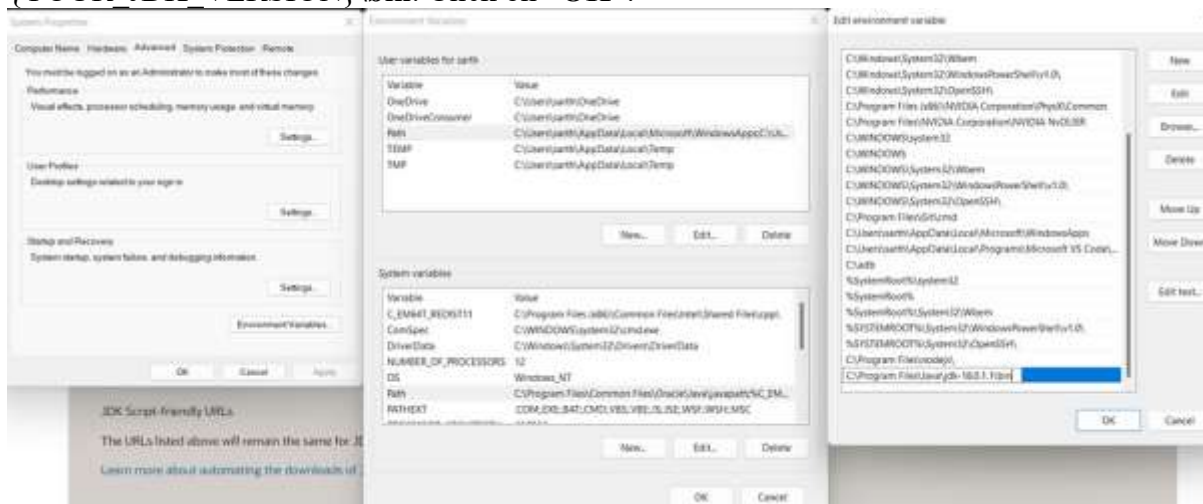
ADDING VALUE TO ENGINEERING



Department of Information Technology



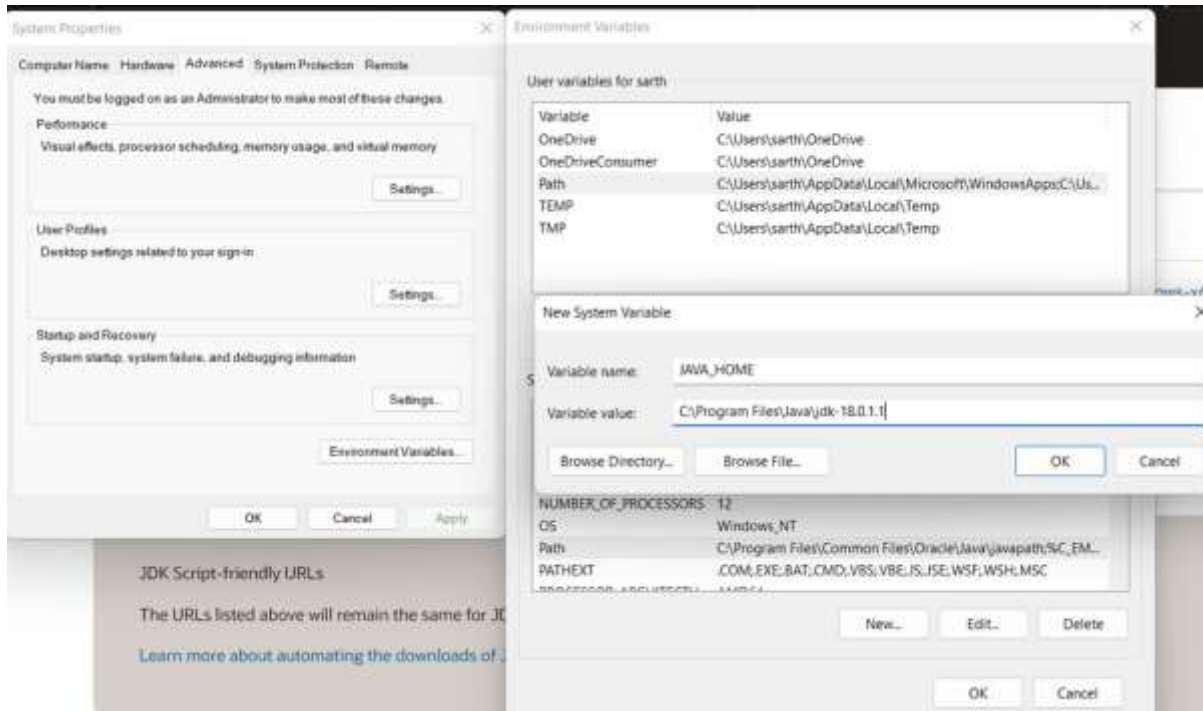
Under **System variables**, select the “**Path**” variable and click on “**Edit**”. Click on “**New**” then paste the Path Address i.e. **C:\Program Files\Java\jdk-{YOUR_JDK_VERSION}\bin**. Click on “**OK**”.



Now, in the **Environment Variables** dialogue, under **System variables**, click on “**New**” and then under **Variable name**: **JAVA_HOME** and **Variable value**: paste address i.e. **C:\Program Files\Java\jdk-{YOUR_JDK_VERSION}**. Click on **OK** => **OK** => **OK**.



Department of Information Technology



Step 3: Check the Java Version

Open **Command Prompt** and enter the following commands

`java -version`

`javac -version`

```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sarth>java -version
java version "18.0.1.1" 2022-04-22
Java(TM) SE Runtime Environment (build 18.0.1.1+2-6)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.1.1+2-6, mixed mode, sharing)

C:\Users\sarth>javac -version
javac 18.0.1.1

C:\Users\sarth>
```

Install JDK on Linux

Follow the steps below to install the Java Development Kit (JDK) on a Linux environment. These instructions are applicable to various Linux distributions such as **Ubuntu**, **Fedora**, and **CentOS**.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Note: For this guide we have used Kali Linux distributions.

Step 1: Download and Install Oracle Java Development Kit (JDK)

The very first step is to download the **Oracle Java Development Kit (JDK)** from the Official Oracle Website. For that, Head over to the

<https://www.oracle.com/java/technologies/downloads/#jdk18-linux>.

The screenshot shows the Oracle website's download page for Java SE Development Kit 18.0.1.1. The page is titled "Java SE Development Kit 18.0.1.1 downloads" and includes a thank you message and a link to the JDK download page. Below the message, there are tabs for "Linux", "macOS", and "Windows". The "Linux" tab is selected, and a table lists the available download options for Linux.

Product/file description	File size	Download
Arm 64 Compressed Archive	172.69 MB	https://download.oracle.com/java/18/latest/jdk-18_linux-aarch64_bin.tar.gz (sha256 [C])
Arm 64 RPM Package	154.17 MB	https://download.oracle.com/java/18/latest/jdk-18_linux-aarch64_bin.rpm (sha256 [C])
x64 Compressed Archive	173.83 MB	https://download.oracle.com/java/18/latest/jdk-18_linux-x64_bin.tar.gz (sha256 [C])
x64 Debian Package	149.24 MB	https://download.oracle.com/java/18/latest/jdk-18_linux-x64_bin.deb (sha256 [C])
x64 RPM Package	155.75 MB	https://download.oracle.com/java/18/latest/jdk-18_linux-x64_bin.rpm (sha256 [C])

You need to identify your system specifications to choose the Product/file description. The website will contain the latest version for your corresponding system. For Linux, we'll be downloading the latest **x64 Debian Package of Java SE Development Kit 18**. To install the downloaded JDK File using terminal, Open terminal and change your directory to downloads by using the command

```
$ cd downloads
```

To list files and packages present in the directory, Type

```
$ ls
```

The screenshot shows a terminal window with the following commands and output:

```
root@kali: ~/Downloads
File Edit View Search Terminal Help
root@kali:~# cd Downloads/
root@kali:~/Downloads# ls
jdk-18.0.1.1_linux-x64_bin.deb
root@kali:~/Downloads#
```




AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Now we use **Debian Package Manager** to configure our downloaded file for installation by typing

```
$ sudo dpkg -i jdk-{YOUR_JDK_VERSION}      (replace {-} with your version)
```

Enter your password as we have run an elevated prompt, i.e. the **sudo** or **superuser** **do** command, which is a quick way to grant specific users permission to perform specific system commands at the system's root (most powerful) level.

```
ritikshrivas@kali-linux: ~/Downloads
(ritikshrivas@kali-linux)-[~]
$ cd Downloads
(ritikshrivas@kali-linux)-[~/Downloads]
$ sudo dpkg -i jdk-18 linux-x64 bin.deb
[sudo] password for ritikshrivas: 
```

Now, Type the following commands to proceed with the installation

```
$ sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-{YOUR_JDK_VERSION}/bin/java 1
```

```
$ sudo update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-{YOUR_JDK_VERSION}/bin/javac 1
```

Step 2: Check the Java Version

Open the **terminal** and enter the following commands

```
$ java --version
```

```
$ javac --version
```




AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
ritikshrivas@kali-linux: ~/Downloads
default-java  java-1.11.0-openjdk-amd64  java-11-openjdk-amd64  jdk-18

(ritikshrivas@kali-linux)-[~/Downloads]
$ sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-18/bin/java 1

(ritikshrivas@kali-linux)-[~/Downloads]
$ sudo update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-18/bin/javac 1

(ritikshrivas@kali-linux)-[~/Downloads]
$ java --version
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
openjdk 11.0.14.1 2022-02-08
OpenJDK Runtime Environment (build 11.0.14.1+1-post-Debian-1)
OpenJDK 64-Bit Server VM (build 11.0.14.1+1-post-Debian-1, mixed mode, sharing)

(ritikshrivas@kali-linux)-[~/Downloads]
$ javac --version
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
javac 18.0.1.1

(ritikshrivas@kali-linux)-[~/Downloads]
$
```

Step 3: Configure JAVA_HOME Environment Variable

After the installation is complete, we have to configure environment variables to notify the system about the directory in which jdk files are located. To find the Location of the JDK Files, run this command

```
$ sudo update-alternatives --config java
```

and copy the File Location.

In order to set the environment variable, you have to edit the environment file using this command

```
$ sudo gedit /etc/environment
```

Proceed to add **JAVA_HOME="/usr/lib/jvm/jdk-{YOUR_JDK_VERSION}"**

Proceed to save and close the file.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
*environment
/etc

1 PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games"
2 JAVA_HOME=""
```

Now we have to refresh the environment file by using this command

```
$ SOURCE /etc/environment
```

And echo the JAVA_HOME Path

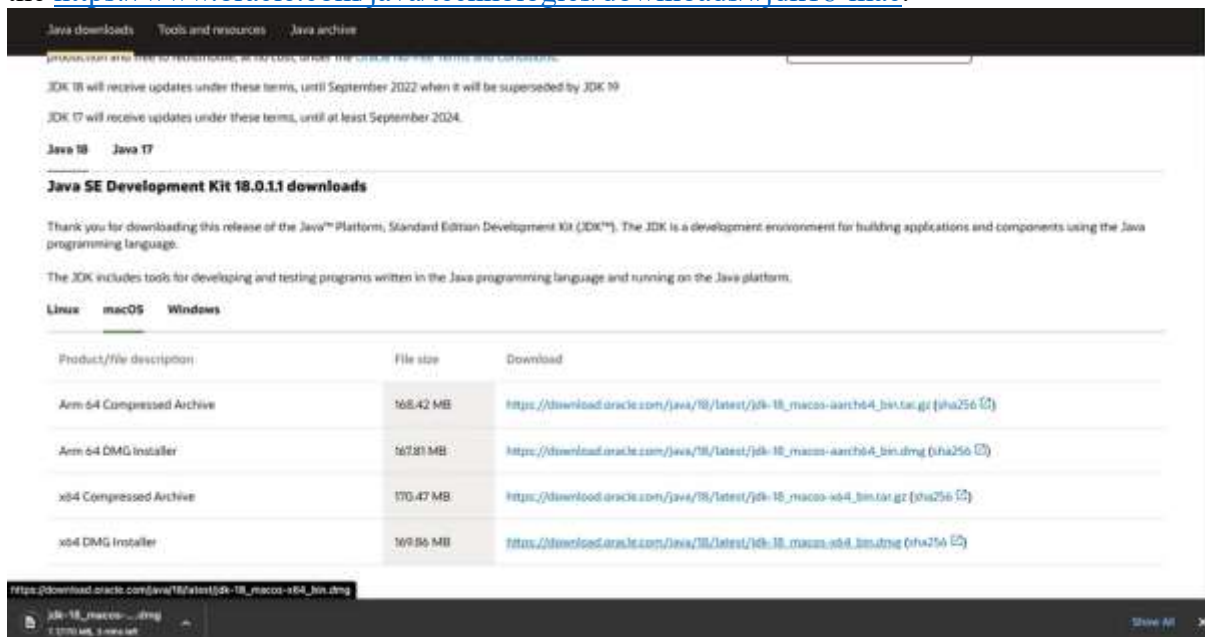
```
$ echo $JAVA_HOME
```

Install JDK on macOS

Follow the steps below to install the Java Development Kit (JDK) on macOS. These instructions are compatible with various versions of macOS, including **Mojave, Catalina, Big Sur, Monterey, Ventura, Sonoma** the latest macOS **Sequoia**.

Step 1: Download and Install Oracle Java Development Kit (JDK)

The very first step is to download the **Oracle Java Development Kit (JDK)** from the Official Oracle Website. For that, Head over to the <https://www.oracle.com/java/technologies/downloads/#jdk18-mac>.



You need to identify your system specifications to choose the Product/file description. The website will contain the latest version for your corresponding system. For Mac, we'll be



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

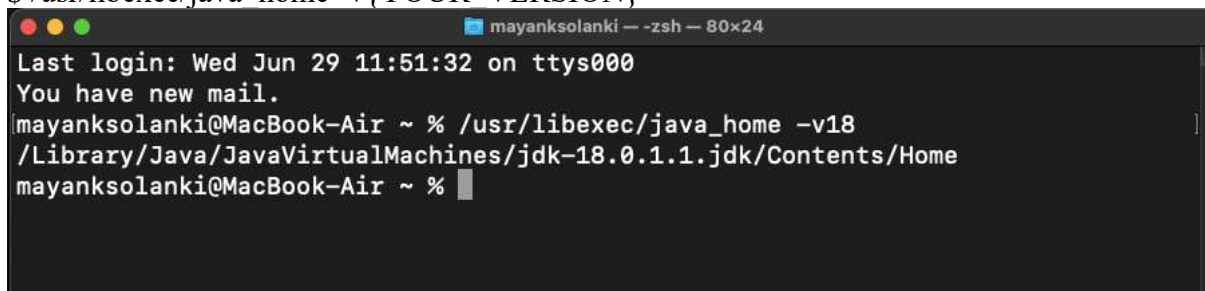
downloading the latest **x64 DMG Installer of Java SE Development Kit 18**. After the download is complete, proceed to install the JDK by following the bootstrapped steps.



Step 2: Configure environment variables

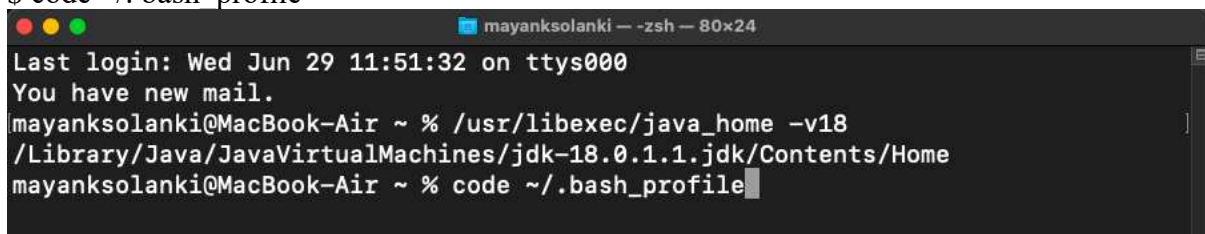
Now to configure, we have to open the terminal and pass the following commands. To find the Location of the JAVA_HOME, run this command

```
$ /usr/libexec/java_home -v{YOUR_VERSION}
```



We have to set this output as our JAVA_HOME Environment Variable. You can use any command or code editor to edit the file, here we are using VS Code

```
$ code ~/.bash_profile
```



At the very bottom, we have to export the path we obtained earlier i.e.

```
$ export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-  
{YOUR_VERSION}.jdk/Contents/Home
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



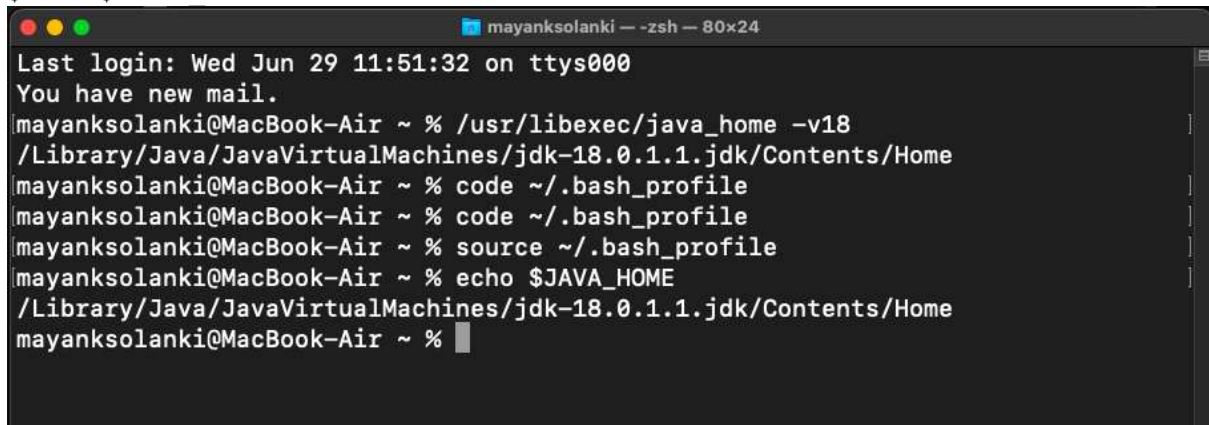
```
.bash_profile
1 export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-18.0.1.1.jdk/Contents/Home
```

Now we have to refresh the environment file by using this command

```
$ source ~/.bash_profile
```

And echo the JAVA_HOME variable

```
$ echo $JAVA_HOME
```



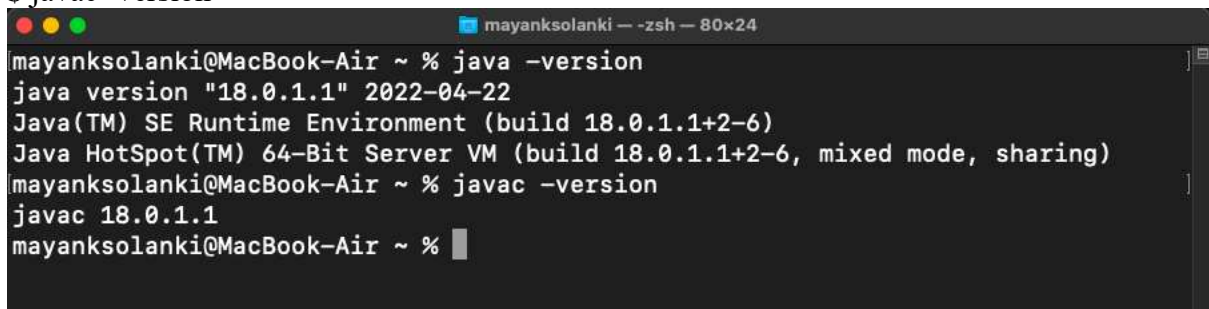
```
mayanksolanki ~ % /usr/libexec/java_home -v18
/Library/Java/JavaVirtualMachines/jdk-18.0.1.1.jdk/Contents/Home
mayanksolanki@MacBook-Air ~ % code ~/.bash_profile
mayanksolanki@MacBook-Air ~ % source ~/.bash_profile
mayanksolanki@MacBook-Air ~ % echo $JAVA_HOME
/Library/Java/JavaVirtualMachines/jdk-18.0.1.1.jdk/Contents/Home
mayanksolanki@MacBook-Air ~ %
```

Step 3: Check the Java Version

In the **terminal**, enter the following commands

```
$ java -version
```

```
$ javac -version
```



```
mayanksolanki@MacBook-Air ~ % java -version
java version "18.0.1.1" 2022-04-22
Java(TM) SE Runtime Environment (build 18.0.1.1+2-6)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.1.1+2-6, mixed mode, sharing)
mayanksolanki@MacBook-Air ~ % javac -version
javac 18.0.1.1
mayanksolanki@MacBook-Air ~ %
```




AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Installation of Eclipse

Linux

Method 1: Install Eclipse IDE using the Installer Package.

Step 1: Downloading installer package

Go to Eclipse's official website and download the installer package for [Linux](#).

The Eclipse Installer 2024-03 R now includes a JRE for macOS, Windows and Linux.

Try the Eclipse Installer 2024-03 R

The easiest way to install and update your Eclipse Development Environment.

📥 1,506,140 Installer Downloads
📥 1,026,585 Package Downloads and Updates

Download

macOS [x86_64](#) | [AArch64](#)
Windows [x86_64](#)
Linux [x86_64](#) | [AArch64](#)

Step 2: Extract installer package.

After download finishes. Navigate to Downloads directory using **cd command**.

```
cd Downloads
```

Now extract content from tar.gz file using [tar -xvf command](#).

```
tar -xvf eclipse*
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
gfg@debian:~$ cd Downloads/  
gfg@debian:~/Downloads$ ls  
eclipse-inst-jre-linux64.tar.gz  
gfg@debian:~/Downloads$ tar -xvf eclipse*  
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.creationtime'  
eclipse-installer/  
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.creationtime'  
eclipse-installer/p2/  
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.creationtime'  
eclipse-installer/p2/org.eclipse.equinox.p2.engine/  
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.creationtime'  
eclipse-installer/p2/org.eclipse.equinox.p2.engine/profileRegistry/  
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.creationtime'  
eclipse-installer/p2/org.eclipse.equinox.p2.engine/profileRegistry/Def-
```

Step 3: Install Eclipse using installer

After extracting file, navigate to extracted directory using **cd** command.

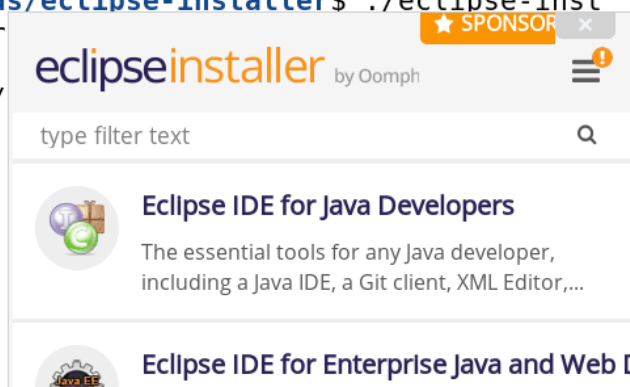
```
cd eclipse-installer
```

Now start installer with following command

```
./eclipse-inst
```

After this new installer window will open.

```
gfg@debian:~/Downloads$ cd eclipse-installer/  
gfg@debian:~/Downloads/eclipse-installer$ ./eclipse-inst  
SLF4J(W): No SLF4J pr  
SLF4J(W): Defaulting  
SLF4J(W): See https://
```



Select [Eclipse IDE](#) for Java Developers option.



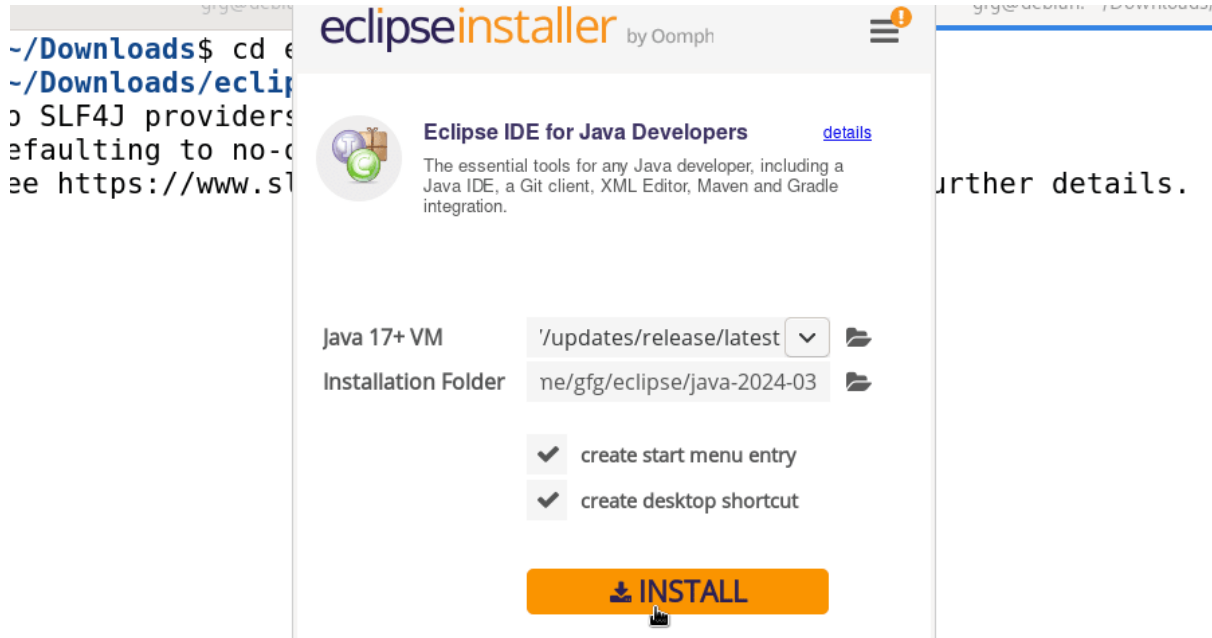
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

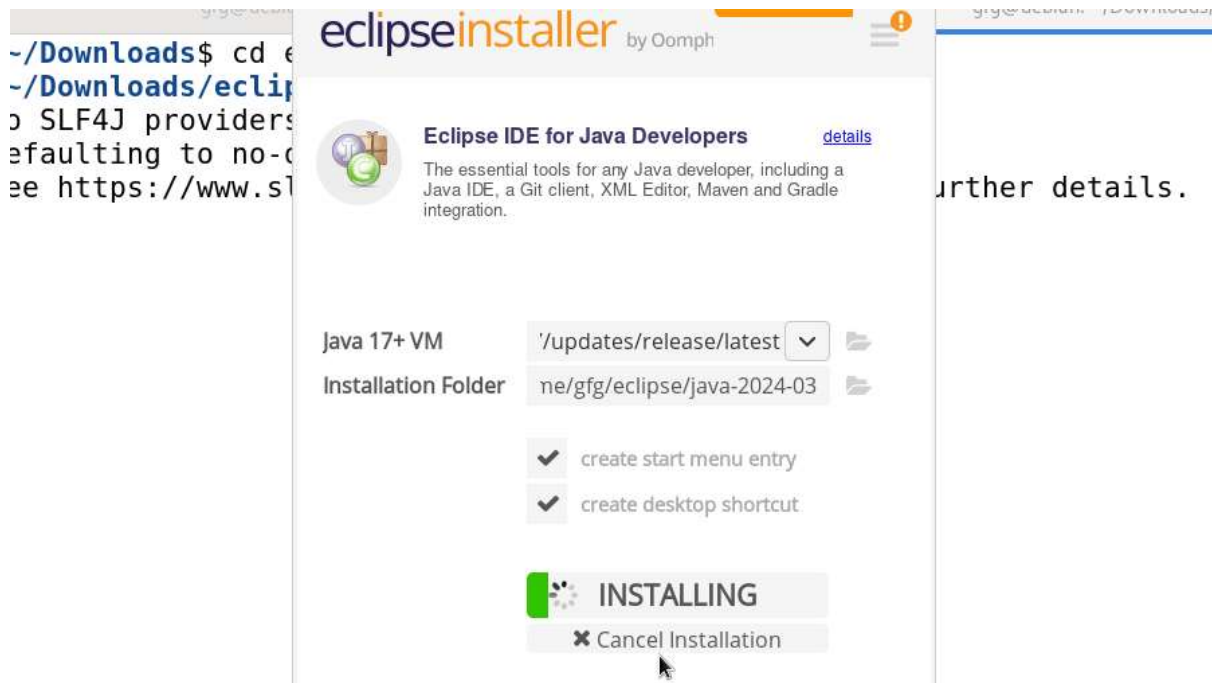
ADDING VALUE TO ENGINEERING



Department of Information Technology



After clicking on install with default options, installation will start.



Step 4: Launching Eclipse

We can launch eclipse ide from application menu as follows.



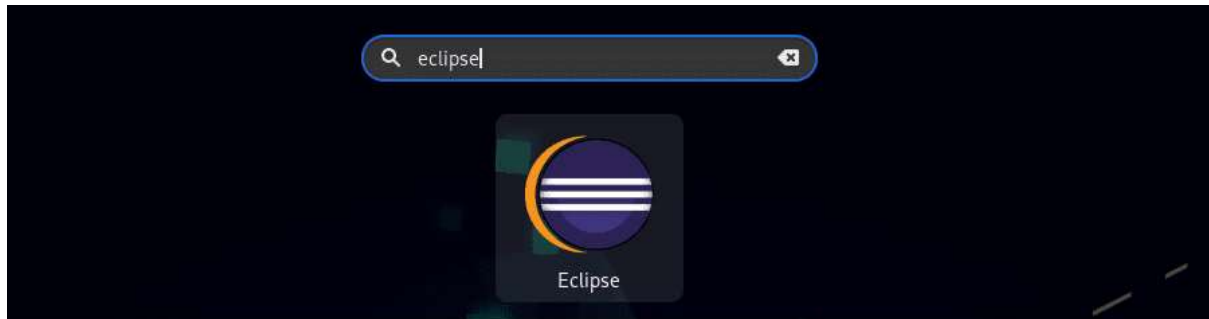
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



Method 2 : Using Snap Store.

Step 1 : Update System

Update system to latest packages using apt update and apt upgrade command.

```
sudo apt update && sudo apt upgrade
```

```
gfg@debian:~$ sudo apt update && sudo apt upgrade
Ign:1 cdrom://[Debian GNU/Linux 11.6.0 _Bullseye_ - Official amd64 DVD Binary-1
20221217-10:40] bullseye InRelease
Err:2 cdrom://[Debian GNU/Linux 11.6.0 _Bullseye_ - Official amd64 DVD Binary-1
20221217-10:40] bullseye Release
Please use apt-cdrom to make this CD-ROM recognized by APT. apt-get update can
not be used to add new CD-ROMs
Hit:3 http://security.debian.org/debian-security bullseye-security InRelease
Reading package lists... Done
E: The repository 'cdrom://[Debian GNU/Linux 11.6.0 _Bullseye_ - Official amd64
DVD Binary-1 20221217-10:40] bullseye Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disa
```

Step 2 : Install Snap store

We will install snap store using **apt install** command.

```
sudo apt install snapd
```

```
gfg@debian:~$ sudo apt install snapd -y
[sudo] password for gfg:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Step 3 : Enable Snap store

After installation, we will start services for snap package manager using systemctl enable and systemctl start command.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

sudo systemctl enable snapd

sudo systemctl start snapd

```
gfg@debian:~$ sudo systemctl enable snapd
Created symlink /etc/systemd/system/multi-user.target.wants/systemd/system/snapd.service.
gfg@debian:~$ sudo systemctl start snapd
gfg@debian:~$
```

Step 4 : Install Eclipse IDE

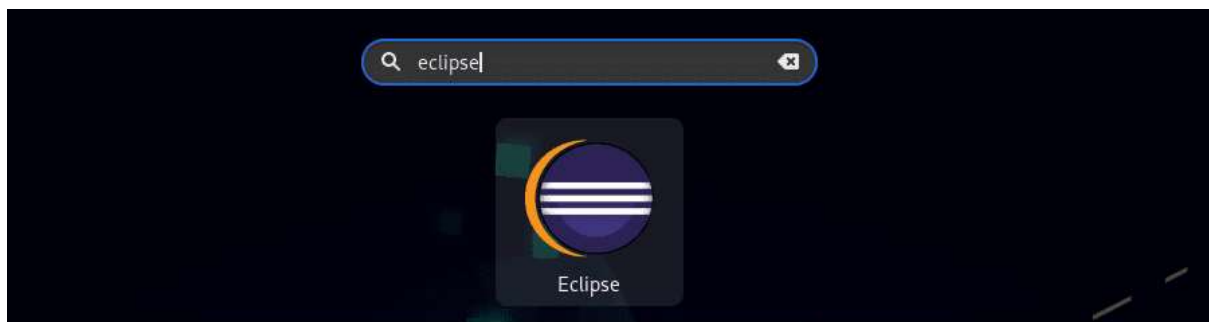
Once, snap package manager is setup, install Eclipse IDE using **snap install** command.

sudo snap install eclipse --classic

```
gfg@debian:~$ sudo snap install eclipse --classic
2024-05-31T22:37:22-04:00 INFO Waiting for automatic refresh of snap "eclipse" (87) from channel "stable"
```

Step 5: Launching Eclipse IDE

We can launch Eclipse IDE from Application menu as follows



Windows:

Steps to Download and install Eclipse on Windows :

Now let's look at the step by step process to Download and install Eclipse on Windows:

Step 1: In the first step, Open your browser and navigate to this [URL](#).



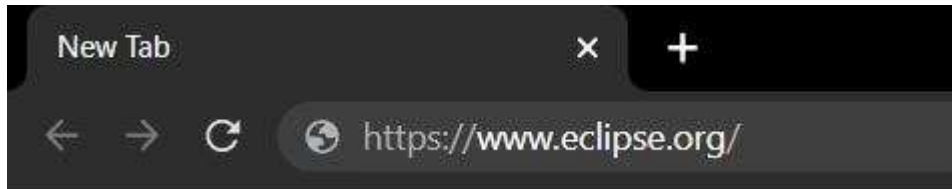
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



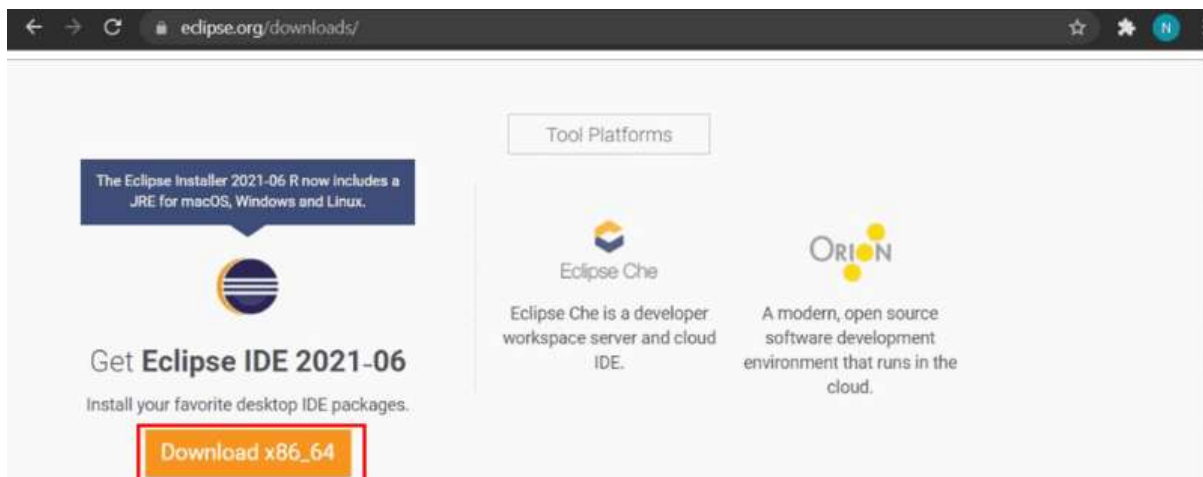
Department of Information Technology



Step 2: Then, click on the “**Download**” button to download Eclipse IDE.



Step 3: Now, click on the “**Download x86_64**” button.



Step 4: Then click on the “**Download**” button. After clicking on the **download** button the .exe file for the eclipse will be downloaded.



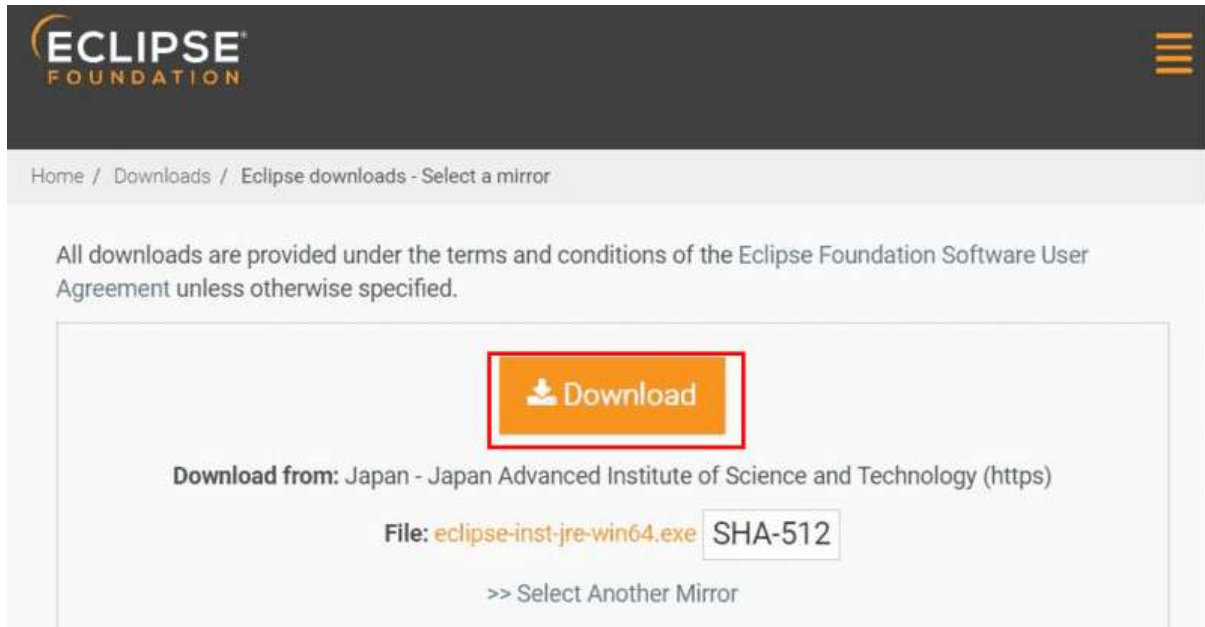
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

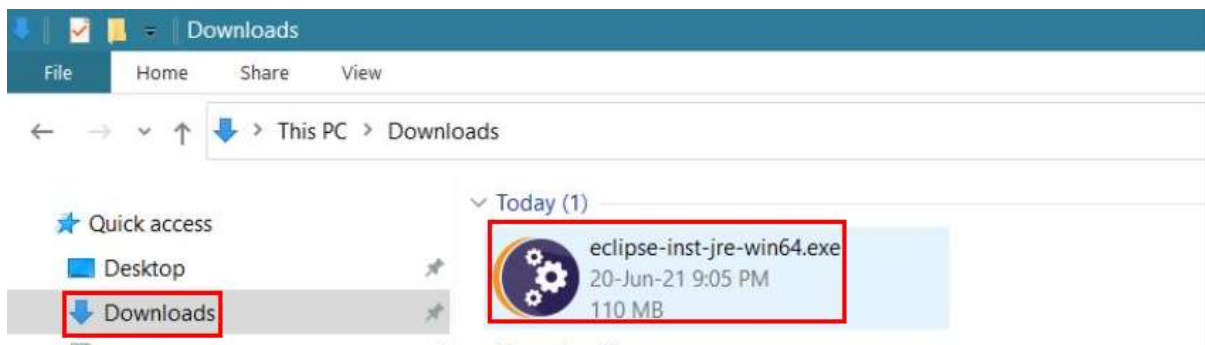
ADDING VALUE TO ENGINEERING



Department of Information Technology



Step 5: Now go to File Explorer and click on “**Downloads**” after that click on the “*eclipse-inst-jre-win64.exe*” file for installing Eclipse IDE.



Step 6: Then, click on “Eclipse IDE for Java Developers”.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



Step 7: Then, click on the “**Install**” button.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



Step 8: Now click on “Create a new Java project”.



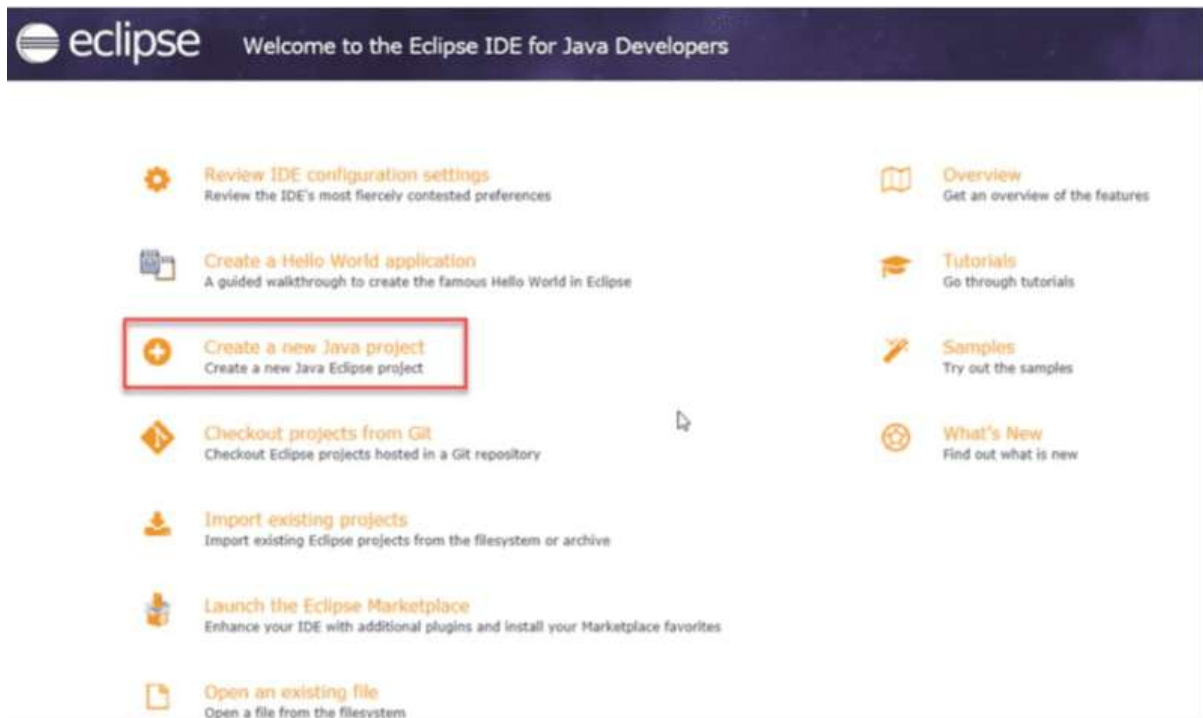
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



Now, you're prepared to make new Java initiatives the usage of eclipse IDE and the display screen will appear like this :



Conclusion:

By installing JDK and IDEs such as NetBeans and Eclipse, developers can efficiently create, compile, and debug Java applications. This setup streamlines the software development process and enhances productivity.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Viva Questions:

1. What is the role of JDK in Java development?
2. How do you set environment variables for JDK?
3. Compare NetBeans and Eclipse IDEs.
4. What are the key features of Eclipse that make it popular for Java development?
5. How can you check if JDK is correctly installed on your system?



Department of Information Technology

Experiment – 2

Aim:

To design a class representing a student entity with relevant abstract data and manage multiple objects using arrays.

Problem Statement:

Design a class for student entity and consider relevant abstract data. Accept and display the data for 5 objects using array of objects.

Objectives:

- Develop object-oriented programming skills by designing a class for student entities.
- Implement and manipulate arrays of objects in Java.
- Enhance problem-solving and coding skills using Java.

Concepts:

- Object-Oriented Programming (OOP) principles
- Classes and Objects
- Constructors and Methods
- Arrays of Objects

Theory:

1. **Classes and Objects:** A class is a blueprint for creating objects. It defines attributes and behaviors of an object. Objects are instances of a class.
2. **Arrays of Objects:** Arrays can store multiple objects of the same type, allowing batch processing and management of entities.

Algorithms:

1. Design a class Student with relevant fields like name, roll number, and grade.
2. Implement constructors and methods for input and display.
3. Create an array to store 5 student objects.
4. Accept data for 5 students and display it.
5. Handle invalid inputs gracefully with exception handling

Code:

INPUT

```
import java.util.Scanner;
```

```
class Student {  
    // Attributes  
    private int rollNumber;  
    private String name;  
    private int age;  
    private double marks;  
  
    // Constructor  
    public Student(int rollNumber, String name, int age, double marks) {
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
this.rollNumber = rollNumber;
this.name = name;
this.age = age;
this.marks = marks;
}

// Method to display student details
public void displayStudentDetails() {
    System.out.println("Roll Number: " + rollNumber);
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Marks: " + marks);
    System.out.println("-----");
}
}

public class StudentManagement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Student[] students = new Student[5]; // Array of Student objects

        // Accepting details for 5 students
        for (int i = 0; i < 5; i++) {
            System.out.println("Enter details for Student " + (i + 1) + ":");
            System.out.print("Roll Number: ");
            int rollNumber = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("Age: ");
            int age = scanner.nextInt();
            System.out.print("Marks: ");
            double marks = scanner.nextDouble();

            // Creating Student object and storing it in the array
            students[i] = new Student(rollNumber, name, age, marks);
        }

        // Displaying details of all students
        System.out.println("\nDisplaying Student Details:");
        for (Student student : students) {
            student.displayStudentDetails();
        }
    }
}
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
        scanner.close();  
    }  
}
```

OUTPUT

Enter details for Student 1:

Roll Number: 101

Name: Alice

Age: 20

Marks: 85.5

Enter details for Student 2:

Roll Number: 102

Name: Bob

Age: 21

Marks: 90.0

Enter details for Student 3:

Roll Number: 103

Name: Charlie

Age: 19

Marks: 78.5

Enter details for Student 4:

Roll Number: 104

Name: David

Age: 22

Marks: 88.0

Enter details for Student 5:

Roll Number: 105

Name: Eva

Age: 20

Marks: 91.5

Displaying Student Details:

Roll Number: 101

Name: Alice

Age: 20

Marks: 85.5

Roll Number: 102

Name: Bob



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Age: 21
Marks: 90.0

Roll Number: 103
Name: Charlie
Age: 19
Marks: 78.5

Roll Number: 104
Name: David
Age: 22
Marks: 88.0

Roll Number: 105
Name: Eva
Age: 20
Marks: 91.5

Explanation:

1. **Class Structure:** The program consists of two classes: Student for defining student attributes and methods, and StudentManagement for handling user input and managing multiple students.
2. **Attributes:** The Student class has four private attributes—rollNumber, name, age, and marks—which store the student's details.
3. **Constructor:** The parameterized constructor of the Student class initializes the attributes with the provided values when an object is created.
4. **Display Method:** The displayStudentDetails() method in the Student class prints the student's roll number, name, age, and marks in a structured format.
5. **Array of Objects:** The StudentManagement class creates an array of Student objects to store the details of five students dynamically.
6. **User Input Handling:** The program uses a Scanner to accept details (roll number, name, age, and marks) for five students. The scanner.nextLine() method is used to handle newline character issues when reading strings after numeric inputs.
7. **Object Creation and Storage:** For each student, a Student object is instantiated with user input and stored in the array.
8. **Displaying Student Details:** After collecting data, the program iterates over the array, calling displayStudentDetails() for each student to print their details.
9. **Resource Management:** The scanner.close() statement ensures proper closure of the Scanner object to prevent resource leaks.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Conclusion:

By designing a Student class and managing multiple objects, fundamental concepts of OOP are reinforced. This project demonstrates the effective use of arrays, constructors, and exception handling in Java programming.

Viva Questions:

1. Explain the purpose of constructors in Java.
2. What are the benefits of using arrays of objects?
3. How does the Scanner class work in Java?
4. How would you extend this program to handle more students dynamically?
5. Why is exception handling important in Java programs?



Department of Information Technology

Experiment – 3

Aim:

To design and implement a class 'Complex' to represent complex numbers and perform arithmetic operations such as addition, subtraction, and multiplication.

Problem Statement:

Design a class 'Complex' with data members for real and imaginary part. Provide default and Parameterized constructors. Write a program to perform arithmetic operations of two complex numbers.

Objectives:

- Understand the concept of complex numbers and their arithmetic operations.
- Learn how to design classes in Java with data members and constructors.
- Develop skills in operator overloading and method implementation.
- Implement and test arithmetic operations on complex numbers.

Concepts:

- Class and Object Design
- Constructor Overloading
- Method Overloading
- Arithmetic Operations

Theory:

Complex numbers have two parts: a real part and an imaginary part. A complex number is represented as: $z = a + bi$ Where **a** is the real part and **b** is the imaginary part.

Operations on complex numbers:

1. **Addition:** $(a + bi) + (c + di) = (a + c) + (b + d)i$
2. **Subtraction:** $(a + bi) - (c + di) = (a - c) + (b - d)i$
3. **Multiplication:** $(a + bi)(c + di) = (ac - bd) + (ad + bc)i$

Algorithms:

1. Start the program.
2. Create a class 'Complex' with data members for real and imaginary parts.
3. Implement a default constructor to initialize values to zero.
4. Implement a parameterized constructor to initialize real and imaginary parts to given values.
5. Implement methods for addition, subtraction, and multiplication.
6. Create instances of the Complex class to perform operations.
7. Display the result.
8. End the program.

Code:

INPUT

```
import java.util.Scanner;
```

```
class Complex {
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
private double real;  
private double imag;
```

```
// Default constructor
```

```
Complex() {  
    this.real = 0;  
    this.imag = 0;  
}
```

```
// Parameterized constructor
```

```
Complex(double real, double imag) {  
    this.real = real;  
    this.imag = imag;  
}
```

```
// Addition of two complex numbers
```

```
Complex add(Complex c) {  
    return new Complex(this.real + c.real, this.imag + c.imag);  
}
```

```
// Subtraction of two complex numbers
```

```
Complex subtract(Complex c) {  
    return new Complex(this.real - c.real, this.imag - c.imag);  
}
```

```
// Multiplication of two complex numbers
```

```
Complex multiply(Complex c) {  
    double realPart = this.real * c.real - this.imag * c.imag;  
    double imagPart = this.real * c.imag + this.imag * c.real;  
    return new Complex(realPart, imagPart);  
}
```

```
// Display method
```

```
void display() {  
    System.out.println(this.real + " + " + this.imag + "i");  
}  
}
```

```
public class Main {
```

```
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);
```

```
        System.out.print("Enter real and imaginary part of first complex number: ");
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
double r1 = sc.nextDouble();
double i1 = sc.nextDouble();

System.out.print("Enter real and imaginary part of second complex number: ");
double r2 = sc.nextDouble();
double i2 = sc.nextDouble();

Complex c1 = new Complex(r1, i1);
Complex c2 = new Complex(r2, i2);

Complex sum = c1.add(c2);
Complex difference = c1.subtract(c2);
Complex product = c1.multiply(c2);

System.out.println("\nResults:");
System.out.print("Addition: ");
sum.display();
System.out.print("Subtraction: ");
difference.display();
System.out.print("Multiplication: ");
product.display();

sc.close();
}
}
```

OUTPUT

Enter real and imaginary part of first complex number: 3.5 2.5
Enter real and imaginary part of second complex number: 1.5 4.0

Results:

Addition: $5.0 + 6.5i$

Subtraction: $2.0 + -1.5i$

Multiplication: $-5.25 + 17.75i$

Explanation:

1. **Class Structure:** The program consists of two classes: 'Complex' for complex number operations and 'Main' to handle user input and perform operations.
2. **Constructors:** The default constructor initializes the complex number to $0 + 0i$, while the parameterized constructor sets the real and imaginary parts.
3. **Methods:** Three methods (add, subtract, multiply) implement the respective arithmetic operations and return new Complex objects.
4. **Display Method:** The display method prints complex numbers in the form $a + bi$.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

5. **User Input:** The main method collects user input and creates two complex number objects.
6. **Results:** The results of addition, subtraction, and multiplication are calculated and displayed.

Conclusion:

The program demonstrates how to design and implement a class to perform arithmetic operations on complex numbers. The use of constructors and methods promotes reusability and modularity, which are essential in object-oriented programming.

Viva Questions:

1. What are complex numbers, and how are they represented in Java?
2. Why do we use constructors in Java?
3. How does method overloading work in Java?
4. Explain the significance of returning new objects in arithmetic operations.
5. How can you extend the class to include division of complex numbers?



Department of Information Technology

Experiment – 4

Aim:

To implement a Java program that demonstrates class inheritance and method overriding by creating a hierarchy of Publication, Book, and Magazine classes to calculate total sales.

Problem Statement:

Identify commonalities and differences between Publication, Book and Magazine classes. Title, Price, Copies are common instance variables and saleCopy is common method. The differences are, Bookclass has author and order Copies(). Magazine Class has orderQty, Currenttissue, recievedtissue(). Write a program to find how many copies of the given books are ordered and display total sale of publication.

Objectives:

- Understand class inheritance and polymorphism.
- Implement and manage instance variables and methods.
- Calculate total sales and order quantities for books and magazines.

Concepts:

- Object-Oriented Programming (OOP)
- Inheritance
- Method Overriding
- Encapsulation

Theory:

Inheritance allows the creation of new classes based on existing ones. Here, the Publication class is a superclass with common attributes like title, price, and copies. The Book and Magazine classes inherit from Publication and add their unique attributes and methods.

Algorithms:

1. Define a superclass Publication with common attributes and methods.
2. Create subclasses Book and Magazine inheriting from Publication.
3. Implement unique attributes and methods for Book and Magazine.
4. Calculate and display total sales and order copies.

Code:

INPUT

```
class Publication {  
    String title;  
    double price;  
    int copies;  
  
    Publication(String title, double price, int copies) {  
        this.title = title;  
        this.price = price;  
        this.copies = copies;  
    }  
}
```



Department of Information Technology

```
}

void saleCopy() {
    System.out.println("Total sale: $" + (price * copies));
}

}

class Book extends Publication {
    String author;

    Book(String title, double price, int copies, String author) {
        super(title, price, copies);
        this.author = author;
    }

    void orderCopies(int qty) {
        copies += qty;
        System.out.println(qty + " copies of book ordered.");
    }
}

class Magazine extends Publication {
    int orderQty;
    int currentIssue;

    Magazine(String title, double price, int copies, int orderQty, int currentIssue) {
        super(title, price, copies);
        this.orderQty = orderQty;
        this.currentIssue = currentIssue;
    }

    void receiveIssue() {
        copies += orderQty;
        System.out.println("Received " + orderQty + " copies of issue " + currentIssue);
    }
}

public class Main {
    public static void main(String[] args) {
        Book b1 = new Book("Java Programming", 50.0, 10, "John Doe");
        Magazine m1 = new Magazine("Tech Today", 10.0, 20, 15, 5);

        // Ordering and receiving copies
    }
}
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
b1.orderCopies(5);  
m1.receiveIssue();  
  
// Display sales  
b1.saleCopy();  
m1.saleCopy();  
}  
}
```

OUTPUT

5 copies of book ordered.
Received 15 copies of issue 5
Total sale: \$750.0
Total sale: \$350.0

Explanation:

1. The Publication class contains shared attributes and a method to calculate sales.
2. The Book class adds an author attribute and a method to order copies.
3. The Magazine class includes order quantity and current issue, with a method to receive new issues.
4. The main method creates instances of Book and Magazine, demonstrating inheritance and method usage.

Conclusion:

Effectively demonstrated class inheritance, method overriding, and encapsulation by simulating a publication sale system for books and magazines highlighting key OOP principles.

Viva Questions:

1. What is inheritance in Java?
2. How does method overriding work?
3. Explain the difference between superclass and subclass.
4. How can you ensure encapsulation in Java?
5. What is the role of the 'super' keyword in Java?
6. How do constructors work in inheritance hierarchies?



Department of Information Technology

Experiment – 5

Aim:

To design and develop a Java program to implement inheritance for an employee management system that generates pay slips for different employee categories such as Programmer, Assistant Professor, Associate Professor, and Professor.

Problem Statement:

Design and Develop inheritance for a given case study, identify objects and relations and implement inheritance wherever applicable. Employee class with Emp_name, Emp_id, Address, Mail_id, and Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all inherited classes with 97% of BP as DA, 10% of Bp as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.

Objectives:

- Understand the concept of inheritance in Object-Oriented Programming (OOP).
- Identify real-world entities and relationships for effective class design.
- Implement hierarchical inheritance using Java.
- Calculate gross and net salaries for different types of employees.
- Generate pay slips dynamically.

Concepts:

- Object Oriented Programming
- Superclass
- Subclass
- Modifiers

Theory:

Inheritance is a mechanism in Java through which one class can acquire properties (fields) and behaviours (methods) of another class. It promotes code reusability and hierarchical class structures.

- **Superclass (Base class):** Employee
- **Subclass (Derived classes):** Programmer, Assistant Professor, Associate Professor, Professor
- **Modifiers:** Inheritance can be public, private, or protected depending on accessibility needs.

Algorithms:

1. Define the superclass Employee with common attributes.
2. Create subclasses Programmer, AssistantProfessor, AssociateProfessor, and Professor that extend the Employee class.
3. Add a method to calculate pay (Basic Pay, DA, HRA, PF, staff club fund) in each subclass.
4. Implement a method to generate pay slips for each employee category.



Department of Information Technology

5. Display gross and net salaries for each employee.

Code:

INPUT

```
class Employee {
    String empName, empId, address, mailId;
    long mobileNo;

    Employee(String empName, String empId, String address, String mailId, long mobileNo) {
        this.empName = empName;
        this.empId = empId;
        this.address = address;
        this.mailId = mailId;
        this.mobileNo = mobileNo;
    }

    void displayDetails() {
        System.out.println("Employee ID: " + empId);
        System.out.println("Employee Name: " + empName);
        System.out.println("Address: " + address);
        System.out.println("Mail ID: " + mailId);
        System.out.println("Mobile No: " + mobileNo);
    }
}

class Programmer extends Employee {
    double basicPay;

    Programmer(String empName, String empId, String address, String mailId, long
mobileNo, double basicPay) {
        super(empName, empId, address, mailId, mobileNo);
        this.basicPay = basicPay;
    }

    void generatePaySlip() {
        double da = 0.97 * basicPay;
        double hra = 0.10 * basicPay;
        double pf = 0.12 * basicPay;
        double staffClubFund = 0.001 * basicPay;
        double grossSalary = basicPay + da + hra;
        double netSalary = grossSalary - pf - staffClubFund;

        displayDetails();
    }
}
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
System.out.println("Basic Pay: " + basicPay);
System.out.println("Gross Salary: " + grossSalary);
System.out.println("Net Salary: " + netSalary);
    }
}

class AssistantProfessor extends Programmer {
    AssistantProfessor(String empName, String empId, String address, String mailId, long
mobileNo, double basicPay) {
        super(empName, empId, address, mailId, mobileNo, basicPay);
    }
}

class AssociateProfessor extends Programmer {
    AssociateProfessor(String empName, String empId, String address, String mailId, long
mobileNo, double basicPay) {
        super(empName, empId, address, mailId, mobileNo, basicPay);
    }
}

class Professor extends Programmer {
    Professor(String empName, String empId, String address, String mailId, long mobileNo,
double basicPay) {
        super(empName, empId, address, mailId, mobileNo, basicPay);
    }
}

public class PaySlipGenerator {
    public static void main(String[] args) {
        Programmer prog = new Programmer("John Doe", "P101", "123 Street",
"john.doe@mail.com", 9876543210L, 50000);
        prog.generatePaySlip();

        AssistantProfessor ap = new AssistantProfessor("Jane Smith", "AP102", "456 Lane",
"jane.smith@mail.com", 9876543221L, 60000);
        ap.generatePaySlip();

        AssociateProfessor aap = new AssociateProfessor("Emily White", "AAP103", "789
Blvd", "emily.white@mail.com", 9876543232L, 70000);
        aap.generatePaySlip();

        Professor prof = new Professor("Michael Brown", "PR104", "101 Hwy",
"michael.brown@mail.com", 9876543243L, 80000);
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
    prof.generatePaySlip();  
  }  
}
```

OUTPUT

Employee ID: P101
Employee Name: John Doe
Address: 123 Street
Mail ID: john.doe@mail.com
Mobile No: 9876543210
Basic Pay: 50000.0
Gross Salary: 157000.0
Net Salary: 150400.0

Employee ID: AP102
Employee Name: Jane Smith
Address: 456 Lane
Mail ID: jane.smith@mail.com
Mobile No: 9876543221
Basic Pay: 60000.0
Gross Salary: 188400.0
Net Salary: 180600.0

Employee ID: AAP103
Employee Name: Emily White
Address: 789 Blvd
Mail ID: emily.white@mail.com
Mobile No: 9876543232
Basic Pay: 70000.0
Gross Salary: 219800.0
Net Salary: 210800.0

Employee ID: PR104
Employee Name: Michael Brown
Address: 101 Hwy
Mail ID: michael.brown@mail.com
Mobile No: 9876543243
Basic Pay: 80000.0
Gross Salary: 251200.0
Net Salary: 241000.0

Explanation:

- **Employee Class:** Contains general information such as name, ID, address, mail, and mobile number.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

- **Programmer Class:** Inherits from Employee and calculates salary based on basic pay.
- **Assistant Professor, Associate Professor, and Professor Classes:** Extend from Programmer and reuse its logic for salary calculation.
- **Main Class:** Instantiates different employee objects and generates pay slips for each.

Conclusion:

This project demonstrates the power of inheritance in Java by creating a hierarchical structure of employee categories. The code achieves efficient pay slip generation and promotes reusability and extensibility through inheritance.

Viva Questions:

1. What is inheritance in Java?
2. How does the super keyword work?
3. Why is inheritance important in OOP?
4. How is method overriding used in this project?
5. What are the advantages of using hierarchical inheritance in employee management systems?



Department of Information Technology

Experiment – 6

Aim:

To design a Java program that uses inheritance and dynamic binding to compute the area of different shapes (Triangle and Rectangle) by implementing an abstract base class Shape and deriving classes Triangle and Rectangle.

Problem Statement:

Design a base class shape with two double type values and member functions to input the data and compute_area() for calculating area of figure, Derive two classes' triangle and rectangle. Make compute_area() as abstract function and redefine this function in the derived class to suit their requirements. Write a program that accepts dimensions of triangle/ rectangle and display calculated area. Implement dynamic binding for given cases.

Objectives:

- To understand and implement the concept of abstract classes and dynamic binding in Java.
- To develop a program that calculates the area of different shapes using polymorphism.
- To apply object-oriented principles such as inheritance and method overriding.

Concepts:

- Object Oriented Programming
- Inheritance
- Abstract Class
- Polymorphism
- Method Overriding

Theory:

1. **Inheritance:** Inheritance allows a class to acquire the properties and methods of another class. The derived class (child) inherits from the base class (parent).
2. **Abstract Class:** An abstract class in Java cannot be instantiated directly and must have at least one abstract method. Abstract methods are defined without an implementation and must be implemented in derived classes.
3. **Dynamic Binding (Polymorphism):** Dynamic binding is the process of linking a method call to the method implementation at runtime. This allows overriding methods in derived classes to be invoked through base class references.
4. **Method Overriding:** This occurs when a subclass provides a specific implementation for a method that is already defined in the superclass.

Algorithms:

1. Create an abstract base class Shape with two double type values and methods to input data and compute the area.
2. Declare an abstract method compute_area() in the base class.
3. Derive two classes Triangle and Rectangle from Shape.
4. Implement the compute_area() method in both derived classes to calculate the area specific to the shape.



Department of Information Technology

5. Accept dimensions of the triangle and rectangle from the user.
6. Use dynamic binding to compute and display the area of the chosen shape.

Code:

INPUT

```
import java.util.Scanner;

// Abstract base class
abstract class Shape {
    double dimension1;
    double dimension2;

    // Method to input dimensions
    void inputDimensions() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first dimension: ");
        dimension1 = scanner.nextDouble();
        System.out.print("Enter second dimension: ");
        dimension2 = scanner.nextDouble();
    }

    // Abstract method to compute area
    abstract void compute_area();
}

// Triangle class
class Triangle extends Shape {
    @Override
    void compute_area() {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}

// Rectangle class
class Rectangle extends Shape {
    @Override
    void compute_area() {
        double area = dimension1 * dimension2;
        System.out.println("Area of Rectangle: " + area);
    }
}
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
// Main class to test the program
public class ShapeAreaCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Choose shape (1 for Triangle, 2 for Rectangle): ");
        int choice = scanner.nextInt();

        Shape shape;
        if (choice == 1) {
            shape = new Triangle();
        } else {
            shape = new Rectangle();
        }

        shape.inputDimensions();
        shape.compute_area();
    }
}
```

OUTPUT

```
Choose shape (1 for Triangle, 2 for Rectangle):
1
Enter first dimension: 10
Enter second dimension: 5
Area of Triangle: 25.0
Choose shape (1 for Triangle, 2 for Rectangle):
2
Enter first dimension: 8
Enter second dimension: 4
Area of Rectangle: 32.0
```

Explanation:

1. **Abstract Class Shape:** This class declares two double values and has methods to input data. It also contains the abstract method `compute_area()`, ensuring that derived classes implement this method.
2. **Derived Classes Triangle and Rectangle:** These classes extend Shape and override the `compute_area()` method to compute areas specific to each shape.
3. **Main Method:**
 - The program asks the user to choose between triangle and rectangle.
 - Based on user input, the appropriate derived class is instantiated.
 - The `compute_area()` method is invoked dynamically based on the object type.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Conclusion:

This program demonstrates the use of abstract classes, method overriding, and dynamic binding in Java. By designing a base class Shape and deriving Triangle and Rectangle, the program efficiently computes areas of different shapes using polymorphism.

Viva Questions:

1. What is an abstract class, and why is it used?
2. Explain the concept of dynamic binding with an example.
3. How does method overriding differ from method overloading?
4. Why can't abstract classes be instantiated directly?
5. How does polymorphism enhance code reusability and flexibility?
6. What is the purpose of the super keyword in Java?
7. How would you modify this program to add another shape, such as a circle?



Department of Information Technology

Experiment – 7

Aim:

To design and develop a Java program to implement an interface for vehicles that includes common functionalities such as changing gears, speeding up, and applying brakes. The interface will be implemented by different vehicle classes such as Bicycle, Bike, and Car, demonstrating polymorphism and abstraction.

Problem Statement:

Design and develop a context for given case study and implement an interface for Vehicles. Consider the example of vehicles like bicycle, car and bike. All Vehicles have common functionalities such as Gear Change, Speed up and apply brakes. Make an interface and put all these functionalities. Bicycle, Bike, Car classes should be implemented for all these functionalities in their own class in their own way.

Objectives:

1. Understand the concept of interfaces in Java.
2. Implement polymorphism by designing a common interface for different types of vehicles.
3. Demonstrate abstraction by defining vehicle functionalities without implementing them directly in the interface.
4. Develop individual classes for different types of vehicles that implement the interface.
5. Gain practical experience in Java OOP principles like inheritance, polymorphism, and abstraction.

Concepts:

- **Interface:** An interface in Java is a blueprint of a class. It contains abstract methods that are implemented by classes.
- **Polymorphism:** The ability to use a single interface to represent different types of objects.
- **Abstraction:** Hiding implementation details and showing only essential features of an object.
- **Encapsulation:** Wrapping code and data together into a single unit.
- **Inheritance:** A mechanism by which one class acquires the properties and behaviors of another class.

Theory:

In object-oriented programming, interfaces are used to achieve complete abstraction. Java allows the creation of interfaces that define methods but do not provide their implementation. Classes that implement the interface must provide the specific behavior for each method. This promotes code reusability, scalability, and maintainability.

In this case study, an interface named **Vehicle** is created to declare methods for changing gears, speeding up, and applying brakes. Three classes, **Bicycle**, **Bike**, and **Car**, implement this interface and provide their own specific implementations of these methods.

Algorithms:

1. Define an interface called **Vehicle**.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

2. Declare methods **changeGear**, **speedUp**, and **applyBrakes** within the interface.
3. Create classes **Bicycle**, **Bike**, and **Car** that implement the **Vehicle** interface.
4. Implement the methods of the **Vehicle** interface in each class, providing functionality specific to that vehicle.
5. Create a main class to demonstrate polymorphism by creating objects of Bicycle, Bike, and Car and invoking the methods through the interface reference.

Code:

INPUT

```
interface Vehicle {
    void changeGear(int newGear);
    void speedUp(int increment);
    void applyBrakes(int decrement);
}

class Bicycle implements Vehicle {
    int speed = 0;
    int gear = 1;

    public void changeGear(int newGear) {
        gear = newGear;
        System.out.println("Bicycle gear changed to: " + gear);
    }

    public void speedUp(int increment) {
        speed += increment;
        System.out.println("Bicycle speed increased to: " + speed);
    }

    public void applyBrakes(int decrement) {
        speed -= decrement;
        System.out.println("Bicycle speed decreased to: " + speed);
    }
}

class Bike implements Vehicle {
    int speed = 0;
    int gear = 1;

    public void changeGear(int newGear) {
        gear = newGear;
        System.out.println("Bike gear changed to: " + gear);
    }
}
```




AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
public void speedUp(int increment) {
    speed += increment;
    System.out.println("Bike speed increased to: " + speed);
}

public void applyBrakes(int decrement) {
    speed -= decrement;
    System.out.println("Bike speed decreased to: " + speed);
}

class Car implements Vehicle {
    int speed = 0;
    int gear = 1;

    public void changeGear(int newGear) {
        gear = newGear;
        System.out.println("Car gear changed to: " + gear);
    }

    public void speedUp(int increment) {
        speed += increment;
        System.out.println("Car speed increased to: " + speed);
    }

    public void applyBrakes(int decrement) {
        speed -= decrement;
        System.out.println("Car speed decreased to: " + speed);
    }
}

public class TestVehicle {
    public static void main(String[] args) {
        Vehicle bicycle = new Bicycle();
        Vehicle bike = new Bike();
        Vehicle car = new Car();

        bicycle.changeGear(2);
        bicycle.speedUp(10);
        bicycle.applyBrakes(3);

        bike.changeGear(3);
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
bike.speedUp(30);  
bike.applyBrakes(10);  
  
car.changeGear(4);  
car.speedUp(60);  
car.applyBrakes(20);  
}  
}
```

OUTPUT

Bicycle gear changed to: 2
Bicycle speed increased to: 10
Bicycle speed decreased to: 7

Bike gear changed to: 3
Bike speed increased to: 30
Bike speed decreased to: 20

Car gear changed to: 4
Car speed increased to: 60
Car speed decreased to: 40

Explanation:

- **Interface Vehicle:** Declares three methods that every vehicle must implement.
- **Bicycle, Bike, Car Classes:** Implement the Vehicle interface and provide specific behavior for each method.
- **TestVehicle Class:** Demonstrates polymorphism by creating objects of different vehicle classes and invoking their methods using the Vehicle interface reference.

Conclusion:

By implementing this Java program, we successfully demonstrated the use of interfaces to design a common blueprint for different types of vehicles. This program highlights the importance of polymorphism and abstraction in object-oriented programming, allowing flexible and reusable code.

Viva Questions:

1. What is an interface in Java, and how is it different from an abstract class?
2. How does polymorphism work in this program?
3. Why did we use the Vehicle interface instead of directly creating methods in each class?
4. What are the advantages of using interfaces in Java?
5. Can an interface have a method with implementation in Java?
6. Explain the difference between overriding and overloading.
7. How is encapsulation demonstrated in this program?
8. What would happen if one of the classes did not implement all methods of the Vehicle interface?



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

9. Can a class implement multiple interfaces in Java? How?
10. How can this program be extended to include more vehicle types?



Department of Information Technology

Experiment – 8

Aim:

To implement a Java program that handles `ArithmeticException`, `ArrayIndexOutOfBoundsException`, and `NumberFormatException` by accepting two numbers from the user and performing division.

Problem Statement:

Implement a program to handle Arithmetic exception, Array Index Out of Bounds. The user enters two numbers Num1 and Num2. The division of Num1 and Num2 is displayed. If Num1 and Num2 were not integers, the program would throw a Number Format Exception. If Num2 were zero, the program would throw an Arithmetic Exception. Display the exception.

Objectives:

- To understand exception handling in Java.
- To learn how to catch and handle runtime exceptions such as `ArithmeticException`, `ArrayIndexOutOfBoundsException`, and `NumberFormatException`.
- To develop robust and error-free code that gracefully handles user input errors.

Concepts:

- **Exception Handling:** Mechanism to handle runtime errors, allowing the program to continue executing without crashing.
- **try-catch block:** Used to catch exceptions that might occur during program execution.
- **ArithmeticException:** Thrown when an exceptional arithmetic condition has occurred, such as division by zero.
- **ArrayIndexOutOfBoundsException:** Thrown to indicate that an array has been accessed with an illegal index.
- **NumberFormatException:** Thrown when attempting to convert a string into a numeric type but the string does not have the appropriate format.

Theory:

Exception handling in Java is managed by five keywords: `try`, `catch`, `finally`, `throw`, and `throws`. The `try` block contains code that might throw an exception, while the `catch` block handles the exception. Multiple `catch` blocks can be used to handle different types of exceptions. The `finally` block (optional) is used to execute code after the `try-catch` block, regardless of whether an exception is thrown or not.

Algorithms:

1. Start the program.
2. Prompt the user to enter two numbers.
3. Try to parse the inputs as integers.
4. Perform division and display the result.
5. Catch and handle the following exceptions:
 - `ArithmeticException` (if division by zero occurs)
 - `NumberFormatException` (if input is not a valid integer)
 - `ArrayIndexOutOfBoundsException` (if array access is out of bounds)



Department of Information Technology

6. Display appropriate error messages for each exception.
7. End the program.

Code:

INPUT

```
import java.util.Scanner;
```

```
public class ExceptionHandlingDemo {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        int[] numbers = new int[2];  
        try {  
            System.out.print("Enter first number (Num1): ");  
            numbers[0] = Integer.parseInt(scanner.nextLine());  
  
            System.out.print("Enter second number (Num2): ");  
            numbers[1] = Integer.parseInt(scanner.nextLine());  
  
            int result = numbers[0] / numbers[1];  
            System.out.println("Result of division: " + result);  
        } catch (ArithmeticException e) {  
            System.out.println("Error: Division by zero is not allowed.");  
        } catch (NumberFormatException e) {  
            System.out.println("Error: Invalid input. Please enter valid integers.");  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("Error: Array index out of bounds.");  
        } finally {  
            scanner.close();  
        }  
    }  
}
```

OUTPUT

```
Enter first number (Num1): 10  
Enter second number (Num2): 2  
Result of division: 5  
Enter first number (Num1): 10  
Enter second number (Num2): 0  
Error: Division by zero is not allowed.  
Enter first number (Num1): 10  
Enter second number (Num2): abc  
Error: Invalid input. Please enter valid integers.
```

Explanation:

- Scanner is used to read user input.



Department of Information Technology

- The program attempts to parse the input strings into integers.
- If the second number is zero, an `ArithmeticException` is thrown and handled.
- If the input is not a valid integer, a `NumberFormatException` is caught.
- The array numbers has a fixed size of 2, so accessing any index beyond 1 would throw an `ArrayIndexOutOfBoundsException`.
- The finally block ensures that the scanner is closed, avoiding resource leaks.

Conclusion:

The program demonstrates how to handle exceptions gracefully in Java. It ensures that invalid inputs or runtime errors do not cause the program to crash. By implementing exception handling, the program becomes more robust and user-friendly.

Viva Questions:

1. What is an exception in Java?
2. How is exception handling implemented in Java?
3. What is the difference between checked and unchecked exceptions?
4. What does the finally block do?
5. Can you have multiple catch blocks for a single try block? Explain.
6. What is the purpose of the throws keyword in Java?
7. How does `NumberFormatException` differ from `ArithmeticException`?
8. Why is it important to close resources in the finally block?
9. How would you handle exceptions for reading from a file in Java?
10. Explain the hierarchy of exceptions in Java.



Department of Information Technology

Experiment – 9

Aim:

To develop a Java program that uses collections to count elements with specific properties such as even numbers, odd numbers, prime numbers, and palindromes.

Problem Statement:

Implement a generic program using any collections class to count the number of elements in a collection that have a specific property such as even numbers, odd number, prime number and palindromes.

Objectives:

- To understand the use of Java collections.
- To implement generic methods for counting elements with specific properties.
- To enhance problem-solving skills by applying algorithms to classify elements.
- To reinforce knowledge of fundamental mathematical properties (even, odd, prime, palindrome).

Concepts:

- **Collections in Java:** Java provides various collection frameworks such as ArrayList, HashSet, and LinkedList to manage groups of objects.
- **Generics:** Allows type-safe collection usage.
- **Lambda Expressions:** Used to pass property-checking logic as functional interfaces.
- **Algorithm Design:** Implementing mathematical checks for even, odd, prime, and palindrome properties.

Theory:

Java collections provide dynamic data structures that can store, retrieve, and manipulate data. By applying generic and lambda concepts, we can filter and count elements matching specific conditions. This flexibility and reusability of Java collections make them suitable for processing large datasets efficiently.

Algorithms:

1. Initialize a collection (ArrayList) with a list of integers.
2. Create generic methods to check for properties (even, odd, prime, palindrome).
3. Iterate through the collection, applying property checks using lambda expressions or functional interfaces.
4. Count the elements that satisfy the given condition.
5. Display the results.

Code:

INPUT

```
import java.util.*;  
import java.util.function.Predicate;
```

```
public class CollectionCounter {
```



Department of Information Technology

```
public static void main(String[] args) {
    List<Integer> numbers = Arrays.asList(121, 34, 67, 89, 22, 131, 44, 97, 101);

    System.out.println("Even Numbers: " + countByCondition(numbers,
CollectionCounter::isEven));
    System.out.println("Odd Numbers: " + countByCondition(numbers,
CollectionCounter::isOdd));
    System.out.println("Prime Numbers: " + countByCondition(numbers,
CollectionCounter::isPrime));
    System.out.println("Palindromes: " + countByCondition(numbers,
CollectionCounter::isPalindrome));
}

public static long countByCondition(List<Integer> list, Predicate<Integer> condition) {
    return list.stream().filter(condition).count();
}

public static boolean isEven(int number) {
    return number % 2 == 0;
}

public static boolean isOdd(int number) {
    return number % 2 != 0;
}

public static boolean isPrime(int number) {
    if (number <= 1) return false;
    for (int i = 2; i <= Math.sqrt(number); i++) {
        if (number % i == 0) return false;
    }
    return true;
}

public static boolean isPalindrome(int number) {
    String str = String.valueOf(number);
    return str.equals(new StringBuilder(str).reverse().toString());
}
}
```

OUTPUT

Even Numbers: 3
Odd Numbers: 6
Prime Numbers: 4
Palindromes: 2



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Explanation:

- **List Initialization:** The program initializes a list of integers using `Arrays.asList()`.
- **countByCondition Method:** A generic method using Java Streams and Predicate functional interface to filter and count elements that satisfy a condition.
- **Predicate Implementations:** Methods `isEven`, `isOdd`, `isPrime`, and `isPalindrome` perform the property checks.
- **Lambda Expressions and Method References:** `CollectionCounter::isEven` passes the method reference as a condition.

Conclusion:

This program demonstrates the power of Java collections combined with generics and lambda expressions to process and analyze data efficiently. The approach is scalable and can be extended to include additional properties or complex filtering conditions.

Viva Questions:

1. What is a collection in Java, and why is it used?
2. Explain the concept of generics in Java.
3. How does the Predicate interface work in Java?
4. What is the significance of lambda expressions in this program?
5. How do you determine if a number is prime programmatically?
6. Can this program handle other data types? If yes, how would you modify it?
7. What is the time complexity of the prime number checking method?
8. How does Java Stream API enhance code readability and efficiency?



Department of Information Technology

Experiment – 10

Aim:

To implement a program for maintaining a student records database using file handling in Java. The program will perform the following operations: a) Create Database b) Display Database c) Clear Records d) Modify Records e) Search Records

Problem Statement:

Implement a program for maintaining a student records database using File Handling. Student has Student_id, name, Roll_no, Class, marks and address. Display the data for five students.

- a) Create Databases
- b) Display Database
- c) Clear Records
- d) Modify Records
- e) Search Records

Objectives:

- To understand file handling techniques in Java.
- To implement CRUD (Create, Read, Update, Delete) operations on a text-based student record database.
- To develop skills in data manipulation through file input/output operations.
- To design and implement simple database systems using Java.

Concepts:

- File handling in Java (FileReader, FileWriter, BufferedReader, BufferedWriter, PrintWriter).
- Object-oriented programming (OOP) concepts.
- Exception handling.
- CRUD operations.
- Iteration and decision-making constructs in Java.

Theory:

File handling is an essential part of Java programming that allows users to store data persistently. Using classes from the java.io package, Java facilitates reading from and writing to files. This project demonstrates how to manage student records through file handling. Operations such as creating, displaying, modifying, clearing, and searching records are implemented using basic file I/O streams.

Algorithms:

1. Start the program.
2. Display a menu with options to Create, Display, Clear, Modify, or Search records.
3. Based on the user's choice:
 - Create: Accept student details and store them in a file.
 - Display: Read and display all records from the file.
 - Clear: Delete all contents of the file.



Department of Information Technology

- Modify: Search for a record by Student_id, modify the details, and update the file.
 - Search: Look for a record by Student_id and display the result.
4. Repeat the process until the user exits.
 5. Stop the program.

Code:

INPUT

```
import java.io.*;
import java.util.*;
```

```
public class StudentDatabase {
    static final String FILE_NAME = "students.txt";

    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("\nStudent Database Management System");
            System.out.println("1. Create Database");
            System.out.println("2. Display Database");
            System.out.println("3. Clear Records");
            System.out.println("4. Modify Records");
            System.out.println("5. Search Records");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    createRecord(sc);
                    break;
                case 2:
                    displayRecords();
                    break;
                case 3:
                    clearRecords();
                    break;
                case 4:
                    modifyRecord(sc);
                    break;
                case 5:
                    searchRecord(sc);
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
        break;
    case 6:
        System.out.println("Exiting...");
        return;
    default:
        System.out.println("Invalid choice. Try again.");
    }
}
}

static void createRecord(Scanner sc) throws IOException {
    FileWriter fw = new FileWriter(FILE_NAME, true);
    BufferedWriter bw = new BufferedWriter(fw);
    PrintWriter pw = new PrintWriter(bw);

    System.out.print("Enter Student ID: ");
    String id = sc.nextLine();
    System.out.print("Enter Name: ");
    String name = sc.nextLine();
    System.out.print("Enter Roll No: ");
    String roll = sc.nextLine();
    System.out.print("Enter Class: ");
    String sClass = sc.nextLine();
    System.out.print("Enter Marks: ");
    String marks = sc.nextLine();
    System.out.print("Enter Address: ");
    String address = sc.nextLine();

    pw.println(id + "," + name + "," + roll + "," + sClass + "," + marks + "," + address);
    pw.close();
    System.out.println("Record Added Successfully!");
}

static void displayRecords() throws IOException {
    BufferedReader br = new BufferedReader(new FileReader(FILE_NAME));
    String line;
    while ((line = br.readLine()) != null) {
        System.out.println(line);
    }
    br.close();
}

static void clearRecords() throws IOException {
```



Department of Information Technology

```
PrintWriter pw = new PrintWriter(FILE_NAME);
pw.close();
System.out.println("All Records Cleared!");
}

static void modifyRecord(Scanner sc) throws IOException {
    List<String> records = new ArrayList<>();
    BufferedReader br = new BufferedReader(new FileReader(FILE_NAME));
    String line;
    while ((line = br.readLine()) != null) {
        records.add(line);
    }
    br.close();

    System.out.print("Enter Student ID to Modify: ");
    String id = sc.nextLine();

    boolean found = false;
    for (int i = 0; i < records.size(); i++) {
        String[] data = records.get(i).split(",");
        if (data[0].equals(id)) {
            System.out.print("Enter New Name: ");
            data[1] = sc.nextLine();
            System.out.print("Enter New Marks: ");
            data[4] = sc.nextLine();
            records.set(i, String.join(",", data));
            found = true;
        }
    }

    if (found) {
        PrintWriter pw = new PrintWriter(FILE_NAME);
        for (String record : records) {
            pw.println(record);
        }
        pw.close();
        System.out.println("Record Modified Successfully!");
    } else {
        System.out.println("Record Not Found!");
    }
}

static void searchRecord(Scanner sc) throws IOException {
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
BufferedReader br = new BufferedReader(new FileReader(FILE_NAME));
System.out.print("Enter Student ID to Search: ");
String id = sc.nextLine();
String line;
boolean found = false;
while ((line = br.readLine()) != null) {
    if (line.startsWith(id)) {
        System.out.println("Record Found: " + line);
        found = true;
    }
}
br.close();
if (!found) {
    System.out.println("Record Not Found!");
}
}
```

OUTPUT

Student Database Management System

1. Create Database
2. Display Database
3. Clear Records
4. Modify Records
5. Search Records
6. Exit

Enter your choice: 1

Enter Student ID: 101
Enter Name: Alice
Enter Roll No: 001
Enter Class: 10A
Enter Marks: 85
Enter Address: 123 Maple Street
Record Added Successfully!

Enter your choice: 1

Enter Student ID: 102
Enter Name: Bob
Enter Roll No: 002
Enter Class: 10B
Enter Marks: 90
Enter Address: 456 Oak Avenue

Object Oriented Programming



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Record Added Successfully!

Enter your choice: 2

101,Alice,001,10A,85,123 Maple Street

102,Bob,002,10B,90,456 Oak Avenue

Enter your choice: 5

Enter Student ID to Search: 101

Record Found: 101,Alice,001,10A,85,123 Maple Street

Enter your choice: 4

Enter Student ID to Modify: 102

Enter New Name: Bob Marley

Enter New Marks: 95

Record Modified Successfully!

Enter your choice: 2

101,Alice,001,10A,85,123 Maple Street

102,Bob Marley,002,10B,95,456 Oak Avenue

Enter your choice: 3

All Records Cleared!

Enter your choice: 2

Enter your choice: 6

Exiting...

Explanation:

- The program uses file handling to manage student data by writing and reading from a text file.
- CRUD operations are implemented through menu-driven options.
- FileReader, FileWriter, BufferedReader, and BufferedWriter are used for efficient I/O operations.
- The use of PrintWriter allows overwriting and appending data to the file.

Conclusion:

This project demonstrates how to handle persistent data using file handling techniques in Java. By implementing CRUD operations, the project provides insight into real-world database management through file I/O operations.

Viva Questions:

1. What is file handling in Java?
2. How does BufferedReader differ from FileReader?
3. Why do we use PrintWriter in Java?
4. How would you handle exceptions during file I/O operations?



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

5. Explain the difference between append and overwrite modes in file handling.
6. What are the advantages of using file handling over database management systems for small applications?



Department of Information Technology

Experiment – 11

Aim:

The aim of this project is to implement a simple Object-Oriented Program in Java that simulates the operations of a database system. These operations will include creating a database, adding data, deleting data, updating data, and displaying data.

Problem Statement:

Using all concepts of Object-Oriented programming develop a solution for any application contains following operations such as

- a) Creation of database
- b) Addition of data
- c) Deletion of data
- d) Updation of data
- e) Display of data

Objectives:

1. To understand and implement the concepts of Object-Oriented Programming (OOP).
2. To simulate basic database operations such as creation, addition, deletion, update, and display of data.
3. To demonstrate how to manage collections of data using classes and objects in Java.

Concepts:

1. **Classes and Objects:** Represent real-world entities (e.g., database, records).
2. **Encapsulation:** Grouping related data and operations together in classes.
3. **Inheritance:** Creating a base class with common functionality that can be extended by subclasses.
4. **Polymorphism:** Allowing different classes to define methods with the same name but different implementations.
5. **Abstraction:** Hiding the implementation details and showing only essential functionality to the user.
6. **Arrays/Lists:** For storing data in a structured manner.
7. **Methods:** Functions used to perform operations like adding, deleting, or updating records.

Theory:

- **Database Management:** A database is a collection of data stored in an organized manner. It can perform operations like creating, adding, deleting, updating, and displaying data.
- **OOP Principles:** The database operations are abstracted into classes and objects, allowing for maintainable, scalable, and reusable code.
- **Java Collections:** Lists (ArrayList) will be used to manage the database records.

Algorithms:

1. **Create Database:**
 - Initialize an empty list to store records.
2. **Add Data:**



Department of Information Technology

- Accept data from the user and add it to the list.
- 3. **Delete Data:**
 - Accept an identifier or a record and remove it from the list.
- 4. **Update Data:**
 - Accept an identifier and new data, then update the corresponding record.
- 5. **Display Data:**
 - Print the list of records to show the current data in the database.

Code:

INPUT

```
import java.util.ArrayList;
import java.util.Scanner;

class Record {
    int id;
    String name;
    String description;

    // Constructor
    public Record(int id, String name, String description) {
        this.id = id;
        this.name = name;
        this.description = description;
    }

    // Display method to show record details
    public void display() {
        System.out.println("ID: " + id + ", Name: " + name + ", Description: " + description);
    }
}

class Database {
    private ArrayList<Record> records;

    // Constructor to initialize the database
    public Database() {
        records = new ArrayList<>();
    }

    // Method to add data
    public void addRecord(int id, String name, String description) {
        Record newRecord = new Record(id, name, description);
        records.add(newRecord);
    }
}
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
System.out.println("Record added successfully.");
}

// Method to delete data
public void deleteRecord(int id) {
    for (Record record : records) {
        if (record.id == id) {
            records.remove(record);
            System.out.println("Record deleted successfully.");
            return;
        }
    }
    System.out.println("Record not found.");
}

// Method to update data
public void updateRecord(int id, String newName, String newDescription) {
    for (Record record : records) {
        if (record.id == id) {
            record.name = newName;
            record.description = newDescription;
            System.out.println("Record updated successfully.");
            return;
        }
    }
    System.out.println("Record not found.");
}

// Method to display all records
public void displayRecords() {
    if (records.isEmpty()) {
        System.out.println("No records to display.");
    } else {
        for (Record record : records) {
            record.display();
        }
    }
}

public class DatabaseApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
    }
}
```



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
Database db = new Database();

while (true) {
    System.out.println("\nChoose an operation:");
    System.out.println("1. Create Database");
    System.out.println("2. Add Data");
    System.out.println("3. Delete Data");
    System.out.println("4. Update Data");
    System.out.println("5. Display Data");
    System.out.println("6. Exit");
    int choice = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    switch (choice) {
        case 1:
            db = new Database();
            System.out.println("Database created successfully.");
            break;
        case 2:
            System.out.println("Enter ID:");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.println("Enter Name:");
            String name = scanner.nextLine();
            System.out.println("Enter Description:");
            String description = scanner.nextLine();
            db.addRecord(id, name, description);
            break;
        case 3:
            System.out.println("Enter ID of record to delete:");
            int deleteId = scanner.nextInt();
            db.deleteRecord(deleteId);
            break;
        case 4:
            System.out.println("Enter ID of record to update:");
            int updateId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.println("Enter new Name:");
            String newName = scanner.nextLine();
            System.out.println("Enter new Description:");
            String newDescription = scanner.nextLine();
            db.updateRecord(updateId, newName, newDescription);
            break;
    }
}
```




AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

```
case 5:
    db.displayRecords();
    break;
case 6:
    System.out.println("Exiting...");
    scanner.close();
    return;
default:
    System.out.println("Invalid choice, try again.");
}
}
}
```

OUTPUT

Choose an operation:

1. Create Database
2. Add Data
3. Delete Data
4. Update Data
5. Display Data
6. Exit

1

Database created successfully.

Choose an operation:

1. Create Database
2. Add Data
3. Delete Data
4. Update Data
5. Display Data
6. Exit

2

Enter ID:

101

Enter Name:

Alice

Enter Description:

Software Engineer

Record added successfully.

Choose an operation:

1. Create Database
2. Add Data



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

3. Delete Data
4. Update Data
5. Display Data
6. Exit

2

Enter ID:

102

Enter Name:

Bob

Enter Description:

Data Scientist

Record added successfully.

Choose an operation:

1. Create Database
2. Add Data
3. Delete Data
4. Update Data
5. Display Data
6. Exit

5

ID: 101, Name: Alice, Description: Software Engineer

ID: 102, Name: Bob, Description: Data Scientist

Choose an operation:

1. Create Database
2. Add Data
3. Delete Data
4. Update Data
5. Display Data
6. Exit

4

Enter ID of record to update:

101

Enter new Name:

Alice Johnson

Enter new Description:

Senior Software Engineer

Record updated successfully.

Choose an operation:

1. Create Database
2. Add Data



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

3. Delete Data
4. Update Data
5. Display Data
6. Exit

5

ID: 101, Name: Alice Johnson, Description: Senior Software Engineer

ID: 102, Name: Bob, Description: Data Scientist

Choose an operation:

1. Create Database
2. Add Data
3. Delete Data
4. Update Data
5. Display Data
6. Exit

3

Enter ID of record to delete:

102

Record deleted successfully.

Choose an operation:

1. Create Database
2. Add Data
3. Delete Data
4. Update Data
5. Display Data
6. Exit

5

ID: 101, Name: Alice Johnson, Description: Senior Software Engineer

Choose an operation:

1. Create Database
2. Add Data
3. Delete Data
4. Update Data
5. Display Data
6. Exit

6

Exiting...

Explanation:

- **Record Class:** Represents each record in the database. It contains attributes such as id, name, and description, with a method display() to show the record's details.



AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

- **Database Class:** Manages the database operations. It uses an `ArrayList<Record>` to store records. Methods in this class allow adding, deleting, updating, and displaying records.
- **DatabaseApp Class:** The main class that interacts with the user via the console. It uses a Scanner to take user input and perform operations on the database.

Conclusion:

The program successfully implements a basic database system with operations like creation, addition, deletion, updating, and displaying records. It demonstrates core Object-Oriented Programming concepts, such as encapsulation (using classes and methods), abstraction (hiding implementation details), and data management via collections.

Viva Questions:

1. What are the main Object-Oriented Programming principles used in this project?
2. How does the Database class manage the records?
3. Can you explain how the `addRecord`, `deleteRecord`, and `updateRecord` methods work?
4. Why is `ArrayList` used in this program for storing records?
5. What happens if you try to delete a record that doesn't exist?
6. How would you modify the program to handle multiple types of records, such as employee or student records?
7. How does the program ensure data encapsulation?
8. What improvements can be made to this program to make it more efficient or feature-rich?



Department of Information Technology

Experiment – 12

Aim:

1. The Aim of Experiment is to demonstrate working and creation of Constructor in Java.
2. The aim of Experiment is to demonstrate working and creation of Inheritance in Java.

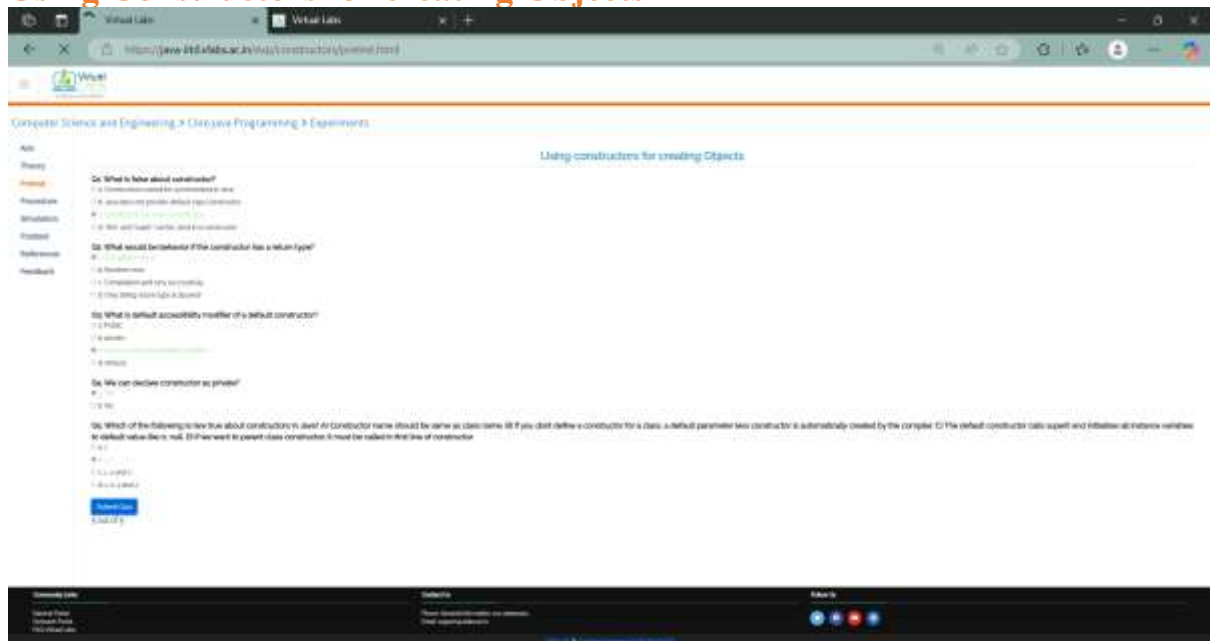
Problem Statement:

Virtual Lab Experiments:

- a) Using constructors for creating Objects: <https://java-iiitd.vlabs.ac.in/exp/constructors/index.html>
- b) Understanding Inheritance in Java: <https://java-iiitd.vlabs.ac.in/exp/inheritance/index.html>

Virtual Lab

Using Constructors for creating Objects





AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology



Preview

Simulator

```
//Main function
public class Student {
    public Student() {
        System.out.println("New object of class student
        created.");
    }

    public static void main(String[] args) {
        Student s1 = new Student();
    }
}
```

Output

Syntax

```
public class ClassName {
    public ConstructorName() {}
    System.out.println("text");
}

public static void main(String args[]){
    ClassName objectName = new ClassName();
}
}
```



Preview

Simulator

execute

Output
New object of class student created

Syntax

```
public class ClassName {
    public ConstructorName() {}
    System.out.println("text");
}

public static void main(String args[]){
    ClassName objectName = new ClassName();
}
}
```




AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

Computer Science and Engineering > Oop java Programming > Experiments

Using constructors for creating Objects

Q1. public class Test { What is the prototype of the default constructor?

A. ...

B. ...

C. ...

D. ...

Q2. In which access should a constructor be defined, so that object of the class can be created in any function?

A. ...

B. ...

C. ...

D. ...

Q3. Default constructor must be default, if parameterized constructor is defined and the object is to be created without arguments.

A. ...

B. ...

C. ...

D. ...

Q4. Which of the modifier can't be used for constructor.....?

A. ...

B. ...

C. ...

D. ...

Q5. The object return type of a constructor is.....

A. ...

B. ...

C. ...

D. ...

Submit Quiz

5 out of 5

Understanding Inheritance in Java

Computer Science and Engineering > Oop java Programming > Experiments

Understanding Inheritance in Java

Q1. Which inheritance to Java programming is not supported?

A. ...

B. ...

C. ...

D. ...

Q2. What is subclass in Java?

A. ...

B. ...

C. ...

D. ...

Q3. If class B is subclassed from class A then which is the correct syntax?

A. ...

B. ...

C. ...

D. ...

Q4. Inheritance relationship in java language is.....

A. ...

B. ...

C. ...

D. ...

Q5. Advantage of inheritance in java programming is/are.....

A. ...

B. ...

C. ...

D. ...

Submit Quiz

5 out of 5



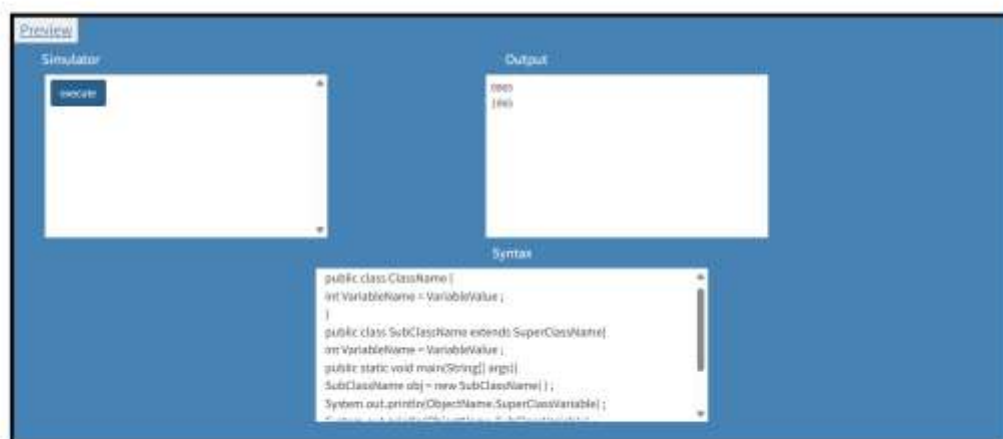
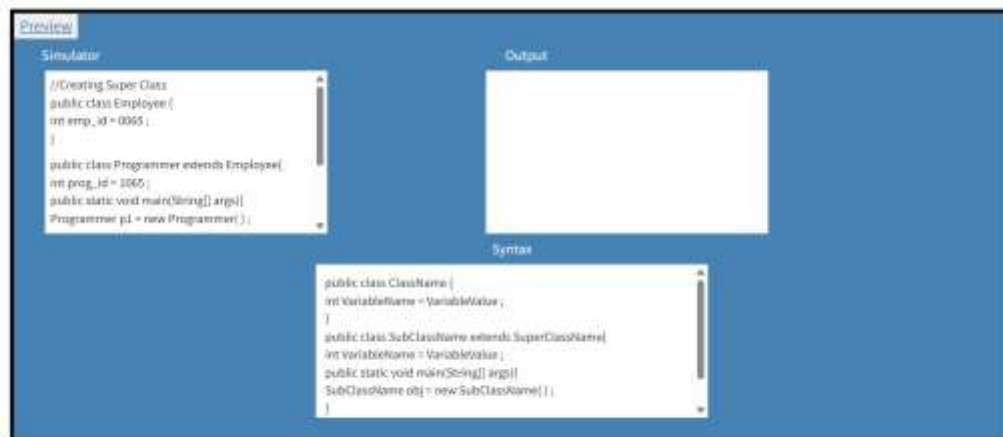
AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology





AISSMS

INSTITUTE OF INFORMATION TECHNOLOGY

ADDING VALUE TO ENGINEERING



Department of Information Technology

The screenshot shows a web browser window with the URL <https://www.aissms.ac.in/virtual-labs/information-technology/>. The page is titled "Computer Science and Engineering > Oop java Programming > Experiments" and the specific experiment is "Understanding Inheritance in Java".

On the left, there is a sidebar with navigation links: Home, Theory, Projects, Practical, Simulations, Experiments (highlighted), and Feedback.

The main content area displays a quiz with the following questions and options:

- Q1. Which of these keywords can be used to prevent inheritance of a class?
A. ☐ public
B. ☐ private
C. ☐ protected
D. ☐ final
- Q2. Which of these is correct way of inheriting class X by class Y?
A. ☐ class Y extends X
B. ☐ class Y inherits X
C. ☐ class Y inherits class X
D. ☐ class Y inherits class X
- Q3. Which of these keywords is used to refer to member of base class from a sub class?
A. ☐ super
B. ☐ this
C. ☐ my
D. ☐ none of the above
- Q4. Multiple inheritance means:
A. ☐ a class has multiple base classes
B. ☐ a class has multiple subclasses
C. ☐ a class has multiple methods
D. ☐ a class has multiple attributes
- Q5. Does a subclass inherit both member variables and methods?
A. ☐ Yes
B. ☐ No
C. ☐ Yes, only member variables are inherited
D. ☐ No, only methods are inherited

At the bottom of the quiz, there is a "Submit Quiz" button and a score of "0 out of 5".