Laboratory Manual for

Object Oriented Programming -I (3140705)

B.E. Semester 4th (Computer Engineering)

Vishwakarma Government Engineering College, Chandkheda
Gujarat

Directorate of Technical Education,
Gandhinagar, Gujarat

Vishwakarma Government Engineering College, Chandkheda

Certificate

This is to certify that Mr./Ms. Mehta Yashesh Jigarkumar Enrollment No. 23017010755 of B.E. Semester 4, Computer Engineering of this Institute (GTU Code: 07) has satisfactorily completed the Practical work for the subject Object Oriented Programming 1-(3140705) for the academic year 2024-25.

Place: __________ Date: __________

Name and Sign of Faculty member **Head of the Department**

Preface

Main motto of any laboratory/practical/field work is for enhancing required skills as well as creating ability amongst students to solve real time problem by developing relevant competencies in psychomotor domain. By keeping in view, GTU has designed competency focused outcome-based curriculum for engineering degree programs where sufficient weightage is given to practical work. It shows importance of enhancement of skills amongst the students and it pays attention to utilize every second of time allotted for practical amongst students, instructors and faculty members to achieve relevant outcomes by performing the experiments rather than having merely study type experiments. It is must for effective implementation of competency focused outcome-based curriculum that every practical is keenly designed to serve as a tool to develop and enhance relevant competency required by the various industry among every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual is designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual students can go through the relevant theory and procedure in advance before the actual performance which creates an interest and students can have basic idea prior to performance. This in turn enhances pre-determined outcomes amongst students. Each experiment in this manual begins with competency, industry relevant skills, course outcomes as well as practical outcomes (objectives). The students will also achieve safety and necessary precautions to be taken while performing practical.

This manual also provides guidelines to faculty members to facilitate student centric lab activities through each experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve the outcomes. It also gives an idea that how students will be assessed by providing rubrics.

Java is a multi-platform, object-oriented, and network-centric language that can be used as a platform. It is a fast, secure, reliable programming language for coding everything from mobile apps and enterprise software to big data applications and server-side technologies. Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.

Utmost care has been taken while preparing this lab manual however always there is chances of improvement. Therefore, we welcome constructive suggestions for improvement and removal of errors if any.

DTE's Vision

- To provide globally competitive technical education
- Remove geographical imbalances and inconsistencies
- Develop student friendly resources with a special focus on girls' education and support to weaker sections
- Develop programs relevant to industry and create a vibrant pool of technical professionals

Institute's Vision

- To create an ecosystem for proliferation of socially responsible and technically sound engineers, innovators and entrepreneurs.

Institute's Mission

- To develop state-of-the-art laboratories and well-equipped academic infrastructure.
- To motivate faculty and staff for qualification up-gradation, and enhancement of subject knowledge.
- To promote research, innovation and real-life problem-solving skills.
- To strengthen linkages with industries, academic and research organizations.
- To reinforce concern for sustainability, natural resource conservation and social responsibility.

Department's Vision

- To create an environment for providing value-based education in Computer Engineering through innovation, team work and ethical practices.

Department's Mission

- To produce computer engineering graduates according to the needs of industry, government, society and scientific community.
- To develop state of the art computing facilities and academic infrastructure.
- To develop partnership with industries, government agencies and R & D organizations for knowledge sharing and overall development of faculties and students.
- To solve industrial, governance and societal issues by applying computing techniques.
- To create environment for research and entrepreneurship.

Program Outcomes (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science,engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyzecomplex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineeringproblems 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.
4. **Conduct investigations of complex problems:** Use research-based knowledgeand research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques,resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextualknowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professionalengineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics andresponsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as amember or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activitieswith 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.
11. **Project management and finance:** Demonstrate knowledge and understandingof 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.
12. **Life-long learning:** Recognize the need for, and have the preparation and abilityto engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

- Sound knowledge of fundamentals of computer science and engineering including software and hardware.
- Develop the software using sound software engineering principles having web based/mobile based interface.
- Use various tools and technology supporting modern software frameworks for solving problems having large volume of data in the domain of data science and machine learning.

Program Educational Objectives (PEOs)

- Possess technical competence in solving real life problems related to Computing.
- Acquire good analysis, design, development, implementation and testing skills to formulate simple computing solutions to the business and societal needs.
- Provide requisite skills to pursue entrepreneurship, higher studies, research, and development and imbibe high degree of professionalism in the fields of computing.
- Embrace life-long learning and remain continuously employable.
- Work and excel in a highly competence supportive, multicultural and professional environment which abiding to the legal and ethical responsibilities.

Practical – Course Outcome matrix

Course Outcomes (COs):

1. Use various Java constructs, features and libraries for simple problems.
2. Demonstrate how to define and use classes, interfaces, create objects and methods, how to override and overload methods, compile and execute programs.
3. Write a program using exception handling, multithreading with synchronization.
4. Write a program using Files, binary I/O, collection Frameworks for a given problem.
5. Design and develop GUI based applications in a group using modern tools and frameworks.

Sr. No.	Objective(s) of Experiment	CO1	CO2	CO3	CO4	CO5
1.	To learn basic java programming constructs.	√				
2.	To learn Arrays and Strings in Java.	√				
3.	To implement basic object-oriented concepts.		√			
4.	To implement inheritance and object-oriented concepts.		√			
5.	To demonstrate the use of abstract classes and interfaces.		√			
6.	To implement packages and exception handling in JAVA application.			√		
7.	To demonstrate I/O from files.				√	
8.	To learn JAVA FX UI Controls.					√
9.	To implement event handling and animation.					√
10.	To learn recursion and generics.				√	
11.	To demonstrate the use of Collection framework.				√	
12.	To demonstrate the use of multithreading.			√		

Industry Relevant Skills

The following industry relevant competency is expected to be developed in the student by undertaking the practical work of this laboratory.

1. Object oriented application development
2. Networking application development
3. GUI based application development

Guidelines for Faculty members

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain basic concepts/theory related to the experiment to the students before starting of each practical
3. Involve all the students in performance of each experiment.
4. Teacher is expected to share the skills and competencies to be developed in the students and ensure that the respective skills and competencies are developed in the students after the completion of the experimentation.
5. Teachers should give opportunity to students for hands-on experience after the demonstration.
6. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected from the students by concerned industry.
7. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions or not.
8. Teacher is expected to refer complete curriculum of the course and follow the guidelines for implementation.

Instructions for Students

1. Students are expected to carefully listen to all the theory classes delivered by the faculty members and understand the COs, content of the course, teaching and examination scheme, skill set to be developed etc.
2. Students shall organize the work in the group and make record of all observations.
3. Students shall develop maintenance skill as expected by industries.
4. Student shall attempt to develop related hand-on skills and build confidence.
5. Students shall make a small project/application in Java.
6. Student shall develop the habits of evolving more ideas, innovations, skills etc. apart from those included in scope of manual.
7. Student shall refer technical magazines and books.
8. Student should develop a habit of submitting the experimentation work as per the schedule and s/he should be well prepared for the same.

Common Safety Instructions

Students are expected to

1. Switch on the PC carefully (not to use wet hands)
2. Shutdown the PC properly at the end of your Lab
3. Carefully Handle the peripherals (Mouse, Keyboard, Network cable etc)
4. Use Laptop in lab after getting permission from Teacher

Index
(Progressive Assessment Sheet)

Sr. No.	Objective(s) of Experiment	Page No.	Date of performance	Date of submission	Assessment Marks	Sign. of Teacher with date	Remarks
1.	To learn basic java programming constructs.						
2.	To learn Arrays and Strings in Java.						
3.	To implement basic object-oriented concepts.						
4.	To implement inheritance and object-oriented concepts.						
5.	To demonstrate the use of abstract classes and interfaces.						
6.	To implement exception handling in JAVA application and Multithreading.						
7.	To demonstrate I/O from files.						
8.	To learn JAVA FX UI Controls and To implement event handling and animation.						
9.	To learn recursion and generics.						
10.	To demonstrate the use of Collection framework.						
Total							

1. COURSE OUTCOMES

After learning the course, the students should be able to:

1. Use various Java constructs, features and libraries for simple problems.
2. Demonstrate how to define and use classes, interfaces, create objects and methods, how to override and overload methods, compile and execute programs.
3. Write a program using exception handling, multithreading with synchronization.
4. Write a program using Files, binary I/O, collection Frameworks for a given problem.
5. Design and develop GUI based applications in a group using modern tools and frameworks.

2. TEACHING AND EXAMINATION SCHEME

Teaching Scheme			**Credits**	**Examination Marks**				**Total Marks**
L	T	P	C	**Theory Marks**		**Practical Marks**		
				ESE (E)	PA (M)	ESE (V)	PA (I)	
4	0	2	5	70	30	30	20	150

3. SUGGESTED LEARNING RESOURCES

Reference Books:

1. Intro to Java Programming, 10th edition, Y.Daniel Liang, Pearson
2. Object oriented programming with Java , RajkumarBuyya,SThamaraiSelvi, Xingchen Chu, McGrawHill
3. Programming in Java, SachinMalhotra, SaurabhChoudhary, Oxford
4. Programming with JAVA , E Balagurusamy, McGrawHill
5. CORE JAVA volume -I Cay Horstmann, Pearson

Major Equipment: Computer, Laptop

List of Open Source Software/learning website:

https://docs.oracle.com/javase/tutorial/java/index.html
https://www.tutorialspoint.com/JAVA/
https://dev.java/learn/
https://www.codecademy.com/learn/learn-java
https://www.w3schools.com/java/

Java:

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture.

OpenJDK:

OpenJDK (Open Java Development Kit) is a free and open-source implementation of the Java Platform, Standard Edition (Java SE). It is the result of an effort Sun Microsystems began in 2006. The implementation is licensed under the GPL-2.0-only with a linking exception. Were it not for the GPL linking exception, components that linked to the Java class library would be subject to the terms of the GPL license. OpenJDK is the official reference implementation of Java SE since version 7.

JVM:

The Java Virtual Machine, or JVM, executes live Java applications. Every JRE includes a default JRE, but developers are free to choose another that meets the specific resource needs of their applications.

JRE:

The Java Virtual Machine, or JVM, executes live Java applications. Every JRE includes a default JRE, but developers are free to choose another that meets the specific resource needs of their applications. The Java Virtual Machine, or JVM, executes live Java applications. Every JRE includes a default JRE, but developers are free to choose another that meets the specific resource needs of their applications.

JAVA IDEs:

IDEs typically provide a code editor, a compiler or interpreter and a debugger that the developer accesses through a unified graphical user interface (GUI). Here are a few popular Java IDEs:

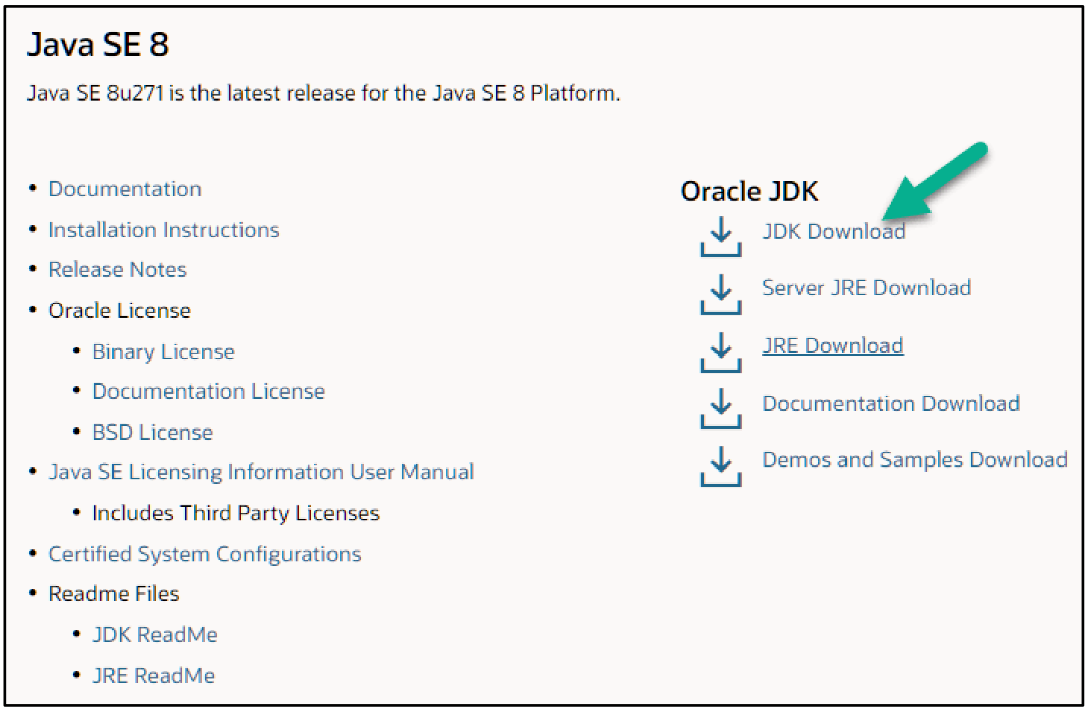
Eclipse: a Java-based open source platform that enables the creation of highly customized IDEs from plug-in components built by Eclipse members. The platform is user-friendly for beginners and also suitable for the creation of more sophisticated applications. Eclipse includes a lot of plug-ins that allow developers to develop and test code written in other languages.

NetBeans: a Java-based IDE and underlying application platform framework. In addition to Java, JavaScript and JavaFX, NetBeans supports C/C++, PHP, Groovy, and HTML5.

How to install Java for Windows:

Following are the steps on how to install Java in Windows 10 for JDK 8 free download for 32 bit or JDK8 download for Windows 64 bit and installation

Step 1) Go to https://www.oracle.com/java/technologies/downloads/. Click on JDK Download for Java download JDK 8.



Java SE 8

Java SE 8u271 is the latest release for the Java SE 8 Platform.

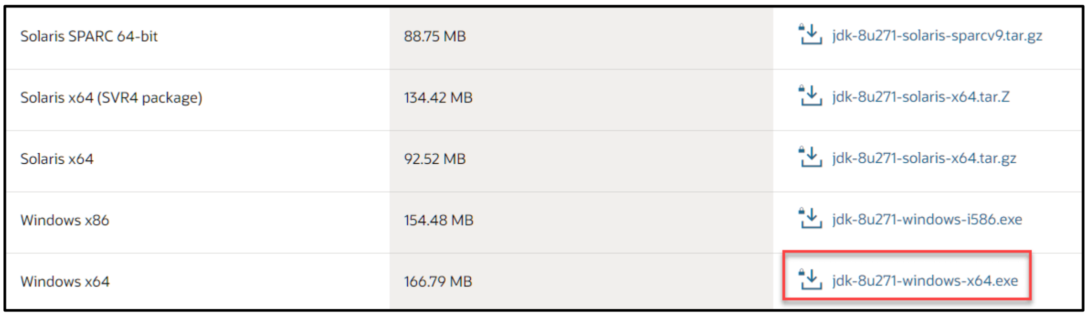
- Documentation
- Installation Instructions
- Release Notes
- Oracle License
  - Binary License
  - Documentation License
  - BSD License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme Files
  - JDK ReadMe
  - JRE ReadMe

Oracle JDK

- JDK Download
- Server JRE Download
- JRE Download
- Documentation Download
- Demos and Samples Download

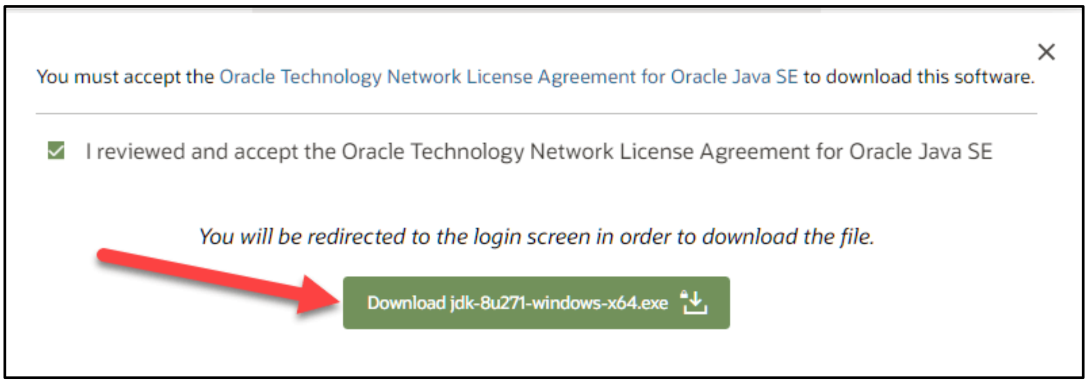
Step 2) Next,

1. Accept License Agreement
2. Download Java 8 JDK for your version 32 bit or JDK download 64 bit.



Solaris SPARC 64-bit	88.75 MB	jdk-8u271-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.42 MB	jdk-8u271-solaris-x64.tar.Z
Solaris x64	92.52 MB	jdk-8u271-solaris-x64.tar.gz
Windows x86	154.48 MB	jdk-8u271-windows-i586.exe
Windows x64	166.79 MB	jdk-8u271-windows-x64.exe

Step 3) When you click on the Installation link the popup will be open. Click on I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE development kit and you will be redirected to the login page. If you don’t have an oracle account you can easily sign up by adding basics details of yours.

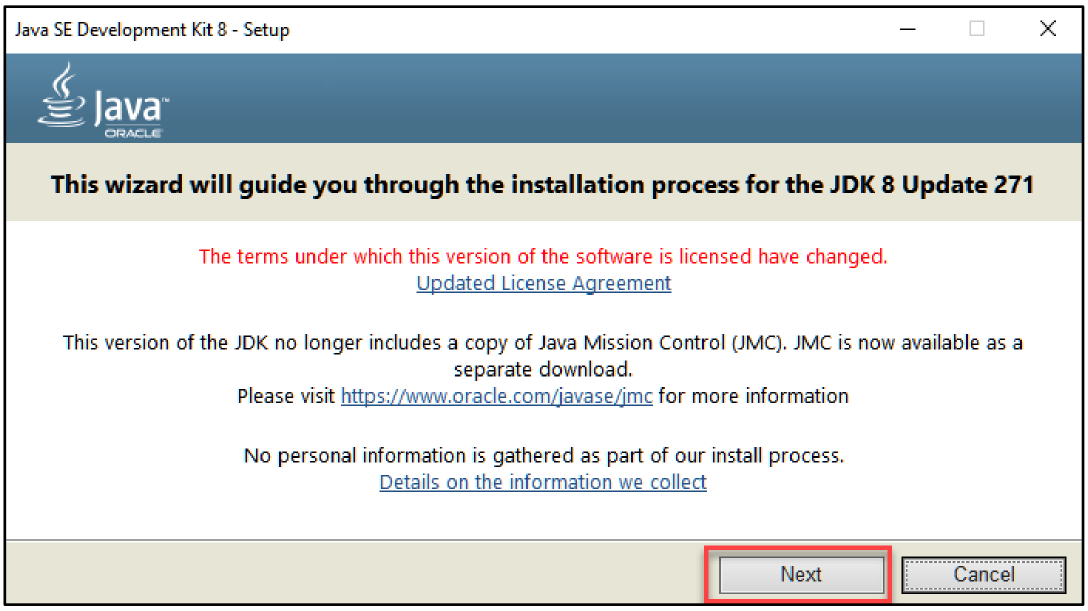


You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.

☑ I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE

You will be redirected to the login screen in order to download the file.

Step 4) Once the Java JDK 8 download is complete, run the exe for install JDK. Click Next



Java SE Development Kit 8 - Setup

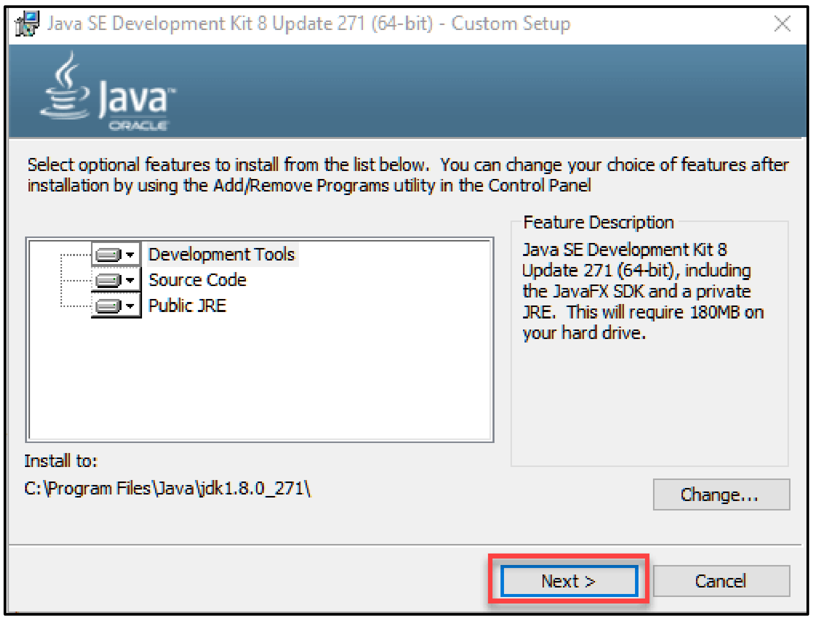
This wizard will guide you through the installation process for the JDK 8 Update 271

The terms under which this version of the software is licensed have changed.
Updated License Agreement

This version of the JDK no longer includes a copy of Java Mission Control (JMC). JMC is now available as a separate download.
Please visit https://www.oracle.com/javase/jmc for more information

No personal information is gathered as part of our install process.
Details on the information we collect

Step 5) Select the PATH to install Java in Windows… You can leave it Default. Click next.



Java SE Development Kit 8 Update 271 (64-bit) - Custom Setup

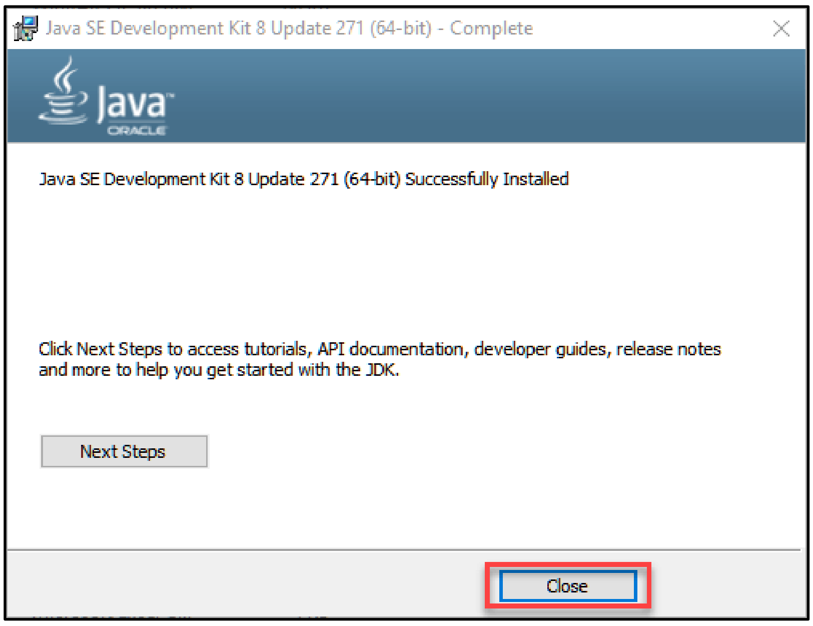
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel

Development Tools
Source Code
Public JRE

Feature Description
Java SE Development Kit 8 Update 271 (64-bit), including the JavaFX SDK and a private JRE. This will require 180MB on your hard drive.

Install to:
C:\Program Files\Java\jdk1.8.0_271\

Step 6) Once you install Java in windows, click Close


Java SE Development Kit 8 Update 271 (64-bit) - Complete

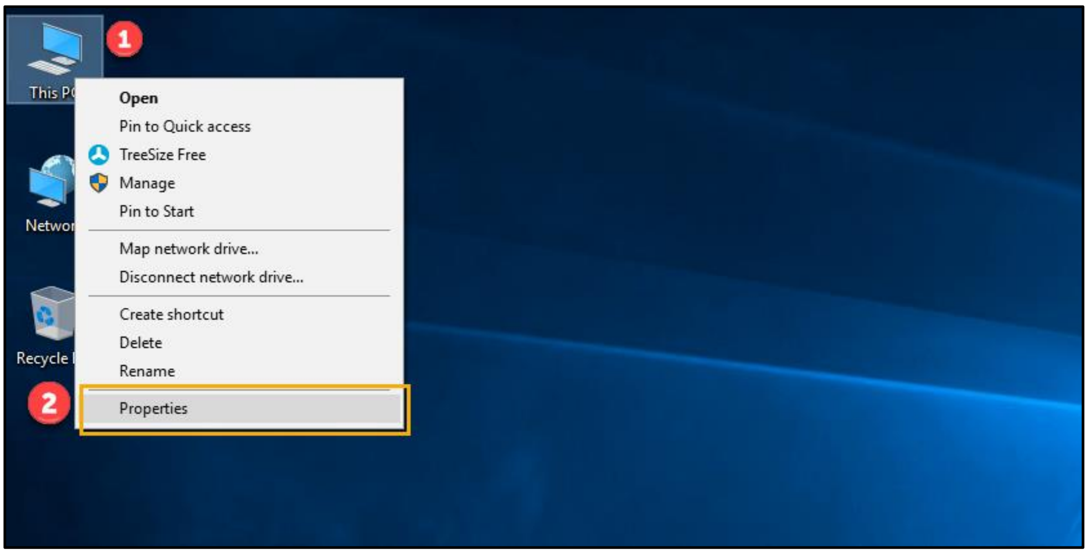
Java SE Development Kit 8 Update 271 (64-bit) Successfully Installed

Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.

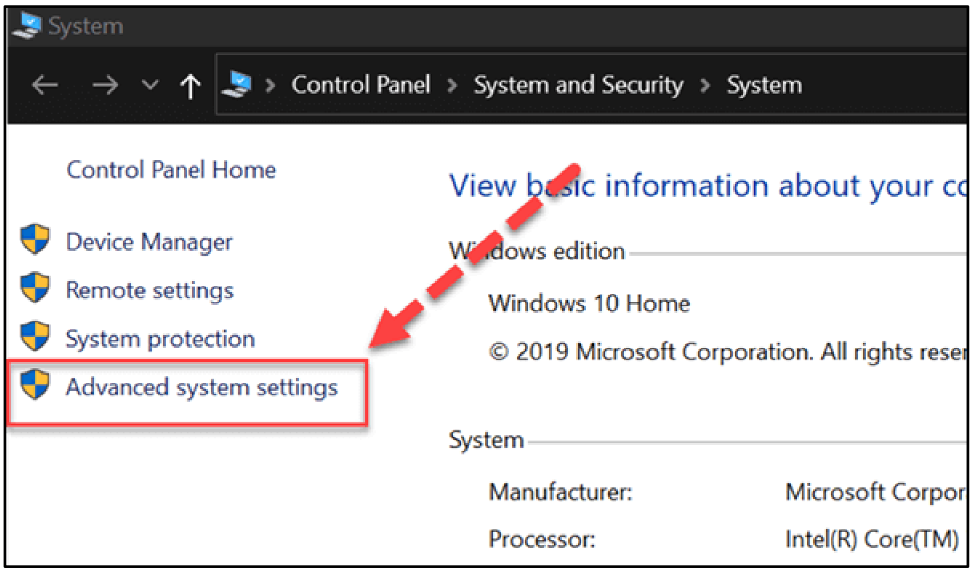
How to set Environment Variables in Java: Path and Classpath:

The PATH variable gives the location of executables like javac, java etc. It is possible to run a program without specifying the PATH but you will need to give full path of executable like **C:\Program Files\Java\jdk1.8.0_271\bin\javac A.java** instead of simple **javac A.java**
The CLASSPATH variable gives location of the Library Files.
Let's look into the steps to set the PATH and CLASSPATH

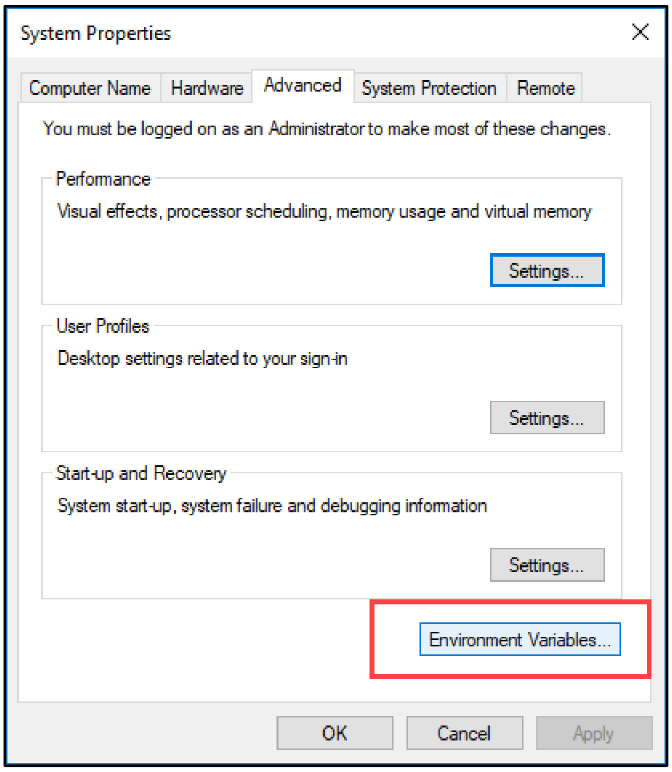
Step 1) Right Click on the My Computer and Select the properties



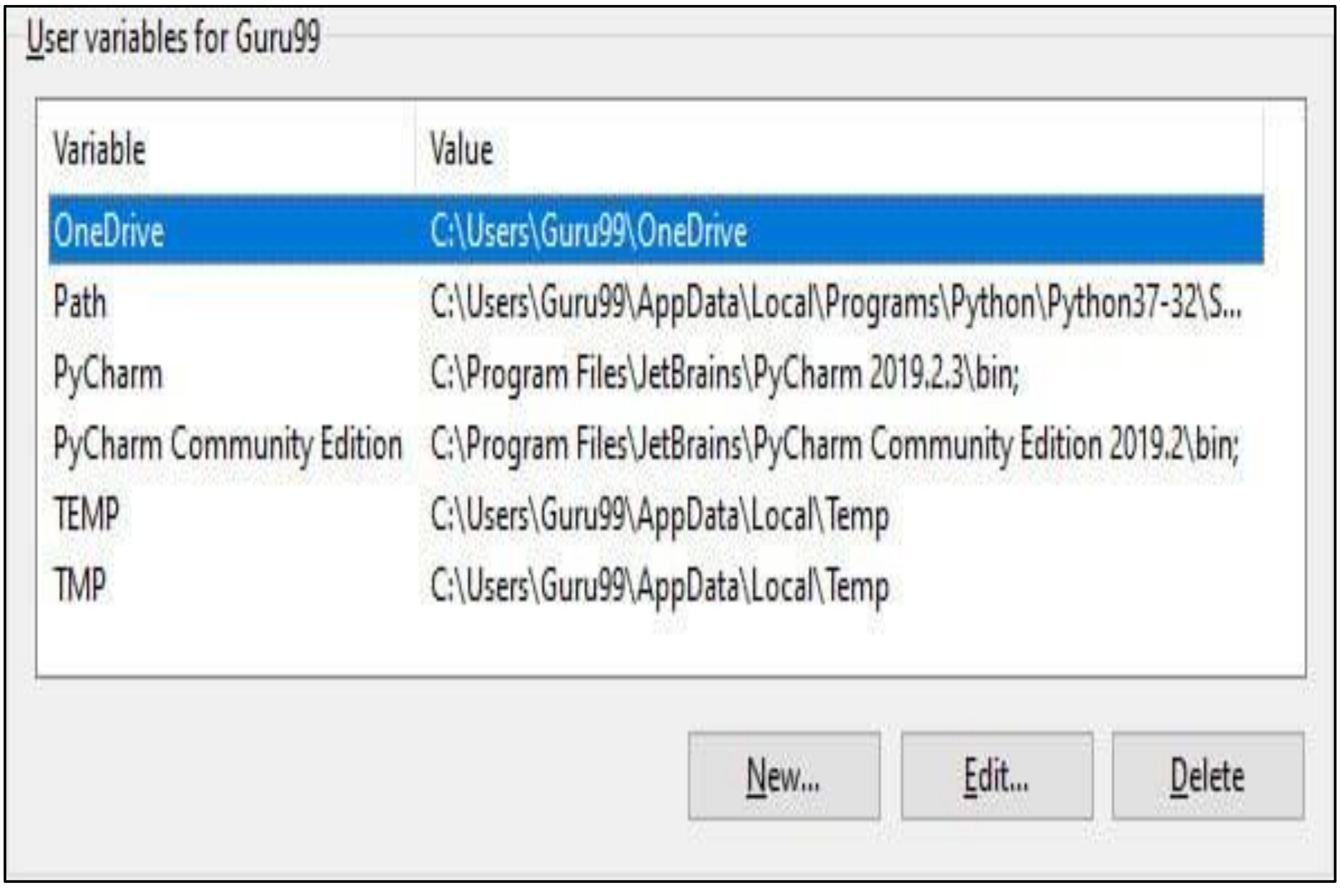
Step 2) Click on advanced system settings


System
Control Panel > System and Security > System
Control Panel Home
Device Manager
Remote settings
System protection
Advanced system settings
View basic information about your c
Windows edition
Windows 10 Home
© 2019 Microsoft Corporation. All rights rese
System
Manufacturer: Microsoft Corpor
Processor: Intel(R) Core(TM)

Step 3) Click on Environment Variables to set Java runtime environment

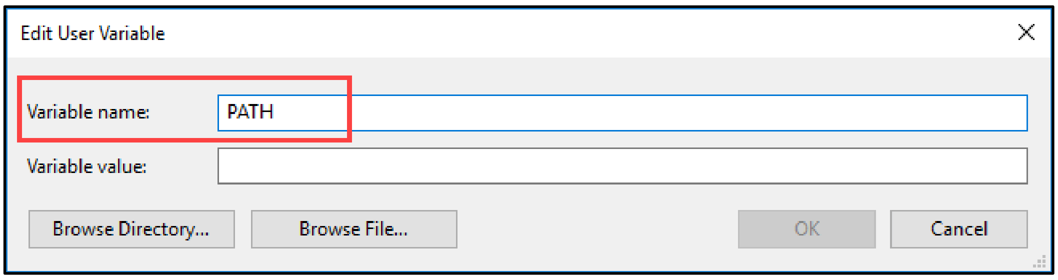

System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Start-up and Recovery
System start-up, system failure and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

Step 4) Click on new Button of User variables


User variables for Guru99

Variable	Value
OneDrive	C:\Users\Guru99\OneDrive
Path	C:\Users\Guru99\AppData\Local\Programs\Python\Python37-32\S...
PyCharm	C:\Program Files\JetBrains\PyCharm 2019.2.3\bin;
PyCharm Community Edition	C:\Program Files\JetBrains\PyCharm Community Edition 2019.2\bin;
TEMP	C:\Users\Guru99\AppData\Local\Temp
TMP	C:\Users\Guru99\AppData\Local\Temp

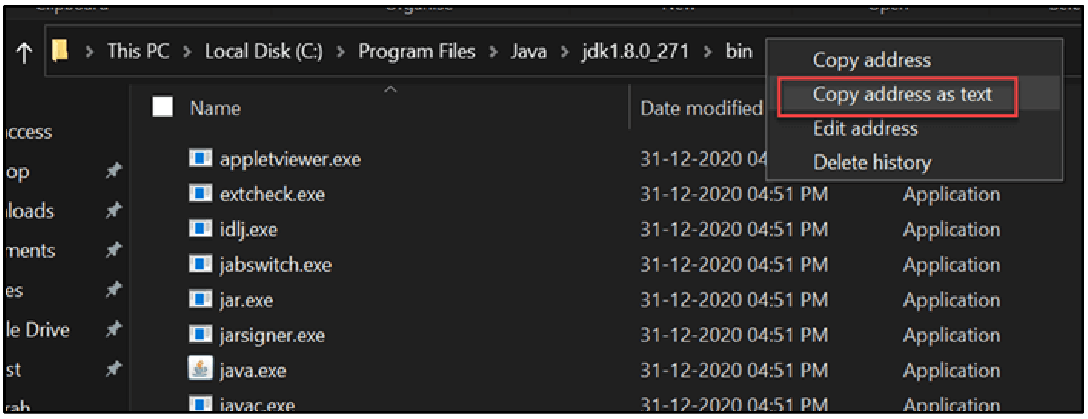
Step 5) Type PATH in the Variable name.


Edit User Variable

Variable name: PATH

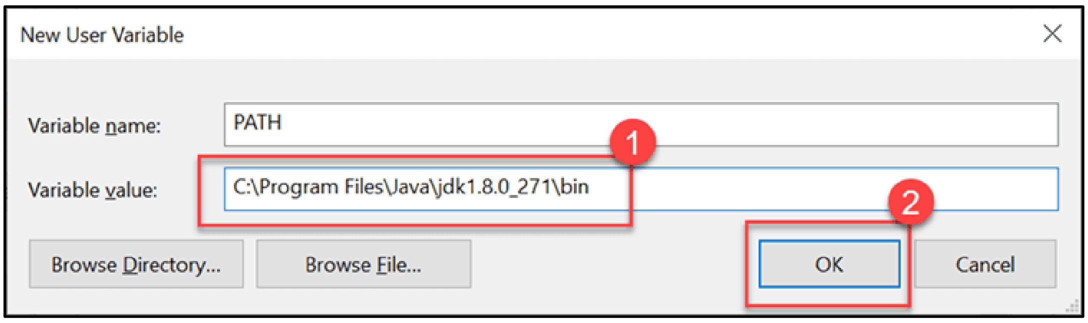
Variable value:

Step 6) Copy the path of bin folder which is installed in JDK folder.



Name	Date modified	
appletviewer.exe	31-12-2020 04	
extcheck.exe	31-12-2020 04:51 PM	Application
idlj.exe	31-12-2020 04:51 PM	Application
jabswitch.exe	31-12-2020 04:51 PM	Application
jar.exe	31-12-2020 04:51 PM	Application
jarsigner.exe	31-12-2020 04:51 PM	Application
java.exe	31-12-2020 04:51 PM	Application
javac.exe	31-12-2020 04:51 PM	Application

Step 7) Paste Path of bin folder in Variable value. Click on OK Button.

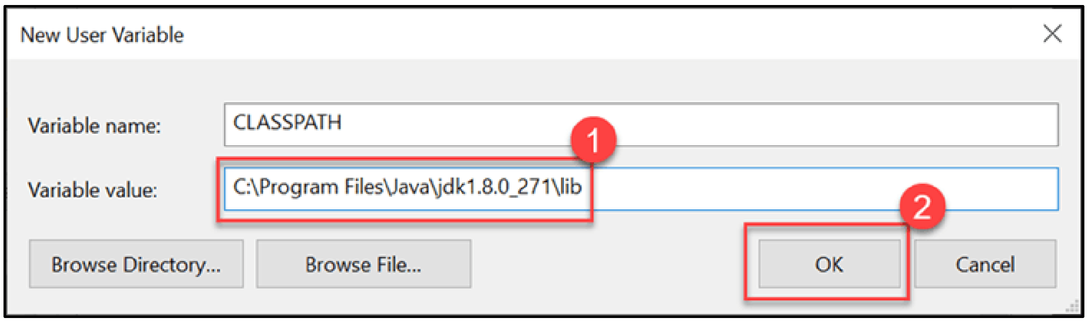


Variable name: PATH
Variable value: C:\Program Files\Java\jdk1.8.0_271\bin

Note: In case you already have a PATH variable created in your PC, edit the PATH variable to PATH = <JDK installation directory>\bin;%PATH%;

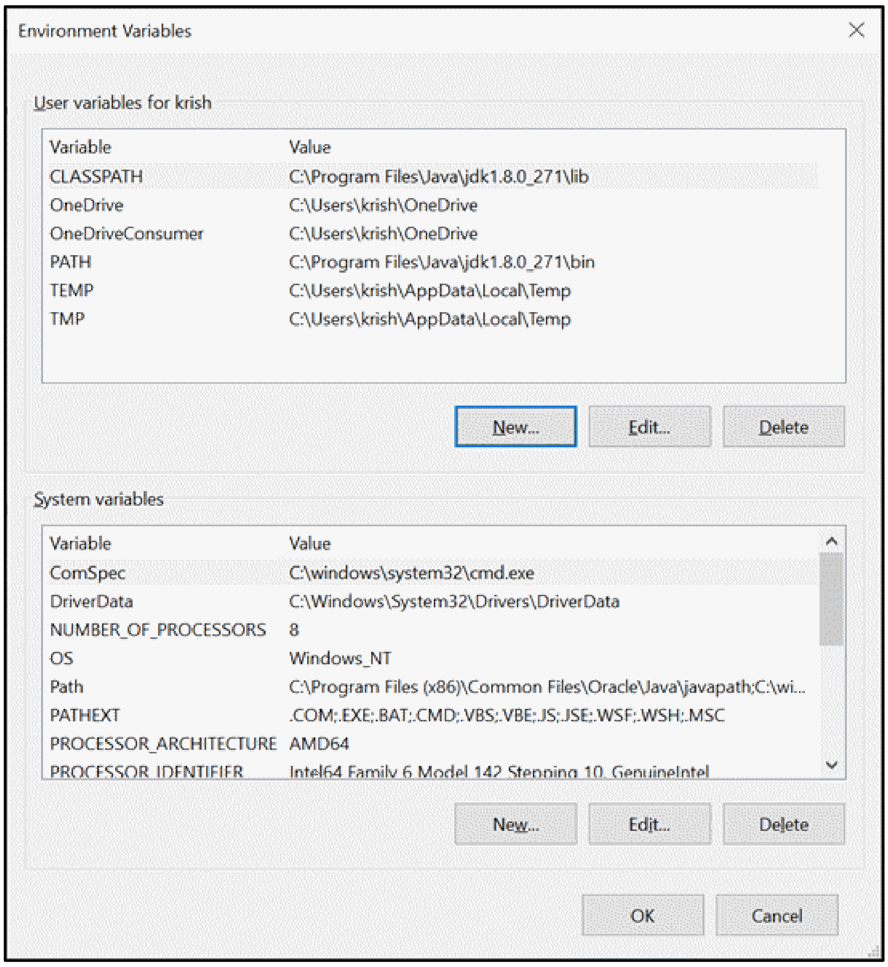
Here, %PATH% appends the existing path variable to our new value

Step 8) You can follow a similar process to set CLASSPATH.



Variable name: CLASSPATH
Variable value: C:\Program Files\Java\jdk1.8.0_271\lib

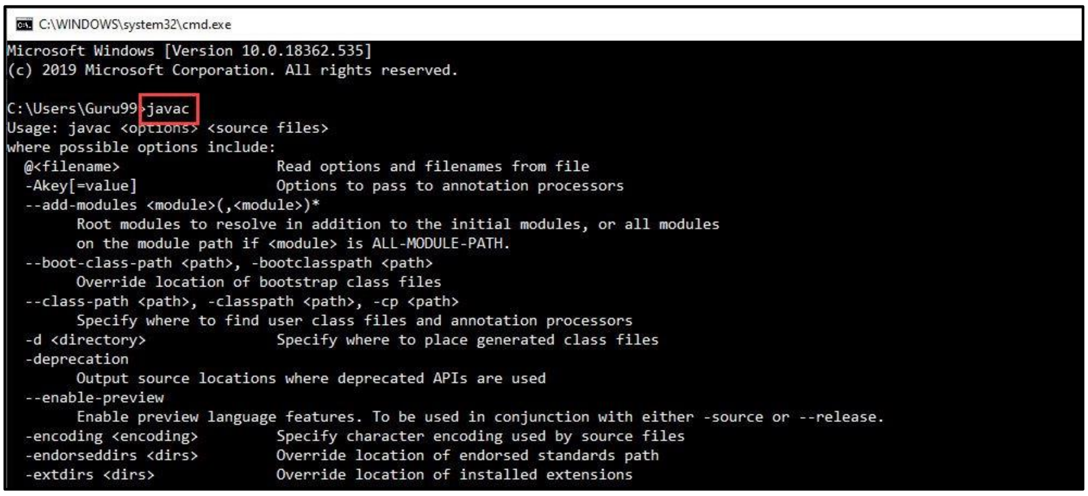
Step 9) Click on OK button



Variable	Value
CLASSPATH	C:\Program Files\Java\jdk1.8.0_271\lib
OneDrive	C:\Users\krish\OneDrive
OneDriveConsumer	C:\Users\krish\OneDrive
PATH	C:\Program Files\Java\jdk1.8.0_271\bin
TEMP	C:\Users\krish\AppData\Local\Temp
TMP	C:\Users\krish\AppData\Local\Temp

Variable	Value
ComSpec	C:\windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\wi...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, GenuineIntel

Step 10) Go to command prompt and type javac commands.
If you see a screen like below, Java is installed.



```
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Guru99>javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>                  Read options and filenames from file
  -Akey[=value]                Options to pass to annotation processors
  --add-modules <module>(,<module>)*
        Root modules to resolve in addition to the initial modules, or all modules
        on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, -bootclasspath <path>
        Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
        Specify where to find user class files and annotation processors
  -d <directory>               Specify where to place generated class files
  -deprecation
        Output source locations where deprecated APIs are used
  --enable-preview
        Enable preview language features. To be used in conjunction with either -source or --release.
  -encoding <encoding>         Specify character encoding used by source files
  -endorseddirs <dirs>         Override location of endorsed standards path
  -extdirs <dirs>              Override location of installed extensions
```

Experiment No: 1

AIM: To learn basic JAVA programming constructs.

Date:

CO mapped: CO-1

Objectives: (a) To learn and understand the different basic structures in java, such as syntax, logics, libraries and proper indentation.

Background:

Java Variables

A variable is a container that holds the value while the Java program is executed. A variable is assigned with a data type. Variable is a name of a memory location. There are three types of variables in java: local, instance, and static.

Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:
Primitive data types: The primitive data types include boolean, char, byte, short, int, long, float and double.

There are 8 types of primitive data types:

- boolean data type
- byte data type
- char data type
- short data type
- int data type
- long data type
- float data type
- double data type

Non-primitive data types: The non-primitive data types include Classes, Interfaces, and Arrays.

Operators in Java

Operator in Java is a symbol that is used to perform operations. For example: +, -, *, / etc.
There are many types of operators in Java which are given below:

- Unary Operator,
- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

Java Control Statements

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements. It is one of the fundamental features of Java, which provides a smooth flow of program.
Java provides three types of control flow statements.

- Decision Making statements
  - if statements
  - switch statement
- Loop statements
  - do while loop
  - while loop
  - for loop
  - for-each loop
- Jump statements
  - break statement
  - continue statement

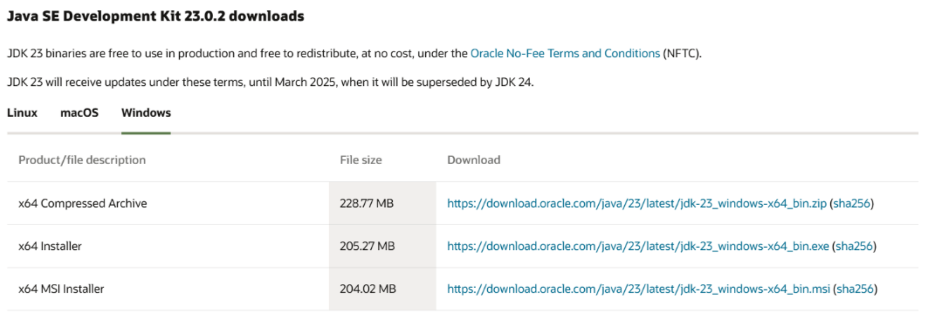
Practical questions:

1. **Install JDK and IDE in your system. Write down the steps of installation with screenshots.**

Answer:-

- **Steps for installing JDK:**

Step-1 Download **JDK**(Java Development Kit) according to the system configurations from Oracle.



Java SE Development Kit 23.0.2 downloads

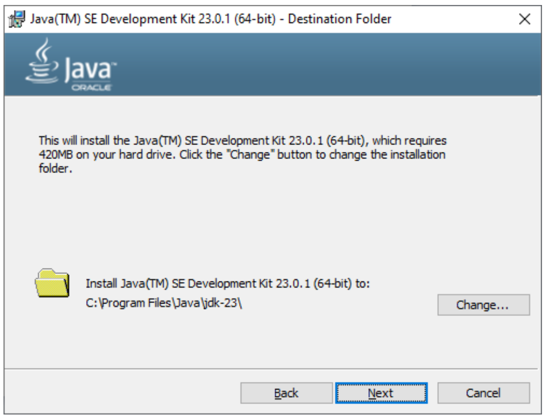
JDK 23 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 23 will receive updates under these terms, until March 2025, when it will be superseded by JDK 24.

Linux macOS Windows

Product/file description	File size	Download
x64 Compressed Archive	228.77 MB	https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.zip (sha256)
x64 Installer	205.27 MB	https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.exe (sha256)
x64 MSI Installer	204.02 MB	https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.msi (sha256)

Step-2 Go through the installation process thoroughly.

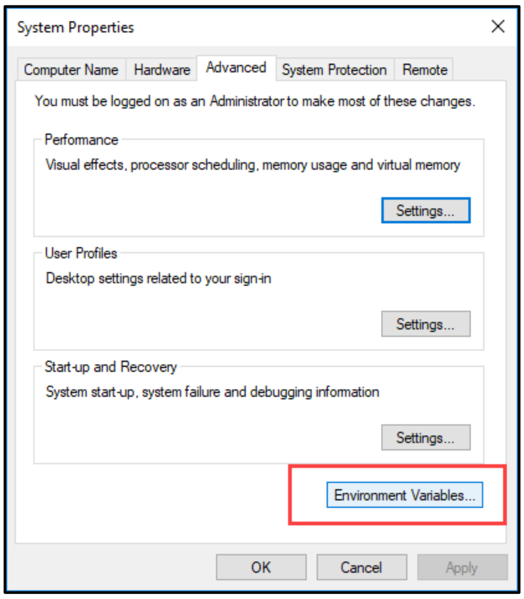


Java(TM) SE Development Kit 23.0.1 (64-bit) - Destination Folder

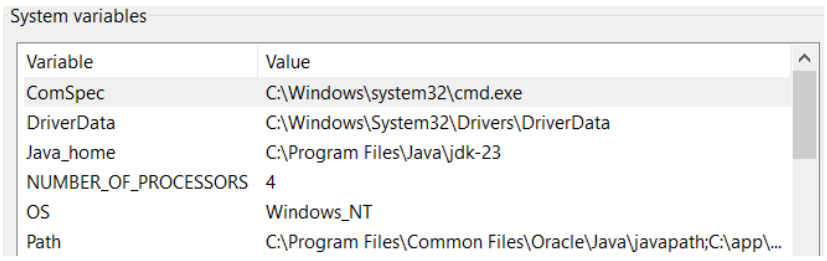
This will install the Java(TM) SE Development Kit 23.0.1 (64-bit), which requires 420MB on your hard drive. Click the "Change" button to change the installation folder.

Install Java(TM) SE Development Kit 23.0.1 (64-bit) to:
C:\Program Files\Java\jdk-23\

Step-3 From Control Panel, open advance system settings.


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Start-up and Recovery
System start-up, system failure and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

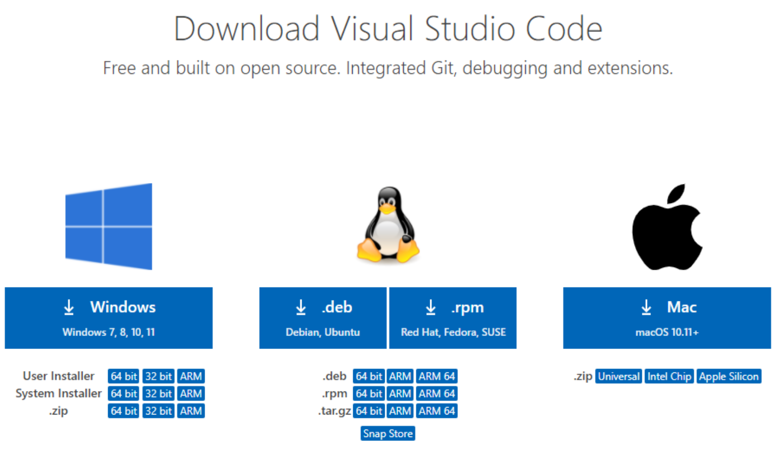
Step-4 Update path in environment variables and add new path for the jdk bin folder.


System variables

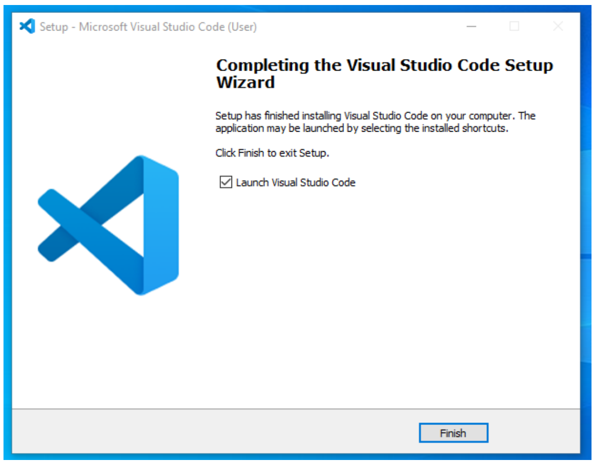
Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
Java_home	C:\Program Files\Java\jdk-23
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\app\...

- <u>**Steps for installing IDE:**</u>

Step-1 Download the VSCode installer from its official website.


Download Visual Studio Code
Free and built on open source. Integrated Git, debugging and extensions.
Windows
Windows 7, 8, 10, 11
.deb
Debian, Ubuntu
.rpm
Red Hat, Fedora, SUSE
Mac
macOS 10.11+
User Installer 64 bit 32 bit ARM
System Installer 64 bit 32 bit ARM
.zip 64 bit 32 bit ARM
.deb 64 bit ARM ARM 64
.rpm 64 bit ARM ARM 64
.tar.gz 64 bit ARM ARM 64
Snap Store
.zip Universal Intel Chip Apple Silicon

Step-2 Go through the steps of installation and finish the installation of the IDE.


Setup - Microsoft Visual Studio Code (User)

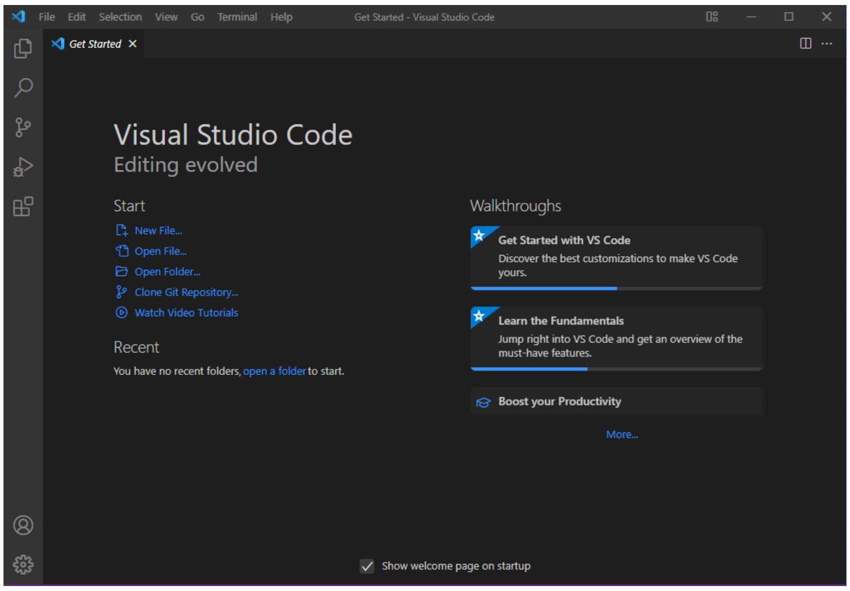
Completing the Visual Studio Code Setup Wizard

Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☑ Launch Visual Studio Code

Step-3 Now the IDE is ready to use.



You have no recent folders, open a folder to start.

Discover the best customizations to make VS Code yours.

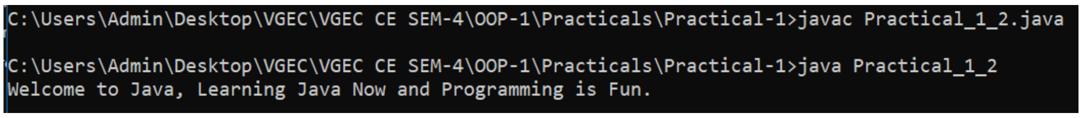
Jump right into VS Code and get an overview of the must-have features.

2. **Write a Program that displays Welcome to Java, Learning Java Now and Programming is fun.**

Procedure:-

```
class Practical_1_2{
   public static void main(String args[]) {
      System.out.println("Welcome to Java, Learning Java Now and Programming is Fun.");
   }
}
```

Observation:-

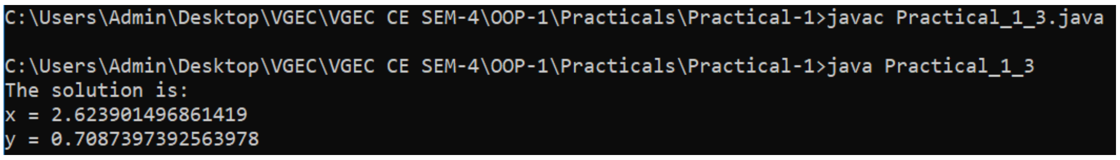

```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_2.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_2
Welcome to Java, Learning Java Now and Programming is Fun.
```

3. **Write a program that solves the following equation and displays the value x and y: a) 3.4x+50.2y=44.5 2) 2.1x+.55y=5.9 (Assume Cramer's rule to solve equation b) ax+by=e x=ed-bf/ad-bc cx+dy=f y=af-ec/ad-bc)**

Procedure:-

```
class Practical_1_3 {
  public static void main(String[] args) {
    // Coefficients of the system of equations
    double a = 3.4, b = 50.2, e = 44.5;
    double c = 2.1, d = 0.55, f = 5.9;

    // Determinant (denominator for both x and y)
    double determinant = a * d - b * c;

    // Check if the determinant is zero (system has no unique solution)
    if (determinant == 0) {
      System.out.println("The system has no unique solution.");
    } else {
      // Cramer's rule for x and y
      double x = (e * d - b * f) / determinant;
      double y = (a * f - e * c) / determinant;

      // Display the solution
      System.out.println("The solution is:");
      System.out.println("x = " + x);
      System.out.println("y = " + y);
    }
  }
}
```

Observation:-



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_3.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_3
The solution is:
x = 2.623901496861419
y = 0.7087397392563978
```

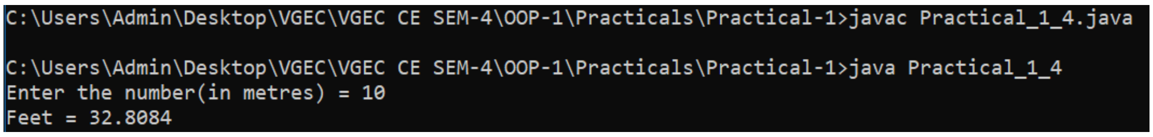
4. **Write a program that reads a number in meters, converts it to feet, and displays the result.**

Procedure:-

```
import java.util.Scanner;
public class Practical_1_4 {
  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the number(in metres) = ");
```

```
        int meter = sc.nextInt();

        //Conversion of meter to feet
        double feet = meter * 3.28084;

        System.out.println("Feet = " + feet);

        sc.close();
    }
}
```

Observation:-



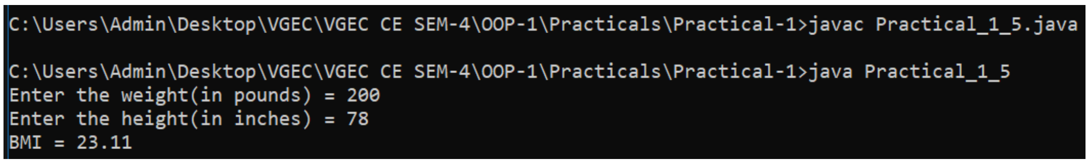
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_4.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_4
Enter the number(in metres) = 10
Feet = 32.8084
```

5. **Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters. Write a program that prompts the user to enter weight in pounds and height in inches and displays the BMI. Note:- 1 pound=.45359237 Kg and 1 inch=.0254 meters.**

Procedure:-

```
// 1 pound=.45359237 Kg and 1 inch=.0254 meters
import java.util.Scanner;
public class Practical_1_5 {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the weight(in pounds) = ");
        int pound = sc.nextInt();

        System.out.print("Enter the height(in inches) = ");
        int inch = sc.nextInt();

        //Conversion of meter to feet
        double meter = inch*0.0254;
        double kgs = pound*0.45359237;

        double bmi = kgs/(meter * meter);

        System.out.printf("BMI = %.2f%n" , bmi);

        sc.close();
    }
}
```

Observation:-



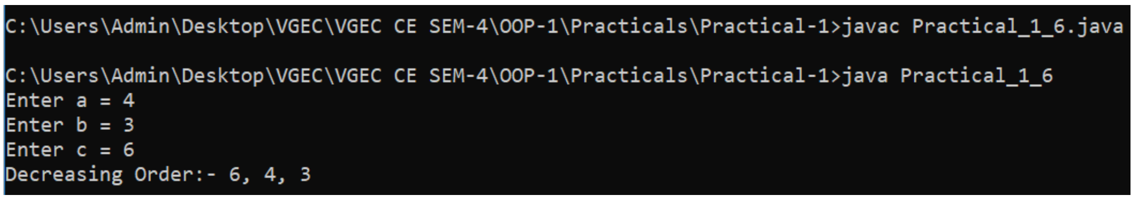
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_5.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_5
Enter the weight(in pounds) = 200
Enter the height(in inches) = 78
BMI = 23.11
```

6. **Write a program that prompts the user to enter three integers and display the integers in decreasing order.**

Procedure:-

```
import java.util.Scanner;
public class Practical_1_6 {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a = ");
        int a = sc.nextInt();
        System.out.print("Enter b = ");
        int b = sc.nextInt();
        System.out.print("Enter c = ");
        int c = sc.nextInt();

        int[] numbers = {a, b, c};

        for(int i=0; i<numbers.length-1; i++) {
            for(int j=i+1; j<numbers.length; j++) {
                if(numbers[i] < numbers[j]) {
                    int temp = numbers[i];
                    numbers[i] = numbers[j];
                    numbers[j] = temp;
                }
            }
        }

        System.out.println("Decreasing Order:- " + numbers[0] + ", " + numbers[1] + ", " +
numbers[2]);

        sc.close();
    }
}
```

Observation:-



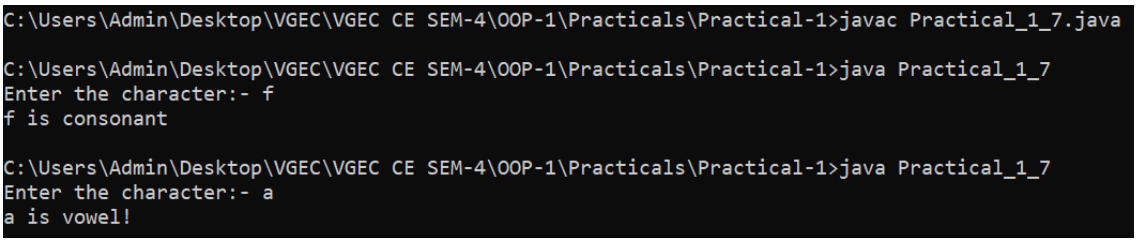
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_6.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_6
Enter a = 4
Enter b = 3
Enter c = 6
Decreasing Order:- 6, 4, 3
```

7. **Write a program that prompts the user to enter a letter and check whether a letter is a vowel or constant.**

Procedure:-

```
import java.util.Scanner;

public class Practical_1_7 {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the character:- ");
      char letter = sc.next().toLowerCase().charAt(0);

      // if(Character.isLetter(letter)) { // or
      if(letter >= 'a' && letter <= 'z') {
         if (letter == 'a' || letter == 'e' || letter == 'i' || letter == 'o' || letter == 'u') {
            System.out.println(letter + " is vowel!");
         } else {
            System.out.println(letter + " is consonant");
         }
      }

      sc.close();
   }
}
```

Observation:-



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_7.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_7
Enter the character:- f
f is consonant

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_7
Enter the character:- a
a is vowel!
```

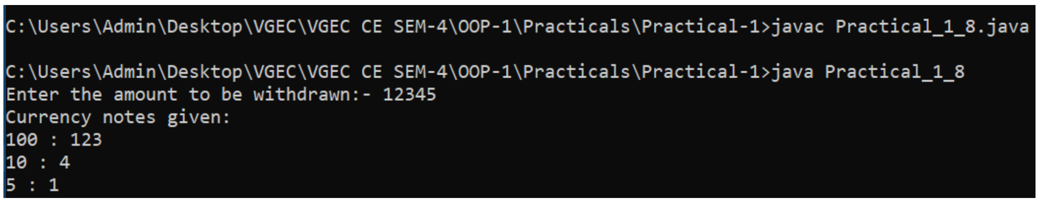
Additional programs:

8. **A cashier has currency notes of denominations 1, 2, 5, 10, 50 and 100. If the amount to be withdrawn is input through the keyboard, find the total number of currency notes of each denomination the cashier will have to give to the withdrawer.**

Procedure:-

```
import java.util.Scanner;
public class Practical_1_8 {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);

      //Available Denominations
      int[] denomination = {100, 50, 10, 5, 2, 1};

      System.out.print("Enter the amount to be withdrawn:- ");
      int amount = sc.nextInt();

      //Storing the note count of each denomination
      int[] noteCount = new int[denomination.length];

      //Calculation of the number of notes of each denomination
      for(int i=0; i<denomination.length; i++) {
         if(amount >= denomination[i]) {
            noteCount[i] = amount / denomination[i];
            amount %= denomination[i];
         }
      }

      System.out.println("Currency notes given:");

      for(int i=0; i<denomination.length; i++) {
         if(noteCount[i] > 0) {
            System.out.println(denomination[i] + " : " + noteCount[i]);
         }
      }

      sc.close();

   }
}
```

Observation:-



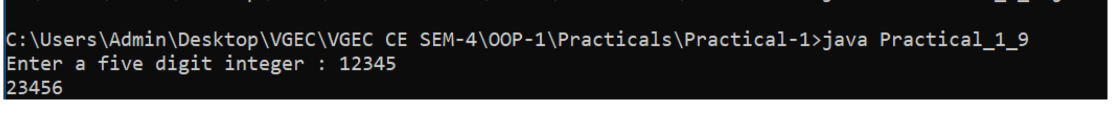
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_8.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_8
Enter the amount to be withdrawn:- 12345
Currency notes given:
100 : 123
10 : 4
5 : 1
```

9. **If a five-digit number is input through the keyboard, write a program to print a new number by adding one to each of its digits. For example, if the number that is input is 12391 then the output should be displayed as 23502.**

Procedure:-

```
import java.util.Scanner;
public class Practical_1_9 {
  public static void main(String args[]) {

    Scanner sc = new Scanner(System.in);

    System.out.print("Enter a five digit integer : ");
    int num = sc.nextInt();
    int[] rem = new int[5];

    if(num>10000 && num<99999){
      for(int i=4; i>=0; i--) {
        rem[i] = (num % 10) + 1;
        num = num / 10;
      }

      for(int i=0; i<5; i++) {
        System.out.print(rem[i]);
      }
    }
    else{
    System.out.println("Enter Valid Number!!!!!");
    }

    sc.close();

  }
}
```

Observation:-



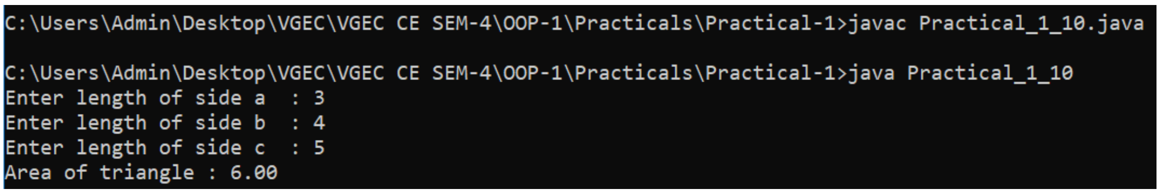
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_9.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_9
Enter a five digit integer : 12345
23456
```

10. If lengths of three sides of a triangle are input through the keyboard, write a program to print the area of the triangle.

Procedure:-

```
import java.util.Scanner;
import java.lang.Math;
public class Practical_1_10 {
   public static void main(String args[]) {

      Scanner sc = new Scanner(System.in);

      System.out.print("Enter length of side a  : ");
      double a = sc.nextDouble();
      System.out.print("Enter length of side b  : ");
      double b = sc.nextDouble();
      System.out.print("Enter length of side c  : ");
      double c = sc.nextDouble();

      if( a + b > c && a + c > b && b + c > a)
      {
         double s = (double) (a+b+c)/2;

         double area = Math.sqrt(s * (s - a) * (s - b) * (s - c));

         System.out.printf("Area of triangle : %.2f\n", area);
      }
      else{
         System.out.println("Given dimensions do not for a valid triangle!!");
      }

      sc.close();

   }
}
```

Observation:-



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_10.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_10
Enter length of side a  : 3
Enter length of side b  : 4
Enter length of side c  : 5
Area of triangle : 6.00
```

11. Write a program to produce the following patterns.

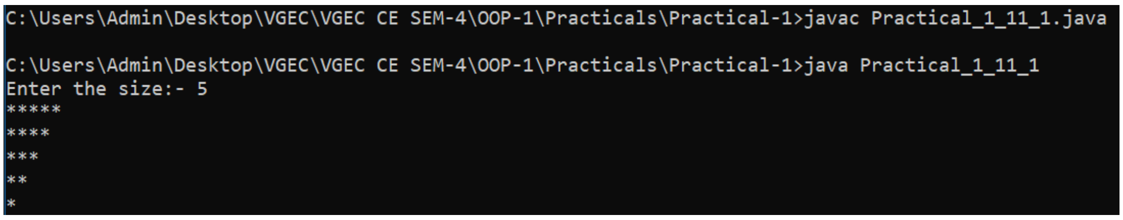
(A)

```
****
***
**
*
```

Procedure:-

```
import java.util.Scanner;
public class Practical_1_11_1 {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the size:- ");
      int n = sc.nextInt();

      for(int i=0; i<n; i++){
         for(int j=0; j<n-i; j++){
            System.out.print("*");
         }
         System.out.println();
      }

      sc.close();
   }
}
```

Observation:-



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_11_1.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_11_1
Enter the size:- 5
*****
****
***
**
*
```

(B)

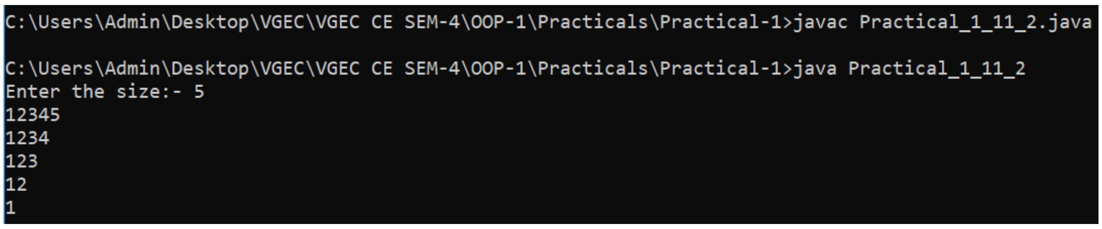
```
1234
123
12
1
```

Procedure:-

```
import java.util.Scanner;
public class Practical_1_11_2 {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the size:- ");
```

```
        int n = sc.nextInt();

        for(int i=0; i<n; i++){
            for(int j=1; j<=n-i; j++){
                System.out.print(j);
            }
            System.out.println();
        }

        sc.close();
    }
}
```

Observation:-



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_11_2.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_11_2
Enter the size:- 5
12345
1234
123
12
1
```

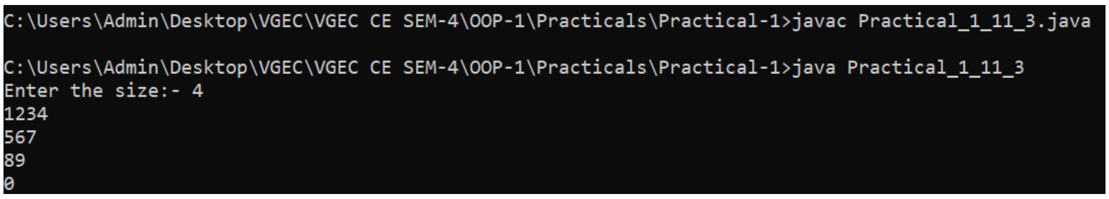
(C)

```
1234
567
89
0
```

Procedure:-

```
import java.util.Scanner;
public class Practical_1_11_3 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the size:- ");
        int n = s.nextInt();
        s.close();

        int x = 1;

        for(int i = 0; i < n; i++) {
            for(int j = 0; j < n - i; j++){
                System.out.print(x);
                x = (x + 1) % 10;
            }
            System.out.println("");
        }
    }
}
```

Observation:-


```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_11_3.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_11_3
Enter the size:- 4
1234
567
89
0
```

(D)

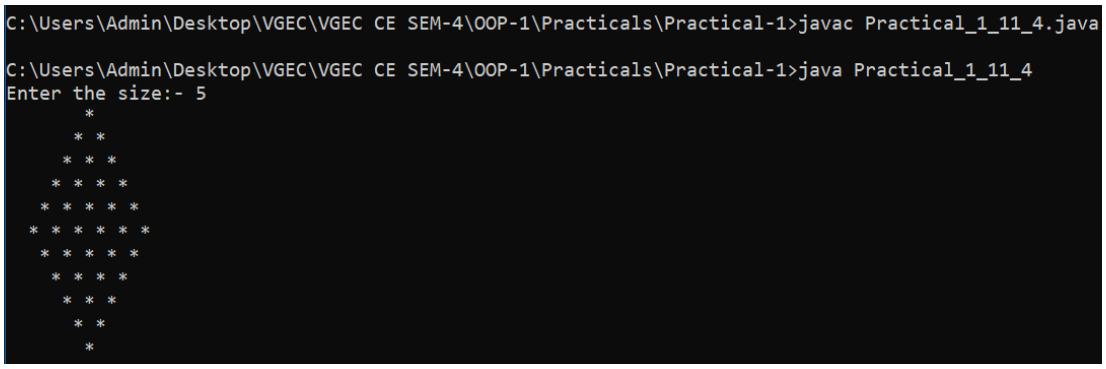
```
        *
       * *
      * * *
     * * * *
    * * * * *
     * * * *
      * * *
       * *
        *
```

Procedure:-

```
import java.util.Scanner;
public class Practical_1_11_4    {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the size:- ");
      int n = sc.nextInt();

      for(int i=0; i<n; i++){
         for(int k=n-i; k>=0; k--){
            System.out.print(" ");
         }
         for(int j=0; j<=i; j++){
            System.out.print(" *");
         }
         System.out.println();
      }

      for(int i=n; i>=0; i--){
         for(int k=n-i; k>=0; k--){
            System.out.print(" ");
         }
         for(int j=0; j<=i; j++){
            System.out.print(" *");
         }
         System.out.println();
      }
```

```
        sc.close();
    }
}
```

Observation:-



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>javac Practical_1_11_4.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-1>java Practical_1_11_4
Enter the size:- 5
       *
      * *
     * * *
    * * * *
   * * * * *
  * * * * * *
   * * * * *
    * * * *
     * * *
      * *
       *
```

Conclusion:

This experiment provided fundamental knowledge of Java programming constructs, essential for writing efficient and structured code. Understanding variables, operators, and control statements helps in building logical programs. Mastering these basics is crucial for progressing to advanced Java concepts like object-oriented programming and exception handling.

Quiz: (Sufficient space to be provided for the answers)

1. What is the primary purpose of Java??
2. What is the main method in Java used for?
3. How Java Language is Platform Independent?
4. What is JVM and JRE?

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students:

1. https://www.geeksforgeeks.org/java/
2. https://www.w3schools.com/java/

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition(2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	
Marks											

Signature of Faculty:

Experiment No: 2

AIM: To learn Arrays and Strings in Java.

Date:

CO mapped: CO-1

Objectives:

- **a)** Array manipulation: Learn how to create, populate, access, and modify arrays in Java.
- **b)** String manipulation: Understand how to create and manipulate strings, including concatenation, comparison, and extraction of substrings.
- **c)** Array and String methods: Explore common array and string methods available in Java's standard library.

Background:

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.
Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on the 1st index, and so on.
There are two types of array.

- Single Dimensional Array
- Multidimensional Array

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. For example:
char[] ch={'j','a','v','a','t','p','o','i','n','t'};
String s=new String(ch);
is same as:
String s="javatpoint";
Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

Practical questions:

1. **Write a program that generate 6*6 two-dimensional matrix, filled with 0's and 1's , display the matrix, check every raw and column have an odd number's of 1's.**

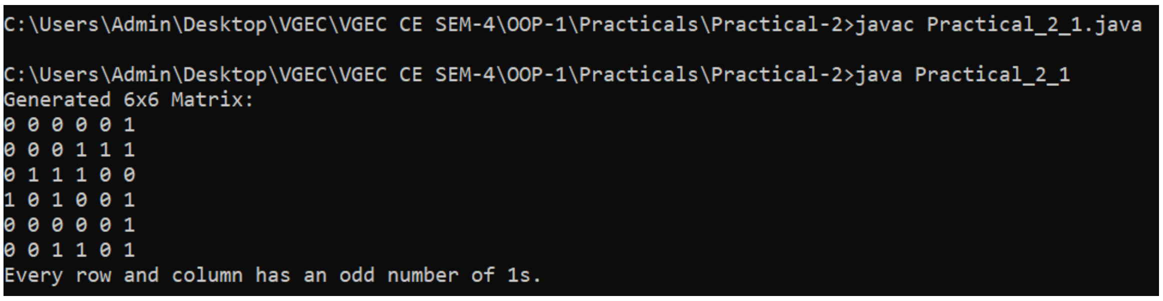
Procedure:

```
import java.util.Random;
public class Practical_2_1 {
   public static void main(String args[]) {
      int size = 6;
      int[][] matrix = new int[size][size];
      Random rand = new Random();
```

```
// Step 1: Fill the matrix row-wise ensuring odd number of 1s in each row
for (int i = 0; i < size; i++) {
   int onesCount = 0;
   for (int j = 0; j < size - 1; j++) {
      matrix[i][j] = rand.nextInt(2); // Randomly assign 0 or 1
      if (matrix[i][j] == 1) {
         onesCount++;
      }
   }
   // Ensure odd number of 1s in the row
   matrix[i][size - 1] = (onesCount % 2 == 0) ? 1 : 0;
}

// Step 2: Adjust columns to ensure each has an odd number of 1s
for (int j = 0; j < size; j++) {
   int onesCount = 0;
   for (int i = 0; i < size; i++) {
      if (matrix[i][j] == 1) {
         onesCount++;
      }
   }
   // If the count is even, flip the last element in the column
   if (onesCount % 2 == 0) {
      matrix[size - 1][j] = (matrix[size - 1][j] == 1) ? 0 : 1;
   }
}

// Step 3: Display the matrix
System.out.println("Generated 6x6 Matrix:");
for (int i = 0; i < size; i++) {
   for (int j = 0; j < size; j++) {
      System.out.print(matrix[i][j] + " ");
   }
   System.out.println();
}

// Step 4: Verify every row and column has an odd number of 1s
boolean isValid = true;
for (int i = 0; i < size; i++) {
   int rowCount = 0, colCount = 0;
   for (int j = 0; j < size; j++) {
      if (matrix[i][j] == 1) rowCount++;
      if (matrix[j][i] == 1) colCount++;
   }
   if (rowCount % 2 == 0 || colCount % 2 == 0) {
      isValid = false;
      break;
   }
}
```

```
        // Step 5: Display final result
        if (isValid) {
            System.out.println("Every row and column has an odd number of 1s.");
        } else {
            System.out.println("Some rows or columns do not have an odd number of 1s.");
        }
    }
}
```

Observations:



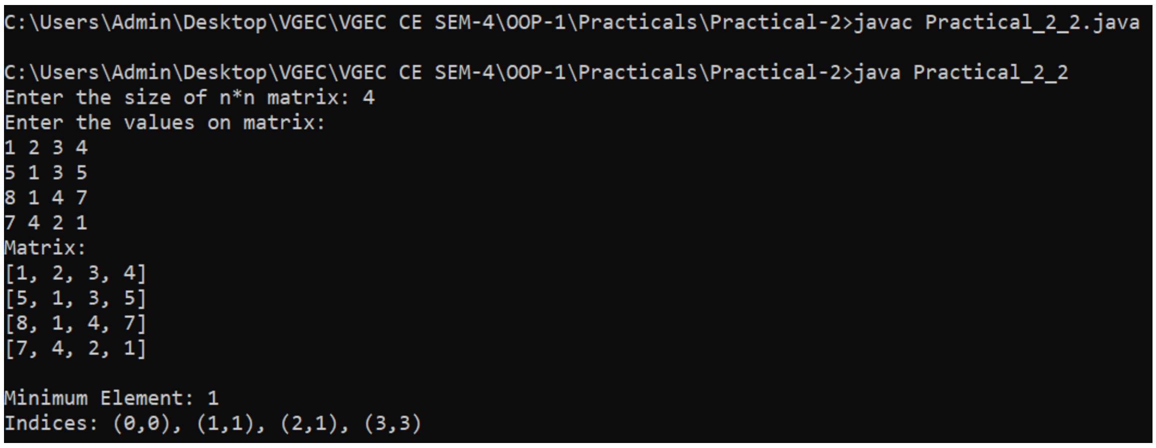
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>javac Practical_2_1.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>java Practical_2_1
Generated 6x6 Matrix:
0 0 0 0 0 1
0 0 0 1 1 1
0 1 1 1 0 0
1 0 1 0 0 1
0 0 0 0 0 1
0 0 1 1 0 1
Every row and column has an odd number of 1s.
```

2. **Write a generic method that returns the minimum elements and their indices in a two dimensional array.**

Procedure:

```
import java.util.Arrays;
import java.util.Scanner;
public class Practical_2_2 {

    public static String findMinAndIndices(int[][] matrix) {
        int min = Integer.MAX_VALUE;
        String indices = ""; // To store the indices as a string

        //Find the minimum element and track its indices
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                if (matrix[i][j] < min) {
                    min = matrix[i][j];
                    indices = "(" + i + "," + j + ")"; // Reset string if new min found
                } else if (matrix[i][j] == min) {
                    indices = indices.concat(", (" + i + "," + j + ")"); // Append if another min found
                }
            }
        }

        // Step 2: Return the result as a formatted string
        return "Minimum Element: " + min + "\nIndices: " + indices;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of n*n matrix: ");
        int size = sc.nextInt();
        int[][] matrix = new int[size][size];

        //Taking input of the matrix
        System.out.println("Enter the values on matrix: ");
        for(int i=0; i<size; i++) {
            for(int j=0; j<size; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }

        // Display the matrix
        System.out.println("Matrix:");
        for (int[] row : matrix) {
            System.out.println(Arrays.toString(row));
        }

        // Get minimum element and its indices
        String result = findMinAndIndices(matrix);
        System.out.println("\n" + result);

        sc.close();
    }
}
```

Observations:



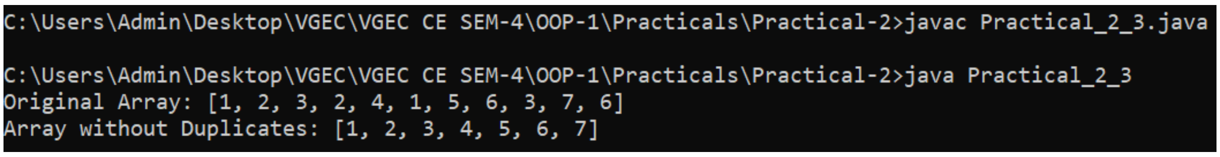
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>javac Practical_2_2.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>java Practical_2_2
Enter the size of n*n matrix: 4
Enter the values on matrix:
1 2 3 4
5 1 3 5
8 1 4 7
7 4 2 1
Matrix:
[1, 2, 3, 4]
[5, 1, 3, 5]
[8, 1, 4, 7]
[7, 4, 2, 1]

Minimum Element: 1
Indices: (0,0), (1,1), (2,1), (3,3)
```

3. **Write a method that returns a new array by eliminating the duplicate values in the array.**

Procedure:

```
import java.util.Arrays;
public class Practical_2_3 {
    public static int[] removeDuplicates(int[] arr) {
        int n = arr.length;
        int[] temp = new int[n]; // Temporary array to store unique elements
        int newSize = 0;
        // Step 1: Iterate and check for unique values
        for (int i = 0; i < n; i++) {
            boolean isDuplicate = false;

            // Check if the value already exists in temp
            for (int j = 0; j < newSize; j++) {
                if (arr[i] == temp[j]) {
                    isDuplicate = true;
                    break;
                }
            }

            // If it's not a duplicate, add it to temp
            if (!isDuplicate) {
                temp[newSize] = arr[i];
                newSize++;
            }
        }

        // Step 2: Copy unique values to the final array
        int[] uniqueArray = Arrays.copyOf(temp, newSize);
        return uniqueArray;
    }

    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 2, 4, 1, 5, 6, 3, 7, 6}; // Sample input array

        // Display original array
        System.out.println("Original Array: " + Arrays.toString(numbers));

        // Remove duplicates
        int[] uniqueNumbers = removeDuplicates(numbers);

        // Display unique array
        System.out.println("Array without Duplicates: " + Arrays.toString(uniqueNumbers));
    }
}
```

Observations:



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>javac Practical_2_3.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>java Practical_2_3
Original Array: [1, 2, 3, 2, 4, 1, 5, 6, 3, 7, 6]
Array without Duplicates: [1, 2, 3, 4, 5, 6, 7]
```

4. **Write a program to add, subtract or multiply two 3*3 integer arrays as per choice of user.**

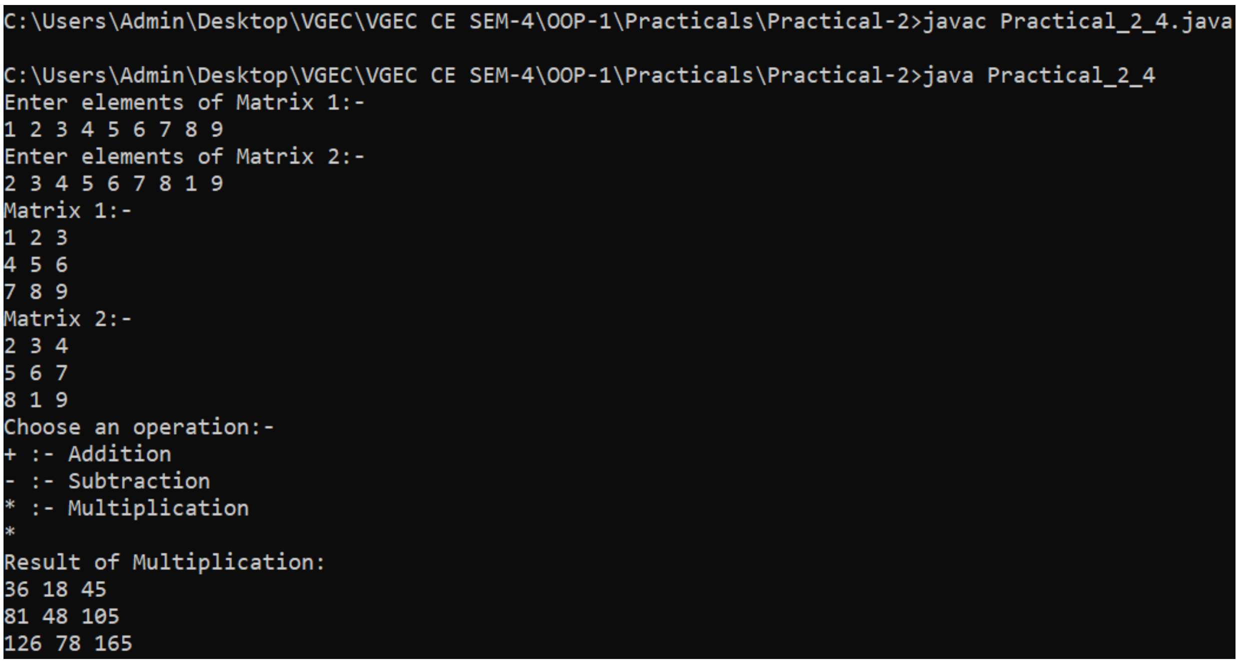
   Sample Input:
   Array 1:
   1 2 3
   4 5 6
   7 8 9
   Array 2:
   5 6 7
   1 2 0
   4 3 2
   Symbol: +
   Sample Output:
   6 8 10
   5 7 6
   11 11 11

Procedure:

```
import java.util.Scanner;
public class Practical_2_4 {
   public static int[][] addMatrices(int[][] matrix1, int[][] matrix2) {
      int[][] result = new int[3][3];
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
         }
      }
      return result;
   }

   public static int[][] subtractMatrices(int[][] matrix1, int[][] matrix2) {
      int[][] result = new int[3][3];
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            result[i][j] = matrix1[i][j] - matrix2[i][j];
         }
      }
      return result;
   }

   public static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2) {
      int[][] result = new int[3][3];
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            result[i][j] = 0;
            for (int k = 0; k < 3; k++) {
               result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
         }
      }
```

```
        return result;
    }

    public static void printMatrix(int[][] matrix) {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int[][] matrix1 = new int[3][3];
        int[][] matrix2 = new int[3][3];
        int[][] result = new int[3][3];

        //Enter data in Marix 1
        System.out.println("Enter elements of Matrix 1:-");
        for(int i=0; i<3; i++) {
            for(int j=0; j<3; j++) {
                matrix1[i][j] = sc.nextInt();
            }
        }

        //Enter data in Marix 2
        System.out.println("Enter elements of Matrix 2:-");
        for(int i=0; i<3; i++) {
            for(int j=0; j<3; j++) {
                matrix2[i][j] = sc.nextInt();
            }
        }

        //Displaying Matrix 1
        System.out.println("Matrix 1:-");
        for(int i=0; i<3; i++) {
            for(int j=0; j<3; j++) {
                System.out.print(matrix1[i][j] + " ");
            }
            System.out.println();
        }

        //Displaying Matrix 2
        System.out.println("Matrix 2:-");
        for(int i=0; i<3; i++) {
            for(int j=0; j<3; j++) {
                System.out.print(matrix2[i][j] + " ");
            }
            System.out.println();
        }

        //Choosing Matrix Operation to be done
        System.out.println("Choose an operation:- \n+ :- Addition\n- :- Subtraction\n* :-
Multiplication");
        int choice = sc.next().charAt(0);
        switch (choice) {
            case '+':
                result = addMatrices(matrix1, matrix2);
                System.out.println("Result of Addition:");
                break;
            case '-':
                result = subtractMatrices(matrix1, matrix2);
                System.out.println("Result of Subtraction:");
                break;
            case '*':
                result = multiplyMatrices(matrix1, matrix2);
                System.out.println("Result of Multiplication:");
                break;
            default:
                System.out.println("Invalid choice!");
                break;
            }
        printMatrix(result);
        sc.close();
    }
}
```

Observations:



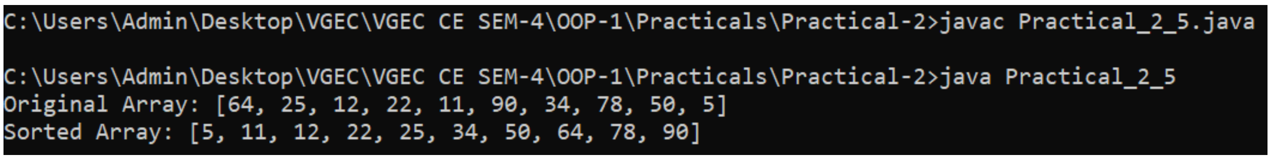
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>javac Practical_2_4.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>java Practical_2_4
Enter elements of Matrix 1:-
1 2 3 4 5 6 7 8 9
Enter elements of Matrix 2:-
2 3 4 5 6 7 8 1 9
Matrix 1:-
1 2 3
4 5 6
7 8 9
Matrix 2:-
2 3 4
5 6 7
8 1 9
Choose an operation:-
+ :- Addition
- :- Subtraction
* :- Multiplication
*
Result of Multiplication:
36 18 45
81 48 105
126 78 165
```

5. **Write a program to sort an array of 10 elements using selection sort.**

Procedure:

```
import java.util.Arrays;
public class Practical_2_5 {
   public static void selectionSort(int[] arr) {
      int n = arr.length; // Get the length of the array

      // Outer loop: Moves the boundary of the unsorted part
      for (int i = 0; i < n - 1; i++) {
         int minIndex = i;

         // Inner loop: Finds the smallest element in the remaining unsorted array
         for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
               minIndex = j;
            }
         }

         // Swap the found minimum element with the first element of the unsorted part
         int temp = arr[minIndex];
         arr[minIndex] = arr[i];
         arr[i] = temp;
      }
   }
   public static void main(String[] args) {
      int[] numbers = {64, 25, 12, 22, 11, 90, 34, 78, 50, 5};

      // Display original array
      System.out.println("Original Array: " + Arrays.toString(numbers));

      selectionSort(numbers);

      // Display sorted array
      System.out.println("Sorted Array: " + Arrays.toString(numbers));
   }
}
```

Observations:



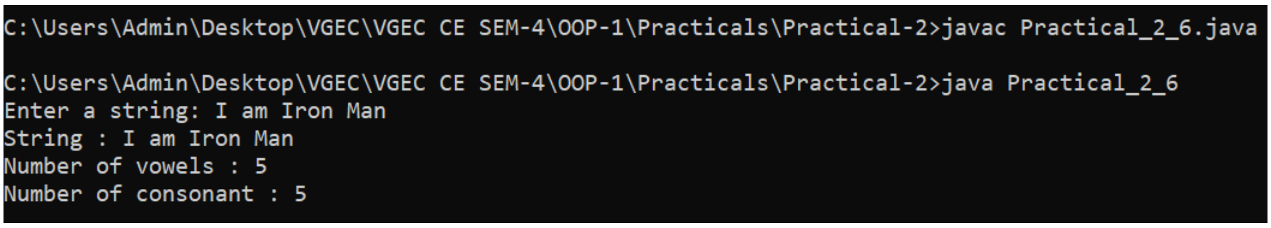
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>javac Practical_2_5.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>java Practical_2_5
Original Array: [64, 25, 12, 22, 11, 90, 34, 78, 50, 5]
Sorted Array: [5, 11, 12, 22, 25, 34, 50, 64, 78, 90]
```

6. **Write a program that prompts the user to enter a string and displays the number of vowels and consonants in the string.**

Procedure:

```
import java.util.Scanner;
public class Practical_2_6 {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a string: ");
      String str = sc.nextLine();

      System.out.println("String : " + str);

      int vowelCount = 0;
      int consonantCount = 0;
      for(int i=0; i<str.length(); i++) {
                                 if(str.toLowerCase().charAt(i)=='a'     ||     str.toLowerCase().charAt(i)=='e'
||str.toLowerCase().charAt(i)=='i'                                        ||str.toLowerCase().charAt(i)=='o'
||str.toLowerCase().charAt(i)=='u') {
            vowelCount++;
         }
         else if(str.toLowerCase().charAt(i) != ' '){
            consonantCount++;
         }
      }

      System.out.println("Number of vowels : " + vowelCount);
      System.out.println("Number of consonant : " + consonantCount);

      sc.close();
   }
}
```

Observations:



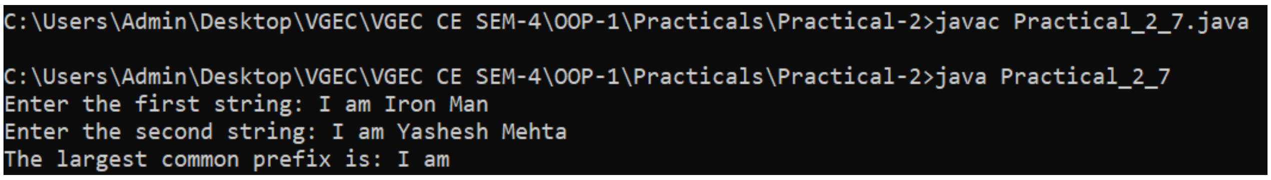
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>javac Practical_2_6.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>java Practical_2_6
Enter a string: I am Iron Man
String : I am Iron Man
Number of vowels : 5
Number of consonant : 5
```

7. **Write a program that prompts the user to enter two strings and displays the largest common prefix of the two strings.**

Procedure:

```
import java.util.Scanner;
public class Practical_2_7 {
   public static String findCommonPrefix(String str1, String str2) {
      int minLength = Math.min(str1.length(), str2.length());
      StringBuilder prefix = new StringBuilder();

      for (int i = 0; i < minLength; i++) {
         if (str1.charAt(i) == str2.charAt(i)) {
            prefix.append(str1.charAt(i));
         } else {
            break;
         }
      }

      return prefix.toString();
   }

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

      // Prompt user to enter two strings
      System.out.print("Enter the first string: ");
      String firstString = scanner.nextLine();
      System.out.print("Enter the second string: ");
      String secondString = scanner.nextLine();

      String commonPrefix = findCommonPrefix(firstString, secondString);

      // Display the result
      if (!commonPrefix.isEmpty()) {
         System.out.println("The largest common prefix is: " + commonPrefix);
      } else {
         System.out.println("No common prefix found.");
      }

      scanner.close();
   }
}
```

Observations:



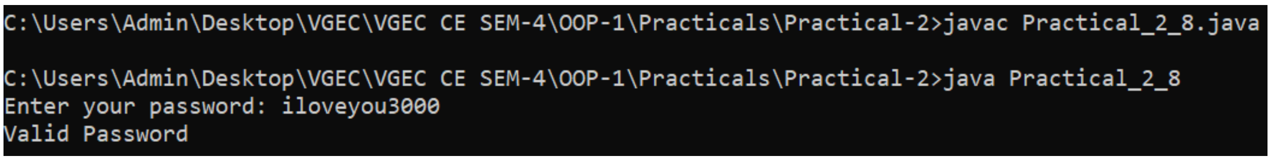
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>javac Practical_2_7.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>java Practical_2_7
Enter the first string: I am Iron Man
Enter the second string: I am Yashesh Mehta
The largest common prefix is: I am
```

8. Some websites impose certain rules for passwords. Write a method that checks whether a string is a valid password. Suppose the password rules are as follows: A password must have at least eight characters. A password consists of only letters and digits. A password must contain at least two digits. Write a program that prompts the user to enter a password and displays Valid Password if the rules are followed or Invalid Password otherwise.

Procedure:

```
import java.util.Scanner;
public class Practical_2_8 {
  public static boolean isValidPassword(String password) {
    // Rule 1: At least 8 characters long
    if (password.length() < 8) {
      return false;
    }

    int digitCount = 0;

    // Rule 2 & 3: Only letters and digits, and at least 2 digits
    for (int i = 0; i < password.length(); i++) {
      char ch = password.charAt(i);

      if (Character.isDigit(ch)) {
        digitCount++;
      } else if (!Character.isLetter(ch)) {
        return false;
      }
    }
    // Password is valid if it has at least 2 digits
    return digitCount >= 2;
  }

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

    // Prompt the user for a password
    System.out.print("Enter your password: ");
    String password = sc.nextLine();

    // Check if the password is valid
    if (isValidPassword(password)) {
      System.out.println("Valid Password");
    } else {
      System.out.println("Invalid Password");
    }
    sc.close();
  }
}
```

Observation:



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>javac Practical_2_8.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-2>java Practical_2_8
Enter your password: iloveyou3000
Valid Password
```

Conclusion:
In this practical session, we explored the fundamentals of arrays and strings in Java, focusing on their creation, manipulation, and built-in methods. We learned how to declare, initialize, access, and modify arrays, along with performing operations like sorting and searching. Additionally, we understood string handling, including concatenation, comparison, and substring extraction. By utilizing Java's standard library methods, we efficiently worked with both arrays and strings, enhancing our problem-solving and programming skills.

Quiz: (Sufficient space to be provided for the answers)

1. What are ragged arrays in java and how are they implemented?
2. Differentiate String class and StringBuffer class.
3. How Create a two dimensional array. Instantiate and Initialize it?
4. Explain the various String functions with their syntax.

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students: (Sufficient space to be provided)

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition(2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	
Marks											

Signature of Faculty:

Experiment No: 3

AIM: To implement basic object-oriented concepts.

Date:

CO mapped: CO-2

Objectives:

a) To apply fundamental object-oriented principles, such as class design, encapsulation, inheritance, and polymorphism, to improve software modularity, code organization, and maintainability.

b) Implementing these basic object-oriented concepts in your software development practices will help you create more structured, maintainable, and reusable code, which is essential for building robust and scalable software systems.

Background:

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

- Object: Any entity that has a state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical. An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.
- Class: Collection of objects is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.
- Inheritance: When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
- Polymorphism: If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc. In Java, we use method overloading and method overriding to achieve polymorphism. Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.
- Abstraction: Hiding internal details and showing functionality is known as abstraction. For example, phone call, we don't know the internal processing. In Java, we use abstract class and interface to achieve abstraction.
- Encapsulation: Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines. A java class is an example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Practical questions:

1. **Write a Java application which takes several command line arguments, which are supposed to be names of students and prints output as given below: (Suppose we enter 3 names then output should be as follows):**

Number of arguments = 3

1: First Student Name is =Tom

2: Second Student Name is =Dick

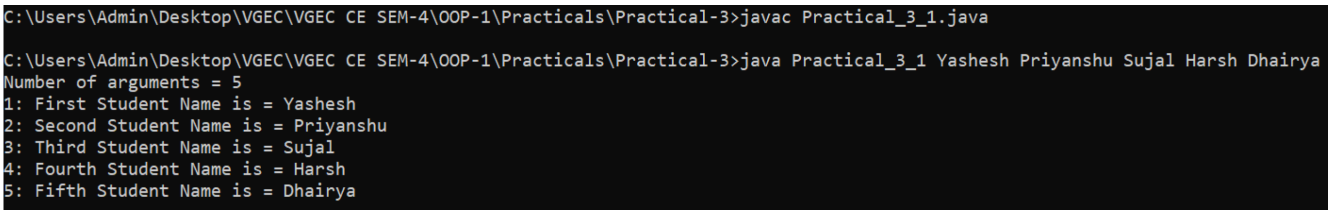
3: Third Student Name is =Harry

(Hint: An array may be used for converting from numeric values from 1 to 20 into String.)

Procedure:

```
public class Practical_3_1 {
    public static void main(String[] args) {
        // Array for number-to-word conversion (1 to 20)
        String[] numberWords = {"", "First", "Second", "Third", "Fourth", "Fifth", "Sixth", "Seventh",
"Eighth", "Ninth", "Tenth",
                                "Eleventh", "Twelfth", "Thirteenth", "Fourteenth", "Fifteenth", "Sixteenth",
"Seventeenth",
                                "Eighteenth", "Nineteenth", "Twentieth"};

        int count = args.length;
        System.out.println("Number of arguments = " + count);

        for (int i = 0; i < count; i++) {
            if (i + 1 <= 20) { // Ensure index is within bounds
                System.out.println((i + 1) + ": " + numberWords[i + 1] + " Student Name is = " + args[i]);
            } else {
                System.out.println((i + 1) + ": Student Name is = " + args[i]); // Default if more than 20
            }
        }
    }
}
```

Observations:



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>javac Practical_3_1.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>java Practical_3_1 Yashesh Priyanshu Sujal Harsh Dhairya
Number of arguments = 5
1: First Student Name is = Yashesh
2: Second Student Name is = Priyanshu
3: Third Student Name is = Sujal
4: Fourth Student Name is = Harsh
5: Fifth Student Name is = Dhairya
```

2. **Design a class named Rectangle to represent a rectangle. The class contains: Two double data fields named width and height that specify the width and height of the rectangle. The default values are 1 for both width and height.**

A no-arg constructor that creates a default rectangle.

A constructor that creates a rectangle with the specified width and height.

A method named getArea() that returns the area of this rectangle.

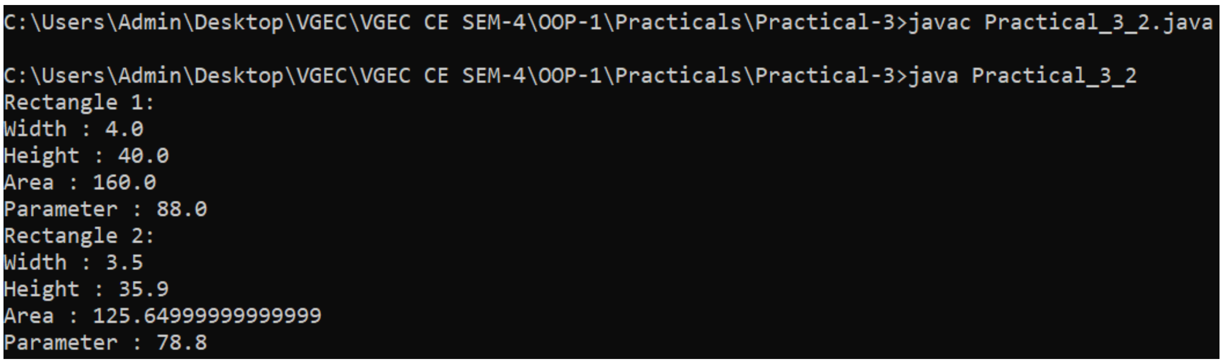
A method named getPerimeter() that returns the perimeter.

Write a test program that creates two Rectangle objects—one with width 4 and height 40 and the other with width 3.5 and height 35.9. Display the width, height, area, and perimeter of each rectangle in this order.

Procedure:

```
class Rectangle {
   double width;
   double height;

   //Default Constructor
   Rectangle() {
      width = height = 1;
   }

   //Parametrized Constructor
   Rectangle(double width, double height) {
      this.width = width;
      this.height = height;
   }

   double getArea() {
      return width * height;
   }

   double getParameter() {
      return 2 * (width + height);
   }

   double getWidth() {
      return width;
   }

   double getHeight() {
      return height;
   }
}

public class Practical_3_2 {
   public static void main(String args[]) {
      Rectangle rect1 = new Rectangle(4, 40);
      Rectangle rect2 = new Rectangle(3.5, 35.9);

      //Rectangle 1
      System.out.println("Rectangle 1:");
      System.out.println("Width : " + rect1.getWidth());
      System.out.println("Height : " + rect1.getHeight());
      System.out.println("Area : " + rect1.getArea());
      System.out.println("Parameter : " + rect1.getParameter());

      //Rectangle 2
      System.out.println("Rectangle 2:");
      System.out.println("Width : " + rect2.getWidth());
      System.out.println("Height : " + rect2.getHeight());
      System.out.println("Area : " + rect2.getArea());
      System.out.println("Parameter : " + rect2.getParameter());

   }
}
```

Observations:



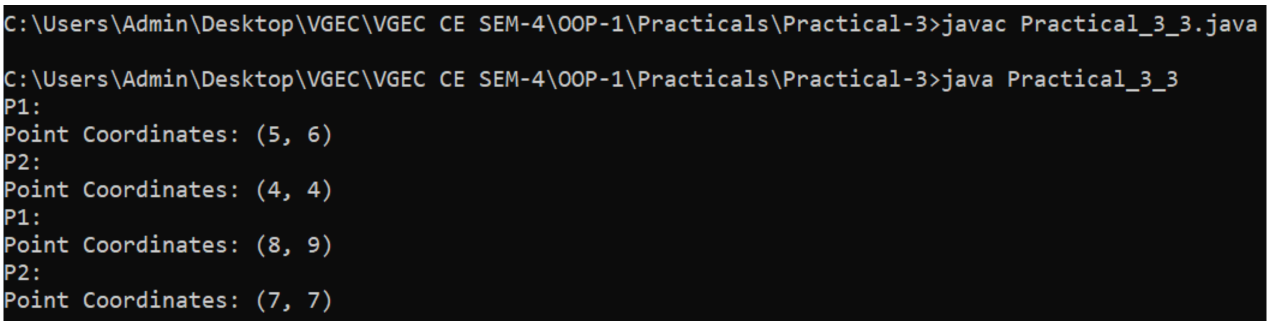
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>javac Practical_3_2.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>java Practical_3_2
Rectangle 1:
Width : 4.0
Height : 40.0
Area : 160.0
Parameter : 88.0
Rectangle 2:
Width : 3.5
Height : 35.9
Area : 125.64999999999999
Parameter : 78.8
```

3. **Define a class called Cartesian Point, which has two instance variables, x and y. Provide the methods get X() and get Y() to return the values of the x and y values respectively, a method called move() which would take two integers as parameters and change the values of x and y respectively, a method called display() which would display the current values of x and y. Now overload the method move() to work with single parameter, which would set both x and y to the same values,provide constructors with two parameters and overload to work with one parameter as well. Now define a class called Test Cartesian Point, with the main method to test the various methods in the Cartesian Point class.**

Procedure:

```
class CartesianPoint {
    int x;
    int y;

    CartesianPoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    CartesianPoint(int value) {
        this.x = value;
        this.y = value;
    }

    int getX() {
        return x;
    }

    int getY() {
        return y;
    }

    void move(int newX, int newY) {
        this.x = newX;
        this.y = newY;
    }

    void move(int newValue) {
        this.x = newValue;
        this.y = newValue;
    }

    void display() {
        System.out.println("Point Coordinates: (" + x + ", " + y + ")");
    }

}

public class Practical_3_3 {
    public static void main(String[] args) {
        CartesianPoint p1 = new CartesianPoint(5, 6);
        CartesianPoint p2 = new CartesianPoint(4);

        //Initial Values
        System.out.println("P1: ");
        p1.display();
        System.out.println("P2: ");
        p2.display();

        //Assigning new values
        p1.move(8, 9);
        p2.move(7);

        //New Values
        System.out.println("P1: ");
        p1.display();
        System.out.println("P2: ");
        p2.display();
    }
}
```

Observations:



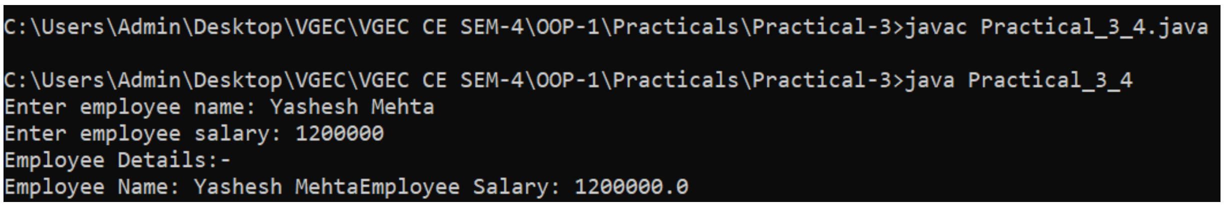
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>javac Practical_3_3.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>java Practical_3_3
P1:
Point Coordinates: (5, 6)
P2:
Point Coordinates: (4, 4)
P1:
Point Coordinates: (8, 9)
P2:
Point Coordinates: (7, 7)
```

4. **Create a class Employee which has two private data members name and salary and it has two public member functions named as getData() and putData() where getData() gets name and salary from the user putData() displays name and salary for any user.**

Procedure:

```
import java.util.Scanner;
class Employee {
    private String name;
    private double salary;

    void getData() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter employee name: ");
        name = sc.nextLine();
        System.out.print("Enter employee salary: ");
        salary = sc.nextDouble();
        sc.close();
    }

    void putData() {
        System.out.println("Employee Name: " + name);
        System.out.println("Employee Salary: " + salary);
    }
}
public class Practical_3_4 {
    public static void main(String[] args) {
        Employee emp = new Employee();
        emp.getData();
        System.out.println("Employee Details:-");
        emp.putData();
    }
}
```

Observations:



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>javac Practical_3_4.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>java Practical_3_4
Enter employee name: Yashesh Mehta
Enter employee salary: 1200000
Employee Details:-
Employee Name: Yashesh MehtaEmployee Salary: 1200000.0
```

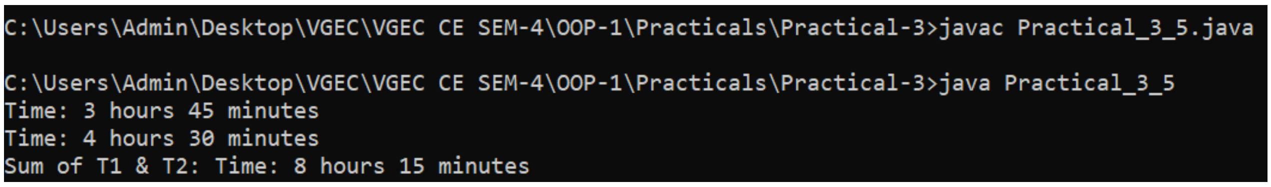
5. **Define a class Time with hours and minutes astwo data members, add necessary member functions to initialize and display data of class. Do not use constructors in a class. Define a member function sum () which adds two Time objects. (Use the statements like T3.sum (T1, T2)).**

Procedure:

```
class Time {
   int hours;
   int minutes;

   void setTime(int h, int m) {
      this.hours = h;
      this.minutes = m;
   }

   void display() {
      System.out.println("Time: " + hours + " hours " + minutes + " minutes");
   }

   void sum(Time t1, Time t2) {
      this.minutes = t1.minutes + t2.minutes;
      this.hours = t1.hours + t2.hours + (this.minutes / 60);
      this.minutes = this.minutes % 60;
   }
}

public class Practical_3_5 {
   public static void main(String[] args) {
      Time T1 = new Time();
      Time T2 = new Time();
      Time T3 = new Time();

      T1.setTime(3, 45);
      T1.display();
      T2.setTime(4, 30);
      T2.display();

      T3.sum(T1, T2);

      System.out.print("Sum of T1 & T2: ");
      T3.display();
```

```
    }
}
```

Observations:



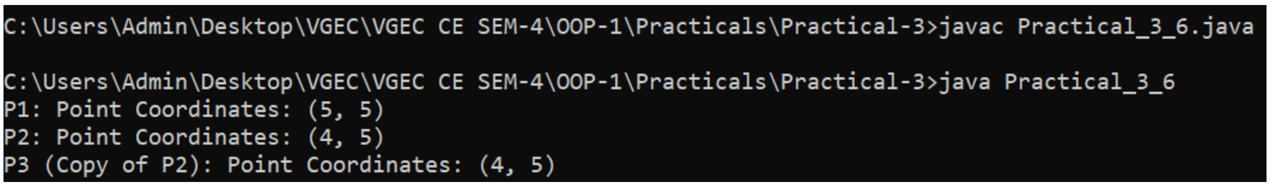
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>javac Practical_3_5.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>java Practical_3_5
Time: 3 hours 45 minutes
Time: 4 hours 30 minutes
Sum of T1 & T2: Time: 8 hours 15 minutes
```

6. **Define Class named Point which represents 2-D Point, i.e. P (x, y). Define Default constructor to initialize both data member value 5, Parameterized constructor to initialize member according to value supplied by user and Copy Constructor. Define Necessary Function and Write a program to test class Point.**

Procedure:

```
class Point {
    int x;
    int y;

    Point() {
        x = y = 5;
    }

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    Point(Point p) {
        this.x = p.x;
        this.y = p.y;
    }

    void display() {
        System.out.println("Point Coordinates: (" + x + ", " + y + ")");
    }
}

public class Practical_3_6 {
    public static void main(String[] args) {
        Point p1 = new Point();
        Point p2 = new Point(4, 5);
        Point p3 = new Point(p2);

        System.out.print("P1: ");
        p1.display();
        System.out.print("P2: ");
        p2.display();
        System.out.print("P3 (Copy of P2): ");
        p3.display();
    }
}
```

Observations:



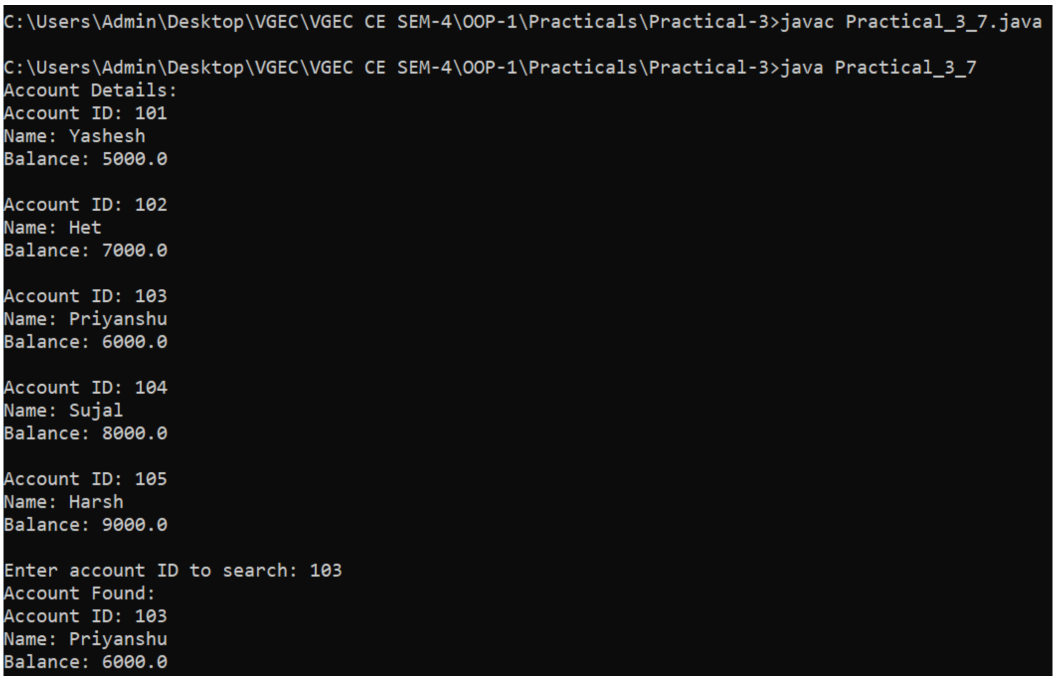
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>javac Practical_3_6.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>java Practical_3_6
P1: Point Coordinates: (5, 5)
P2: Point Coordinates: (4, 5)
P3 (Copy of P2): Point Coordinates: (4, 5)
```

7. **Create a class Account. It has three data member account id, name and balance. Define function to assign value and display value. Define function that search account number given by the user. If account number exists, print detail of that account. Write a program using array of object. Declare at least 5 account and print details.**

Procedure:

```
import java.util.Scanner;
class Account {
    int accountId;
    String name;
    double balance;
    void setAccount(int accountId, String name, double balance) {
        this.accountId = accountId;
        this.name = name;
        this.balance = balance;
    }
    void display() {
        System.out.println("Account ID: " + accountId);
        System.out.println("Name: " + name);
        System.out.println("Balance: " + balance);
    }
    int getAccountId() {
        return accountId;
    }
}
public class Practical_3_7 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Account[] accounts = new Account[5];
        accounts[0] = new Account();
        accounts[0].setAccount(101, "Yashesh", 5000.0);
        accounts[1] = new Account();
        accounts[1].setAccount(102, "Het", 7000.0);
        accounts[2] = new Account();
        accounts[2].setAccount(103, "Priyanshu", 6000.0);
        accounts[3] = new Account();
        accounts[3].setAccount(104, "Sujal", 8000.0);
        accounts[4] = new Account();
        accounts[4].setAccount(105, "Harsh", 9000.0);
        System.out.println("Account Details:");
        for (Account acc : accounts) {
            acc.display();
            System.out.println();
        }
        System.out.print("Enter account ID to search: ");
        int searchId = sc.nextInt();
        boolean found = false;
        for (Account acc : accounts) {
            if (acc.getAccountId() == searchId) {
                System.out.println("Account Found:");
                acc.display();
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("Account not found.");
        }
        sc.close();
    }
}
```

Observations:



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>javac Practical_3_7.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-3>java Practical_3_7
Account Details:
Account ID: 101
Name: Yashesh
Balance: 5000.0

Account ID: 102
Name: Het
Balance: 7000.0

Account ID: 103
Name: Priyanshu
Balance: 6000.0

Account ID: 104
Name: Sujal
Balance: 8000.0

Account ID: 105
Name: Harsh
Balance: 9000.0

Enter account ID to search: 103
Account Found:
Account ID: 103
Name: Priyanshu
Balance: 6000.0
```

Conclusion:
In conclusion, implementing basic object-oriented concepts such as class design, encapsulation, inheritance, and polymorphism enhances software modularity, organization, and maintainability. By applying these principles, developers can create structured, reusable, and scalable code, leading to more efficient and robust software systems. Mastering these fundamentals is essential for building high-quality applications that can be easily maintained and extended over time.

Quiz: (Sufficient space to be provided for the answers)

1. Explain the concept of encapsulation and why it is important in OOP.
2. What is a class, and how does it relate to objects in OOP?
3. Define and explain static and dynamic binding.
4. Explain the concept of method overloading and method overriding in OOP.

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students:
1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/

Rubric wise marks obtained:

Rubrics	**Knowledge (2)**		**Problem Recognition(2)**		**Logic Building (2)**		**Completeness and accuracy (2)**		**Ethics (2)**		**Total**
	Good (2)	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	
Marks											

Signature of Faculty:

Experiment No: 4

AIM: To implement inheritance and object-oriented concepts.

Date:

CO mapped: CO-2

Objectives:

a) To master the fundamental principles of inheritance and object-oriented concepts, enabling the design and development of efficient, maintainable, and scalable software solutions by leveraging the power of class hierarchies and code reuse.
b) Implementing these basic object-oriented concepts in your software development practices will help you create more structured, maintainable, and reusable code, which is essential for building robust and scalable software systems.

Background:

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.
Terms used in Inheritance

- Class: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- Sub Class/Child Class: Subclass is a class that inherits the other class. It is also called a derived class, extended class, or child class.
- Super Class/Parent Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- Reusability: As the name specifies, reusability is a mechanism that facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

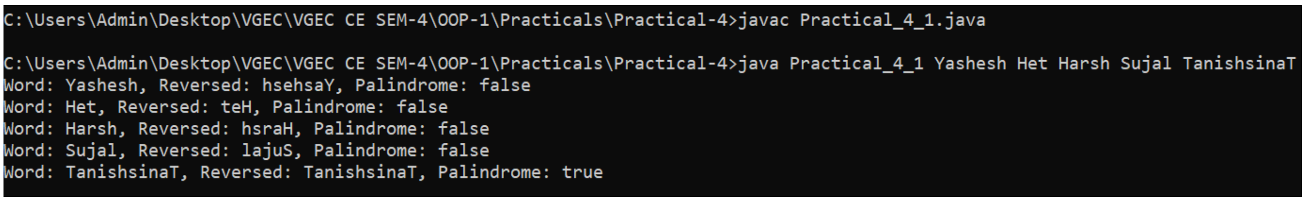
Practical questions:

1. **A set of 5 words (strings) will be taken as command line arguments. Write a program to reverse each word and check whether it is palindrome or not using method.**

Procedure:

```
class WordProcessor {
  String reverseWord(String word) {
    return new StringBuilder(word).reverse().toString();
  }

  boolean isPalindrome(String word) {
    return word.equalsIgnoreCase(reverseWord(word));
    }
}

class PalindromeChecker extends WordProcessor {
    void checkWords(String[] words) {
        for (String word : words) {
            String reversed = reverseWord(word);
            boolean palindrome = isPalindrome(word);
                System.out.println("Word: " + word + ", Reversed: " + reversed + ", Palindrome: " +
palindrome);
        }
    }
}

public class Practical_4_1 {
    public static void main(String[] args) {
        if (args.length != 5) {
            System.out.println("Please provide exactly 5 words as command-line arguments.");
            return;
        }

        PalindromeChecker checker = new PalindromeChecker();
        checker.checkWords(args);
    }
}
```

Observation:



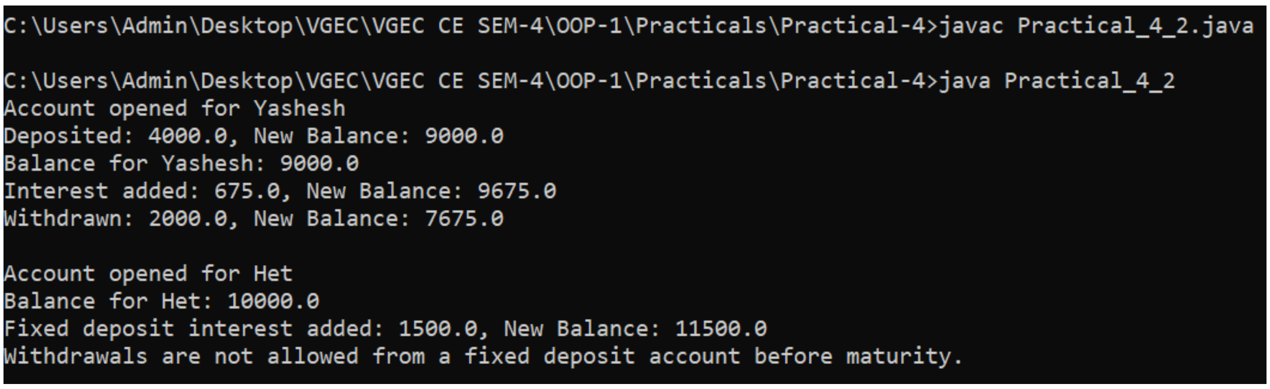
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>javac Practical_4_1.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>java Practical_4_1 Yashesh Het Harsh Sujal TanishsinaT
Word: Yashesh, Reversed: hsehsaY, Palindrome: false
Word: Het, Reversed: teH, Palindrome: false
Word: Harsh, Reversed: hsraH, Palindrome: false
Word: Sujal, Reversed: lajuS, Palindrome: false
Word: TanishsinaT, Reversed: TanishsinaT, Palindrome: true
```

2. **Define the class BankAccount to represent an account we open with bank. Define the subclasses SavingAccount and FixedDepositAccount. Implement the operations like openAccount(), deposit(), checkBalance(), withdraw() and calInterest() for these classes.**

Procedure:

```
class BankAccount {
    String accountHolder;
    double balance;

    BankAccount(String accountHolder, double initialBalance) {
        this.accountHolder = accountHolder;
        this.balance = initialBalance;
    }

    void openAccount() {
        System.out.println("Account opened for " + accountHolder);
    }

    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + ", New Balance: " + balance);
    }

    void checkBalance() {
        System.out.println("Balance for " + accountHolder + ": " + balance);
    }

    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ", New Balance: " + balance);
        } else {
            System.out.println("Insufficient balance");
        }
    }

    void calInterest() {
        System.out.println("Interest calculation not implemented for base class.");
    }
}

class SavingAccount extends BankAccount {
    double interestRate;

    SavingAccount(String accountHolder, double initialBalance, double interestRate) {
        super(accountHolder, initialBalance);
        this.interestRate = interestRate;
    }

    void calInterest() {
        double interest = balance * (interestRate / 100);
        balance += interest;
        System.out.println("Interest added: " + interest + ", New Balance: " + balance);
    }
}

class FixedDepositAccount extends BankAccount {
    double interestRate;
    int tenure;

    FixedDepositAccount(String accountHolder, double initialBalance, double interestRate, int
tenure) {
        super(accountHolder, initialBalance);
        this.interestRate = interestRate;
        this.tenure = tenure;
    }

    void callInterest() {
        double interest = balance * (interestRate / 100) * tenure;
        balance += interest;
        System.out.println("Fixed deposit interest added: " + interest + ", New Balance: " + balance);
    }

    void withdraw(double amount) {
          System.out.println("Withdrawals are not allowed from a fixed deposit account before
maturity.");
    }
}

public class Practical_4_2 {
    public static void main(String[] args) {
        SavingAccount savings = new SavingAccount("Yashesh", 5000, 7.5);
        savings.openAccount();
        savings.deposit(4000);
        savings.checkBalance();
        savings.callInterest();
        savings.withdraw(2000);

        System.out.println();

        FixedDepositAccount fixedDeposit = new FixedDepositAccount("Het", 10000, 5.0, 3);
        fixedDeposit.openAccount();
        fixedDeposit.checkBalance();
        fixedDeposit.callInterest();
        fixedDeposit.withdraw(1000);
    }
}
```

Observation:



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>javac Practical_4_2.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>java Practical_4_2
Account opened for Yashesh
Deposited: 4000.0, New Balance: 9000.0
Balance for Yashesh: 9000.0
Interest added: 675.0, New Balance: 9675.0
Withdrawn: 2000.0, New Balance: 7675.0

Account opened for Het
Balance for Het: 10000.0
Fixed deposit interest added: 1500.0, New Balance: 11500.0
Withdrawals are not allowed from a fixed deposit account before maturity.
```

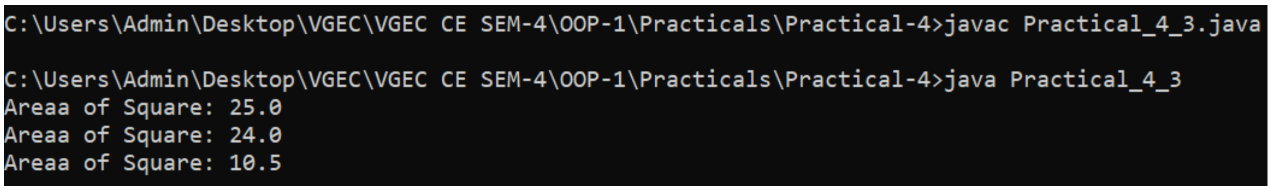
3. **Write a program that finds area of any shape by overloading area () method for Square, Rectangle, Triangle and Square.**

Procedure:

```
class Shape {
   void area() {
      System.out.println("Area calculation not implemented for base class.");
   }
}

class Square extends Shape {
   double side;

   Square(double side) {
      this.side = side;
   }

   void area() {
      System.out.println("Areaa of Square: " + (side * side));
   }
}

class Rectangle extends Shape {
   double length, width;

   Rectangle(double length, double width) {
      this.length = length;
      this.width = width;
   }

   void area() {
      System.out.println("Areaa of Square: " + (length * width));
   }
}

class Triangle extends Shape {
   double base, height;

   Triangle(double base, double height) {
      this.base = base;
      this.height = height;
   }

   void area() {
      System.out.println("Areaa of Square: " + (0.5 * base * height));
   }
}
```

```
public class Practical_4_3 {
    public static void main(String[] args) {
        Shape square = new Square(5);
        Shape rectangle = new Rectangle(4, 6);
        Shape triangle = new Triangle(3, 7);

        square.area();
        rectangle.area();
        triangle.area();
    }
}
```

Observation:



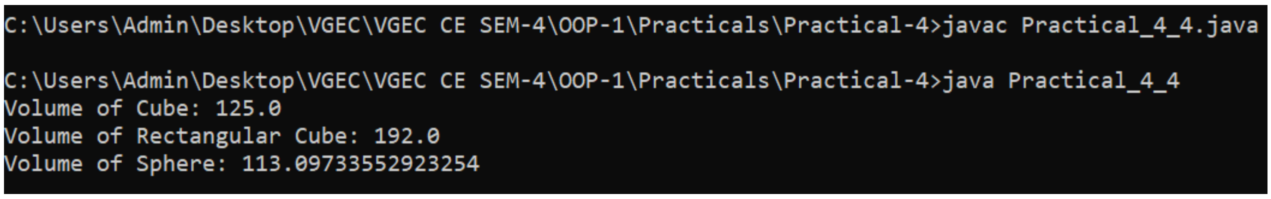
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>javac Practical_4_3.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>java Practical_4_3
Areaa of Square: 25.0
Areaa of Square: 24.0
Areaa of Square: 10.5
```

4. **Write a program that finds Volume of any shape by overloading volume () method for Cube, Rectangular Cube and Sphere.**

Procedure:

```
class Shape {
    void volume() {
        System.out.println("Volume calculation not implemented for base class.");
    }
}

class Cube extends Shape {
    double side;

    Cube(double side) {
        this.side = side;
    }

    void volume() {
        System.out.println("Volume of Cube: " + (side * side * side));
    }
}

class RectangularCube extends Shape {
    double length, width, height;

    RectangularCube(double length, double width, double height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }

    void volume() {
        System.out.println("Volume of Rectangular Cube: " + (length * width * height));
    }
}

class Sphere extends Shape {
    double radius;

    Sphere(double radius) {
        this.radius = radius;
    }

    void volume() {
        System.out.println("Volume of Sphere: " + ((4.0 / 3.0) * Math.PI * Math.pow(radius, 3)));
    }
}

public class Practical_4_4 {
    public static void main(String[] args) {
        Shape cube = new Cube(5);
        Shape rectangularCube = new RectangularCube(4, 6, 8);
        Shape sphere = new Sphere(3);

        cube.volume();
        rectangularCube.volume();
        sphere.volume();
    }
}
```

Observation:



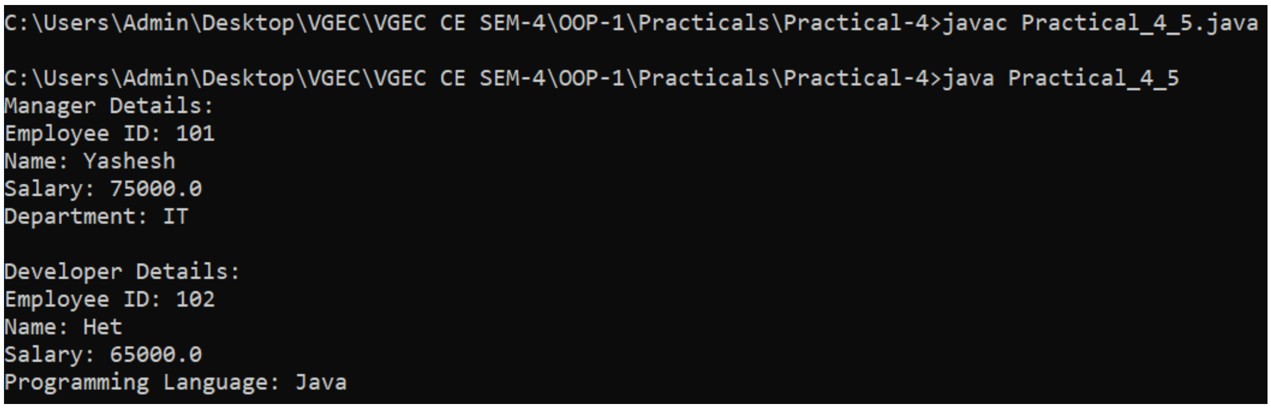
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>javac Practical_4_4.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>java Practical_4_4
Volume of Cube: 125.0
Volume of Rectangular Cube: 192.0
Volume of Sphere: 113.09733552923254
```

5. **Write a Program to maintain employee's information. Program should illustrate Inheritance concept. (Use your imagination to create class or subclass used for employee).**

Procedure:

```
class Employee {
    String name;
    int employeeId;
    double salary;
    Employee(String name, int employeeId, double salary) {
        this.name = name;
        this.employeeId = employeeId;
        this.salary = salary;
    }

    void displayInfo() {
        System.out.println("Employee ID: " + employeeId);
        System.out.println("Name: " + name);
        System.out.println("Salary: " + salary);
    }
}

class Manager extends Employee {
    String department;

    Manager(String name, int employeeId, double salary, String department) {
        super(name, employeeId, salary);
        this.department = department;
    }

    void displayInfo() {
        super.displayInfo();
        System.out.println("Department: " + department);
    }
}

class Developer extends Employee {
    String programmingLanguage;

    Developer(String name, int employeeId, double salary, String programmingLanguage) {
        super(name, employeeId, salary);
        this.programmingLanguage = programmingLanguage;
    }

    void displayInfo() {
        super.displayInfo();
        System.out.println("Programming Language: " + programmingLanguage);
    }
}

public class Practical_4_5 {
    public static void main(String[] args) {
        Manager manager = new Manager("Yashesh", 101, 75000, "IT");
        Developer developer = new Developer("Het", 102, 65000, "Java");

        System.out.println("Manager Details:");
        manager.displayInfo();
        System.out.println();

        System.out.println("Developer Details:");
        developer.displayInfo();
    }
}
```

Observation:



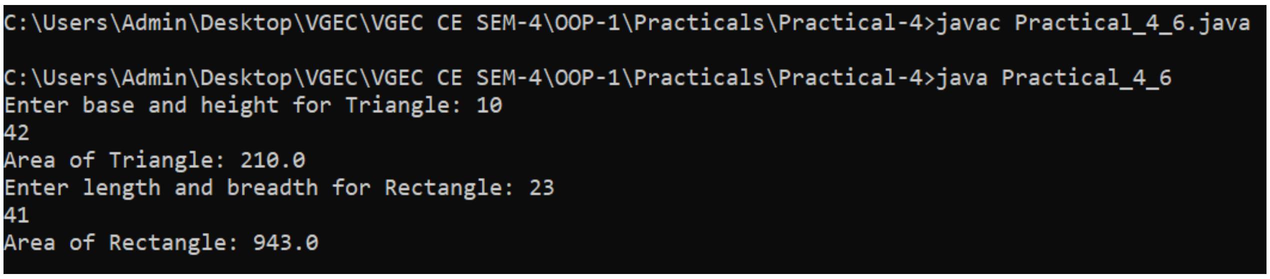
```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>javac Practical_4_5.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>java Practical_4_5
Manager Details:
Employee ID: 101
Name: Yashesh
Salary: 75000.0
Department: IT

Developer Details:
Employee ID: 102
Name: Het
Salary: 65000.0
Programming Language: Java
```

6. **Create a base class Shape. Use this class to store two double type values that could be used to compute area of any shape. Derive two specific classes called Triangle and Rectangle from the base shape. Add to the base a member function getdata() to initialize base class data member and another member function display_area() to compute and display the area of figures. (Use Method Overriding).**

Procedure:

```
import java.util.Scanner;
class Shape {
    double dim1, dim2;

    void getData(double d1, double d2) {
        this.dim1 = d1;
        this.dim2 = d2;
    }

    void displayArea() {
        System.out.println("Area calculation not implemented for base class.");
    }
}

class Triangle extends Shape {
    void displayArea() {
        double area = 0.5 * dim1 * dim2;
        System.out.println("Area of Triangle: " + area);
    }
}

class Rectangle extends Shape {
    void displayArea() {
        double area = dim1 * dim2;
        System.out.println("Area of Rectangle: " + area);
    }
}

public class Practical_4_6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter base and height for Triangle: ");
        double base = sc.nextDouble();
        double height = sc.nextDouble();
        Triangle triangle = new Triangle();
        triangle.getData(base, height);
        triangle.displayArea();

        System.out.print("Enter length and breadth for Rectangle: ");
        double length = sc.nextDouble();
        double breadth = sc.nextDouble();
        Rectangle rectangle = new Rectangle();
        rectangle.getData(length, breadth);
        rectangle.displayArea();

        sc.close();
    }
}
```

Observations:



```
C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>javac Practical_4_6.java

C:\Users\Admin\Desktop\VGEC\VGEC CE SEM-4\OOP-1\Practicals\Practical-4>java Practical_4_6
Enter base and height for Triangle: 10
42
Area of Triangle: 210.0
Enter length and breadth for Rectangle: 23
41
Area of Rectangle: 943.0
```

Conclusion:
In conclusion, mastering inheritance and object-oriented concepts allows for efficient code reuse, improved maintainability, and scalable software design. By leveraging class hierarchies and encapsulation, developers can create structured and modular code, leading to robust and adaptable software solutions.

Quiz: (Sufficient space to be provided for the answers)

1. What is inheritance in java? Explain different types of inheritance with proper example.
2. Explain the use of final and Super keyword in JAVA
3. Define polymorphism with its need.
4. Explain about Encapsulation, Abstraction.

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition(2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	
Marks											

Signature of Faculty:

Experiment No: 5

AIM: To demonstrate the use of abstract classes and interfaces.

Date:

CO mapped: CO-2

Objectives:

a) To understand the purpose and usage of abstract classes and interfaces in object-oriented programming. Develop the ability to design and implement abstract classes and interfaces effectively to promote code reusability, ensure consistent behavior in class hierarchies, and facilitate the development of flexible and extensible software systems.
b) Abstract classes and interfaces are important OOP concepts that allow you to define common contracts and behaviors for classes. Achieving this objective will enable you to use these tools to create more modular and maintainable software, especially when dealing with class hierarchies and multiple implementations.

Background:

A class that is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.
Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

An interface in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Practical questions:

1. Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object, i.e., area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.
2. Write a program that demonstrates the instance of operator. Declare interfaces I1 and I2. Interface I3 extends both of these interfaces. Also declare interface I4. Class X implements I3. Class W extends X and implements I4. Create an object of class W. Use the instance of operator to test if that object implements each of the interfaces and is of type X.
3. Write a java program to implement an interface called Exam with a method Pass (int mark) that returns a boolean. Write another interface called Classify with a method Division (int average) which returns a String. Write a class called Result which implements both Exam and Classify. The Pass method should return true if the mark is greater than or equal to 50 else false. The Division method must return "First" when the

parameter average is 60 or more, "Second" when average is 50 or more but below 60, "No division" when average is less than 50.

Procedure:

//Write program here

Observations:

//Write program output here

Conclusion: (Sufficient space to be provided)

Quiz: (Sufficient space to be provided for the answers)

1. Explain how interfaces promote the concept of multiple inheritance in OOP.
2. What is an interface, and how does it differ from an abstract class?
3. When would you choose to use an abstract class over an interface, and vice versa, in your software design?
4. Can a class implement multiple interfaces? If so, what benefits does this provide?
5. Can you declare an interface method static? Justify your answer.

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	**Knowledge (2)**		**Problem Recognition(2)**		**Logic Building (2)**		**Completeness and accuracy (2)**		**Ethics (2)**		**Total**
	Good (2)	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	
Marks											

Signature of Faculty:

Experiment No: 6
A.
AIM: To implement packages and exception handling in JAVA application.

Date:

CO mapped: CO-3

Objectives:

To effectively implement packages and exception handling in a Java application, organizing code into logical modules for improved maintainability, and ensuring robust error handling to enhance the application's reliability and user experience.

Background:

A java package is a group of similar types of classes, interfaces, and sub-packages. Package in java can be categorized in two forms, built-in package, and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc. Here, we will have the detailed learning of creating and using user-defined packages.
Exception Handling in Java is one of the powerful mechanisms to handle runtime errors so that the normal flow of the application can be maintained. In this practical, we will learn about Java exceptions, their types, and the difference between checked and unchecked exceptions.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
2) Java package provides access protection.
3) Java package removes naming collision.

Practical questions:

1. Write a program in Java to develop user defined exception for “Divide by Zero” error.
2. Write a program in Java to demonstrate throw, throws, finally, multiple try block and multiple catch exception.
3. Write a small application in Java to develop Banking Application in which user deposits the amount Rs 1000.00 and then start withdrawing ofRs 400.00, Rs 300.00 and it throws exception "Not Sufficient Fund" when user withdraws Rs 500 thereafter.
4. Write an application that contains a method named average () has one argument that is an array of strings. It converts these to double values and returns their average. The method generates a NullPointerException,if an array elements is null or a NumberFormatException, if an element is incorrectly formatted. Include throws statement in method declaration.
5. Write an application that generates custom exception if first argument from command line argument is 0.
6. A marklist containing reg.no and marks for a subject is given.if the marks are <0,user-defined IllegalMarkException is thrown out and handled with the message "Illegal Mark". For all valid marks, the candidate will be declared as "PASS" if the marks are equal to or greater than 40, otherwise it will be declared as "FAIL".Write a class called IllegalMarkException.

B.

AIM: To demonstrate the use of multithreading.

Date:

CO mapped: CO-3

Objectives:

a) To effectively demonstrate the use of multithreading in software applications, including creating and managing multiple threads, synchronizing their execution, and leveraging the power of concurrent programming to improve performance, responsiveness, and resource utilization.

b) Demonstrating the use of multithreading is crucial for building responsive and efficient software applications, and this objective emphasizes understanding the concepts and practical implementation of multithreading to achieve these goals.

Background:

Multithreading in Java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory areas so saves memory, and context-switching between the threads takes less time than the process. Java Multithreading is mostly used in games, animation, etc.

Practical questions:

1. Write a program to create a thread extending Thread class and demonstrate the use of sleep() method.
2. Write a program to create a thread implementing Runnable interface and demonstrate the use of join() method.
3. Write a program that launches 10 threads. Each thread adds 1 to a variable sum that initially is 0. Define an Integer wrapper object to hold sum. Run the program with and without synchronization to see its effect.

4. Write a program that demonstrate thread priority four threads each with a different priority level then the other are started objects and not the behave of each Thread

5. Write a program that demonstrate use of Executor Framework in mutitasking.

6. Write a program for handling producer consumer problem.

Procedure:

//Write program here

Observations:

//Write program output here

Conclusion: (Sufficient space to be provided)

Quiz: (Sufficient space to be provided for the answers)

1. Can you explain the difference between a process and a thread in the context of multithreading?
2. What are the different states in the lifecycle of a Java thread, and how does a thread transition between them?
3. What is runnable interface? How can you use this interface in creating thread?
4. Explain the concept of thread synchronization and the role of the synchronized keyword.
5. Explain: wait, sleep, notify and notify all.
6. How do you declare and define a package in Java?
7. Explain the benefits of organizing classes into packages in a Java application**.**
8. What is an exception in Java, and why is exception handling important in software development?
9. Explain the try-catch-finally block and its role in handling exceptions.
10. What is difference between throw and throws?

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	**Knowledge (2)**		**Problem Recognition(2)**		**Logic Building (2)**		**Completeness and accuracy (2)**		**Ethics (2)**		**Total**
	Good (2)	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	
Marks											

Signature of Faculty:

Experiment No: 7

AIM: To demonstrate I/O from files.

Date:

CO mapped: CO-4

Objectives:

To showcase proficiency in reading data from and writing data to files in various formats using programming languages, demonstrating the ability to implement reliable and efficient file I/O operations, which are essential for tasks such as data storage, retrieval, and processing in software applications.

Background:

Java I/O (Input and Output) is used to process the input and produce the output. Java uses the concept of a stream to make I/O operations fast. The java.io package contains all the classes required for input and output operations.

Practical questions:

1. Write a program that removes all the occurrences of a specified string from a text file. For example, invoking java Practical7_1 John filename removes the string John from the specified file. Your program should read the string as an input.
2. Write a program that will count the number of characters, words, and lines in a file. Words are separated by whitespace characters. The file name should be passed as a command-line argument.
3. Write a program to create a file named Practical7.txt if it does not exist. Write 100 integers created randomly into the file. Integers are separated by spaces in the file. Read the data back from the file and display the data in increasing order.

Observations: Put Output of the program

Conclusion: (Sufficient space to be provided)

Quiz: (Sufficient space to be provided for the answers)

1. What is file input/output (I/O), and why is it important in software development?
2. What are the common modes for opening files, and how do they differ (e.g., read, write, append)?
3. Describe the concept of file streams and how they are used in file I/O operations.
4. Write short notes about I/O stream classes.

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/

4. https://www.javatpoint.com/

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	**Knowledge (2)**		**Problem Recognition(2)**		**Logic Building (2)**		**Completeness and accuracy (2)**		**Ethics (2)**		**Total**
	Good (2)	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	
Marks											

Signature of Faculty:

Experiment No: 8
A.
AIM: To learn JAVA FX UI Controls.

Date:

CO mapped: CO-5

Objectives:

a) To gain proficiency in JavaFX UI controls, including understanding their features and capabilities, and developing the ability to create interactive and visually appealing user interfaces for Java applications. This knowledge will enable the design and development of user-friendly, responsive, and feature-rich graphical user interfaces (GUIs) in Java applications.
b) Learning JavaFX UI controls is essential for creating modern and engaging graphical user interfaces for Java applications. This objective emphasizes not only understanding the various UI controls but also the practical skills to design and implement user interfaces effectively.

Background:

Every user interface considers the following three main aspects –

UI elements − These are the core visual elements that the user eventually sees and interacts with. JavaFX provides a huge list of widely used and common elements varying from basic to complex, which we will cover in this practical.
Layouts − They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface). This part will be covered in the Layout chapter.
Behavior − These are events that occur when the user interacts with UI elements.
JavaFX provides several classes in the package javafx.scene.control. To create various GUI components (controls), JavaFX supports several controls such as date picker, button text field, etc. Each control is represented by a class; you can create a control by instantiating its respective class.

Common elements in a JavaFX application

All JavaFX applications contain the following elements:

1. A main window, called a stage in JavaFX.
2. At least one Scene in the stage.
3. A system of panes and boxes to organize GUI elements in the scene.
4. One or more GUI elements, such as buttons and labels.

The usual procedure for setting up a scene is to build it from the bottom up. First, we make the GUI elements, then we make boxes and panes to organize the elements, and finally, we put everything in the scene.

All JavaFX elements such as boxes and panes that are meant to contain other elements have a child list that we can access via the getChildren() method. We put elements inside other elements by adding things to child lists. In the code above you can see the button and the label objects being added as children of a VBox, and the VBox, in turn, is set as the child of a StackPane.

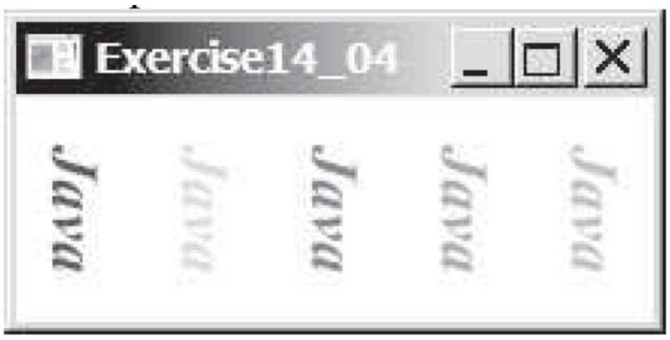
In addition to setting the structure for the window, we also call methods designed to set the properties of various elements. For example, the code in this example uses the button's setText() method to set the text the button will display.

Follow the procedure outlined in the section above to make a new JavaFX application. Replace the start() method in the App class with the following code:

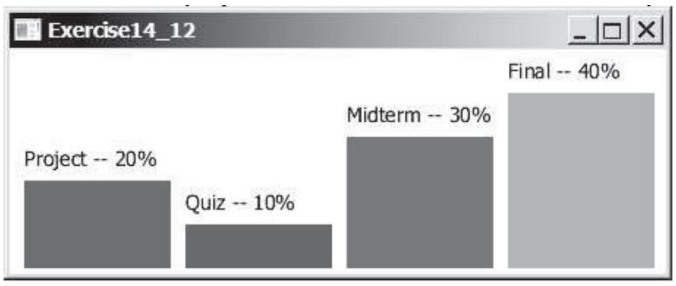
```
public void start(Stage primaryStage) {
    Button btn = new Button();
btn.setText("Say 'Hello World'");


StackPane root = new StackPane();
VBox box = new VBox();
box.getChildren().add(btn);
    Label label = new Label();
box.getChildren().add(label);
root.getChildren().add(box);


btn.setOnAction(new ClickHandler(label));


    Scene scene = new Scene(root, 300, 250);


primaryStage.setTitle("Hello World!");
primaryStage.setScene(scene);
primaryStage.show();
}
```

Practical questions:

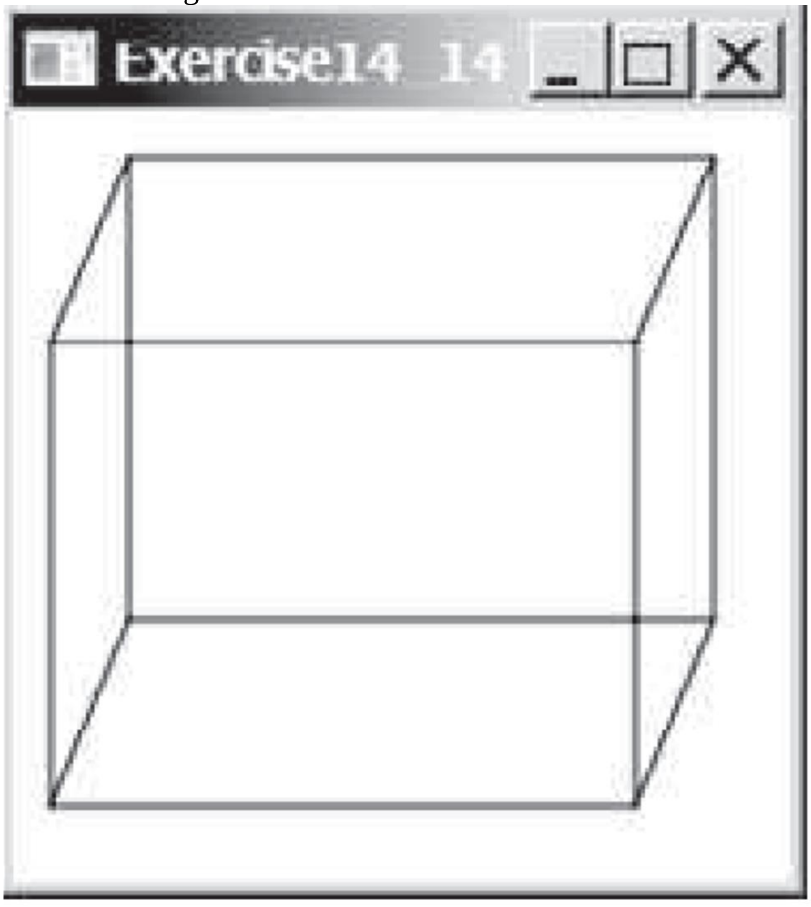
1. Write a program that displays five texts vertically, as shown in Figure. Set a random color and opacity for each text and set the font of each text to Times Roman, bold, italic, and 22 pixels.



2. Write a program that uses a bar chart to display the percentages of the overall grade represented by projects, quizzes, midterm exams, and the final exam, as shown in Figure b. Suppose that projects take 20 percent and are displayed in red, quizzes take 10 percent and are displayed in blue, midterm exams take 30 percent and are displayed in green, and the final exam takes 40 percent and is displayed in orange. Use the Rectangle class to display the bars. Interested readers may explore the JavaFXBarChart class for further study.



3. Write a program that displays a rectanguloid, as shown in Figure a. The cube should grow and shrink as the window grows or shrinks.



B.

AIM: To implement event handling and animation.

Date:

CO mapped: CO-5

Objectives:

To proficiently implement event handling and animation in software applications, fostering user interaction and engagement. Mastery of event-driven programming and animation techniques will empower the creation of dynamic, responsive, and visually captivating software experiences that cater to user needs and preferences.

Background:

Responding to user events

For a GUI application to be interactive, various elements such as buttons have to be able to respond to interactions from the user, such as clicks. In GUI applications user actions such as mouse clicks and key presses are called *events*. To set up an element such as a button to respond to user events, we arrange to connect special *event handling* code to the button.

Our first example demonstrates one way to do this in JavaFX. The first step is to connect an object to the button as the button's event handler via the button's setOnAction() method. The requirement here is that the object that we link to the button has to implement a particular interface, the EventHandler<ActionEvent> interface. That interface has one method in it, a handle() method that will get called when the user clicks on the button.

For the event handler code to do something useful, it will typically need to have access to one or more elements in the scene that will be affected by the button click. In this example, clicking the button will trigger a change in the text displayed in a label in the scene. To make this all work, the class we set up needs to have a member variable that is a reference to the label object. The code in handle() will use that reference to change the text shown in the label when the user clicks on the button.

Also, insert the code for the following class at the bottom of the App.java file:

```
classClickHandler implements EventHandler<ActionEvent> {

publicClickHandler(Label label) {

this.label = label;

   }


public void handle(ActionEventevt) {

label.setText("Hello, World!");

   }
private Label label;

}
```

A better way to handle events

Although the process for linking an event handler to a button is fairly straightforward, it is a little clunky. This process can get even more tedious when we start building applications with many buttons that need event handlers. As a fix for this, JavaFX allows us to use a simpler mechanism to set up event handlers.

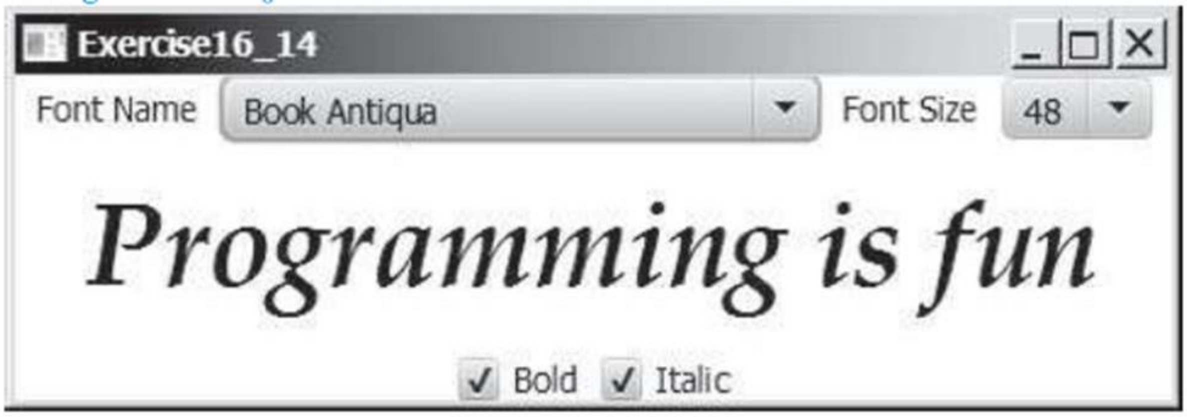
To see how this mechanism works, remove the ClickHandler class completely and replace the line of code in start() that calls the button's setOnAction() method with this:

```
btn.setOnAction((e)->{label.setText("Hello, World!");});
```

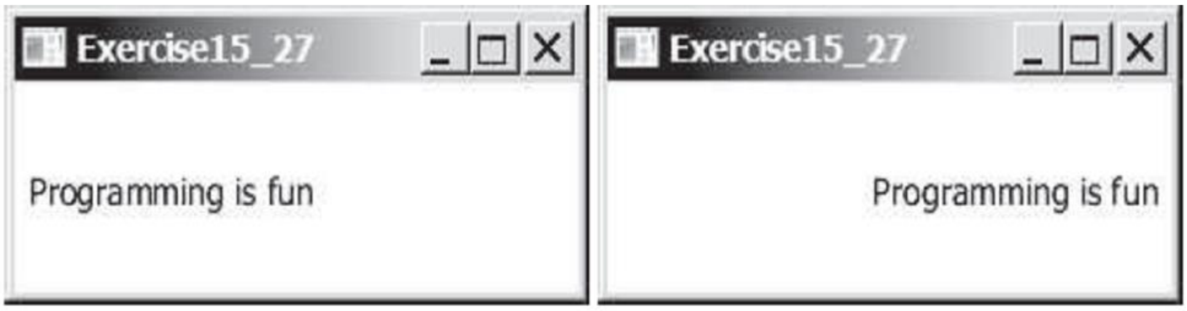
The ClickHandler class that you just eliminated had a handle() method in it. That method took a single parameter, which was an ActionEvent object, e. The code we just put in place of the original code contains a *lambda expression* mapping that parameter e to a chunk of code that will run when the event takes place. This code is the code that used to live in the body of the handle() method. The new statement saves a lot of space over the original. The lambda expression replaces the ClickHandler class and its handle() method with a simpler alternative.

Practical questions:

1. Write a program that can dynamically change the font of a text in a label displayed on a stack pane. The text can be displayed in bold and italic at the same time. You can select the font name or font size from combo boxes, as shown in Figure. The available font names can be obtained using Font.getFamilies(). The combo box for the font size is initialized with numbers from 1 to 100.



2. Write a program that displays a moving text, as shown in Figure. The text moves from left to right circularly. When it disappears in the right, it reappears from the left. The text freezes when the mouse is pressed and moves again when the button is released.

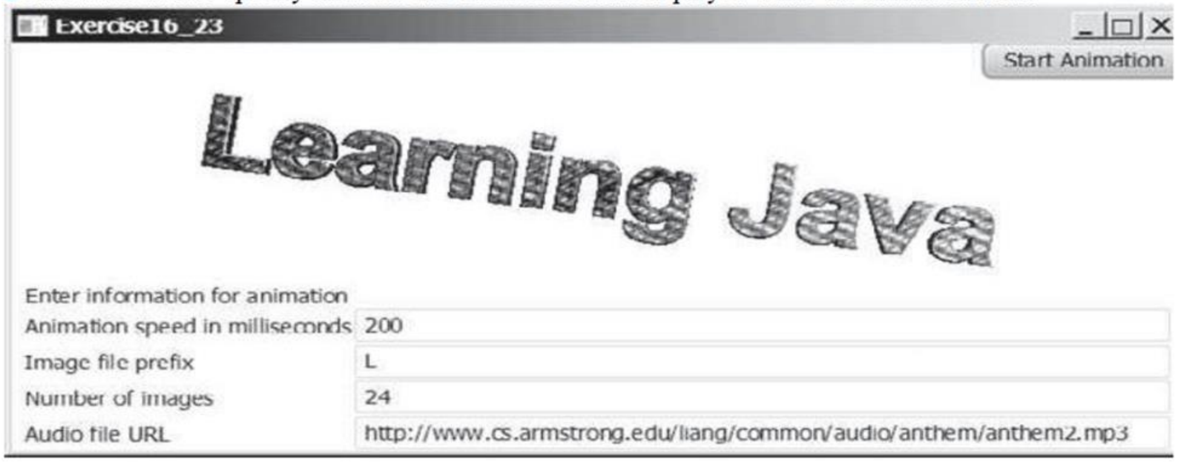


3. Create animation in Figure to meet the following requirements:

■ Allow the user to specify the animation speed in a text field.

■ Get the number of iamges and image's file-name prefix from the user. For example, if the user enters n for the number of images and L for the image prefix, then the files are L1.gif, L2.gif, and so on, to Ln.gif. Assume that the images are stored in the image directory, a subdirectory of the program's class directory. The animation displays the images one after the other.

■ Allow the user to specify an audio file URL. The audio is played while the animation runs.



Enter information for animation
Animation speed in milliseconds 200
Image file prefix L
Number of images 24
Audio file URL http://www.cs.armstrong.edu/liang/common/audio/anthem/anthem2.mp3

Procedure:

//Write program here

Observations:

//Write program output here

Conclusion: (Sufficient space to be provided)

Quiz: (Sufficient space to be provided for the answers)

1. How does event handling work in Java, and what is the event-driven programming model?
2. What is the role of the java.awt.event and javafx.event packages in Java event handling?
3. Explain: MouseEvent, KeyEvent, ActionEvent
4. What are the primary libraries or frameworks for creating animations in Java, and which

one do you prefer?

5. How to set the cycle count of an animation to infinite?
6. Explain the evolution of Java GUI technologies since awt,swing and JavaFX.
7. What is the purpose of a TextField control, and how can it be used to collect user input?
8. How to create an ImageView from an Image, or directly from a file or a URL?.
9. What are the primary layout controls in JavaFX, and how do they impact the arrangement of UI components?
10. What is CSS, and how is it used for styling JavaFX UI controls?

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	**Knowledge (2)**		**Problem Recognition(2)**		**Logic Building (2)**		**Completeness and accuracy (2)**		**Ethics (2)**		**Total**
	Good (2)	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	**Good (2)**	**Avg. (1)**	
Marks											

Signature of Faculty:

Experiment No: 9

AIM: To learn recursion and generics.

Date:

CO mapped: CO-4

Objectives:

a) To develop a deep understanding of recursion and generics in programming. Mastery of recursion will enable the development of elegant and efficient algorithms for solving complex problems. Understanding generics will facilitate the creation of flexible, reusable, and type-safe code in various programming languages.

b) Learning recursion and generics is crucial for building efficient algorithms and writing more versatile and type-safe code in software development. Achieving this objective will help you become a more proficient and well-rounded programmer.

Background:

Recursion in java is a process in which a method calls itself continuously. A method in java that calls itself is called the recursive method.
Java Generics programming is introduced in J2SE 5 to deal with type-safe objects. It makes the code stable by detecting the bugs at compile time. Before generics, we can store any type of object in the collection, i.e., non-generic. Now generics force the java programmer to store a specific type of object.

Practical questions:

1. Write a recursive method that converts a decimal number into a binary number as a string. The method header is: public static String dec2Bin(int value)

Write a test program that prompts the user to enter a decimal number and displays its binary equivalent.

2. Write the following method that returns a new ArrayList. The new list contains the non-duplicate elements from the original list.

public static <E>ArrayList<E>removeDuplicates(ArrayList<E> list)

3. Implement the following method using binary search.

public static <E extends Comparable<E>>

intbinarySearch(E list, E key)

Procedure:

//Write program here

Observations:

//Write program output here

Conclusion: (Sufficient space to be provided)

Quiz: (Sufficient space to be provided for the answers)

1. What is recursion in Java, and how does it differ from iteration in solving problems?
2. What are the advantages and disadvantages of using recursion in Java?
3. What are generics in Java, and why are they used for creating parameterized types?
4. How to define Generic class? What are restrictions of generic programming?
5. Can you provide an example of a generic class in Java, such as a generic ArrayList?

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition(2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	
Marks											

Signature of Faculty:

Experiment No: 10

AIM: To demonstrate the use of Collection framework.

Date:

CO mapped: CO-4

Objectives:

a) To proficiently demonstrate the use of Java's Collection framework, including understanding its core interfaces (List, Set, Map), implementing and manipulating data structures like lists, sets, and maps, and effectively applying collections for data storage, retrieval, and manipulation in Java applications.

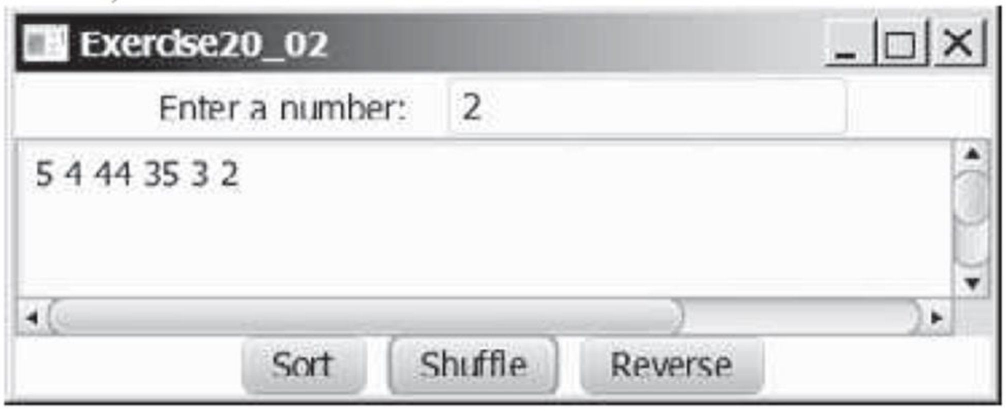
b) Mastery of the Java Collection framework is essential for managing and organizing data efficiently in Java applications. This objective focuses on understanding the core collection interfaces and using them to build versatile data structures to meet various application needs.

Background:

The Collection in Java is a framework that provides architecture to store and manipulate a group of objects. Java Collections can achieve all the operations that you perform on data such as searching, sorting, insertion, manipulation, and deletion. Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

Practical questions:

1. Write a program that lets the user enter numbers from a graphical user interface and displays them in a text area, as shown in Figure. Use a linked list to store the numbers. Do not store duplicate numbers. Add the buttons Sort, Shuffle, and Reverse to sort, shuffle, and reverse the list.



```
2
5 4 44 35 3 2
```

2. Create two priority queues, {"George", "Jim", "John", "Blake", "Kevin", "Michael"} and {"George", "Katie", "Kevin", "Michelle", "Ryan"}, and find their union, difference, and

intersection.

3. Store pairs of 10 states and its capital in a map. Your program should prompt the user to enter a state and should display the capital for the state.

Procedure:

//Write program here

Observations:

//Write program output here

Conclusion: (Sufficient space to be provided)

Quiz: (Sufficient space to be provided for the answers)

1. Write a note on 'Collection in JAVA'. Also discuss List and Enumeration Interface.
2. Differentiate between Enumeration and Iterator.
3. Compare List, Set and Map interfaces. Also compare ArrayList, TreeSet and HashMap classes in java.
4. Explain the unique features of Map interface.
5. How do you perform common operations like sorting, searching, or filtering on Collections?

Suggested Reference:

1. https://www.tutorialspoint.com/java/
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/java/
4. https://www.javatpoint.com/

References used by the students: (Sufficient space to be provided)

Rubric wise marks obtained:

Rubrics	Knowledge (2)		Problem Recognition(2)		Logic Building (2)		Completeness and accuracy (2)		Ethics (2)		Total
	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	Good (2)	Avg. (1)	
Marks											

Signature of Faculty: