Alliance School of Advanced Computing

Department of Computer Science and Engineering

Course Name: Java Full Stack Development

Laboratory Manual

Prepared by:

Dhanush S

Course Coordinator

Approved By:

Dr. Ezil Sam Leni

HoD

VISION OF THE UNIVERSITY

Alliance University's vision is to be a world-class University that nurtures talent and catalytically transforms the lives of millions through excellence in teaching, research, service and community development. To uphold a commitment to shaping lives through scholarly teaching and learning, and that which contributes to an equitable and holistic transformation of society at large.

MISSION OF THE UNIVERSITY

The mission of the University is to create and sustain a community of lifelong learners in an environment that emphasizes literacy, critical thinking, and humanistic and scientific inquiry. The University provides a dynamic, challenging and ethical environment for pursuing high quality teaching, research, learning and service across all areas of university, where students, faculty and other key constituents can interact, collaborate and partner with the global community for creation and dissemination of knowledge and transform lives of people through innovation and excellence in higher education.

University Strives to

- Pursue excellence in teaching, learning and scholarship.
- Prepare students for leadership through enlightened learning partnerships with faculty mentors and the community.
- Support faculty and other scholars in pursuing world-class research: clinical, theoretical, empirical, experiential and foster creative endeavour.
- Promote and preserve academic freedom, diversity, equality, harmony and justice.
- Develop mastery of disciplines and professions and instil confidence among its key constituents in their application for a future of meaningful pursuits and productive work in the service of humanity.
- Channelise faculty and student talent for professionally related service to the university, the community and society at large.

Core Values

- Uncompromising Academic Integrity.
- Scholarly work and continuous self-improvement.
- Encouraging and building student ability, character and creativity.
- Pride in Self, University and the Community.
- Leadership, Service, Philanthropy, Social Justice, and Entrepreneurship.

Alliance School of Advanced Computing

VISION:

To be a technical institution of repute nationally and internationally in higher learning and research and to produce creative solutions to societal needs.

MISSION:

1. To provide every individual with a conducive environment suitable to achieve career goals, with a strong emphasis on personality development, and to offer resources to the academically inclined,
2. To gain quality education in all spheres of engineering, applied sciences and management, without compromising the quality and code of ethics to each student of the Institution.

Department of Computer Science and Engineering

VISION:

Envisions to develop world class Engineers, researchers, and entrepreneurs in the field of Computer Science & Information Technology driven by scientific curriculum research, and experiential learning that will enable students to solve real world problems.

MISSION:

3. To impart critical thinking, humanistic and scientific inquiry in disciplinary and interdisciplinary competencies and skills to solve-real world challenges.
4. To provide an environment of continuous learning and improvement for students, faculty members, and other stakeholders by instilling a culture of entrepreneurial spirit through academic and research collaborations within university and across national and international institutions of repute.
5. To create highly skilled graduates with potential, to design and develop software systems using innovative approaches in programming and problem solving

Programme Details:

Program Educational Objectives:

PEO1: Apply critical thinking to design, develop, analyze and implement solutions to complex problems in computer science and engineering.

PEO2: Effectively deliver technical ideas and collaborate with multidisciplinary teams to address societal needs and contribute to the advancement of technology.

PEO3: Engage in innovative practices, entrepreneurial endeavors, or technology-based startups, contributing to the development of new products, services, or solutions that address societal needs and challenges

Program Outcomes:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2: Problem analysis: Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice.
PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with the 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.
PO11: Project management and finance: Demonstrate knowledge understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes:

PSO1: Apply knowledge of mathematics, science, and engineering to solve complex engineering problems.

PSO2: Implement secured network system using standard protocols and security principles.

PSO3: Design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.

Student Outcomes (EAC) [FOR ENGINEERING PROGRAMS ONLY]

SO1: an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics.

SO2: an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors.

SO3: an ability to communicate effectively with a range of audiences.

SO4: an ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts.

SO5: an ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives

SO6: an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions

SO7: an ability to acquire and apply new knowledge as needed, using appropriate learning strategies.

Objectives of the Lab

1. To develop a strong foundation in Java programming concepts, including object-oriented principles, inheritance, polymorphism, interfaces, and exception handling.
2. To gain hands-on experience in designing and building dynamic and interactive web pages using HTML, CSS, and JavaScript.
3. To understand and implement front-end development concepts using React, including components, lifecycle methods, and state management.
4. To acquire skills in back-end development using Node.js and Express, including creating RESTful APIs, handling databases, and managing server-client communication.
5. To master database operations and NoSQL concepts using MongoDB for storing, retrieving, and managing data efficiently in web applications.

Overview of the Lab Activities

1. **Java Programming**
    - Implementing command-line arguments and inheritance.
    - Exploring object-oriented programming concepts like polymorphism, interfaces, abstract classes, and exception handling.
2. **Web Development Basics**
    - Designing simple HTML pages with personal and product-related content.
    - Styling web pages using CSS selectors (class, ID, and attribute-based).
    - Creating interactive forms with JavaScript for real-world scenarios like voting eligibility.
3. **React Development**
    - Building dynamic and responsive user interfaces using React components.
    - Implementing functionality such as profile cards and login registration forms.
    - Managing data flow and state with Redux and client-server communication.
4. **Server-Side Programming with Node.js and Express**
    - Developing a simple chat application using Node.js event emitters.
    - Creating and testing RESTful APIs with HTTP methods like GET, POST, PUT, and DELETE.
    - Building CRUD applications integrated with databases for robust server-side functionalities.
5. **Database Management with MongoDB**
    - Creating and managing NoSQL database tables for e-commerce applications.
    - Connecting MongoDB to web applications for dynamic user registration and validation.
    - Developing full-stack applications that integrate CRUD operations with a MongoDB back end.

GENERAL LABORATORY INSTRUCTIONS

Students are advised to strictly follow the instructions given below while working in the lab to complete the experiments well in time and understand them clearly.

Instructions for Safety and Discipline

1. Handle the computer and peripherals with care to avoid physical damage
2. Install and update software only with permission from lab administrators or IT staff.
3. Avoid downloading or installing unauthorized software to prevent security risks and system instability.
4. Use secure login procedures and passwords to restrict access to authorized users only.
5. Log off or shut down properly when leaving them unattended to prevent unauthorized access.
6. Avoid eating or drinking near computer equipment to prevent spills and damage.
7. Avoid Mobile/Laptop/Earphones using or charging in the lab premises.

Attendance Instructions:

1. Punctuality: Arrive on time for all scheduled lab sessions. Late arrivals may not be allowed to participate in the lab activities for safety reasons. Sign in upon arrival and sign out when leaving the lab.
2. Attendance Requirements: Attend all scheduled lab sessions. If you cannot attend, notify the lab supervisor in advance and provide a valid reason. Make up for any missed lab sessions according to the lab supervisor's instructions.
3. Participation: Actively participate in all lab activities and discussions. Complete all assigned tasks and experiments within the lab session time frame

Syllabus

<table>
<tr><td>Course Title</td><td colspan="3">JAVA FULL STACK DEVELOPMENT</td><td colspan="7">CREDITS</td><td>3</td></tr>
<tr><td rowspan="2">Course Code</td><td rowspan="2">5CS1012</td><td rowspan="2">Course Category</td><td rowspan="2">Core</td><td>L</td><td>T</td><td>P</td><td>S</td><td>C</td><td>CH</td><td rowspan="2">1-0-2-4-3</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>4</td><td>3</td><td>7</td></tr>
<tr><td>Version</td><td></td><td>Approval Details</td><td>ACM</td><td colspan="7">LEARNING LEVEL</td></tr>
<tr><td>Co-requisite</td><td colspan="2">Network Programming</td><td>Pre-requisite</td><td colspan="7">Web Programming</td></tr>
<tr><td>Course Coordinator</td><td colspan="10">Dhanush S</td></tr>
<tr><td>Course Description</td><td colspan="10">Java Full-Stack Development is the subject where students are exposed to the current Web Technologies trends and methods followed in the corporate world. With fast-evolving Technology, a student gains expertise in the field by understanding the entire process of development from design to actual deployment. This has introduced a new role for developers like Java Full Stack Developers. This course compiles all technologies in perfect synchronization to help developers transition from simple programmers to full-stack developers. This course comprises five segments including the Front end, Backend, Database, other essential Technologies, and debugging/version control.</td></tr>
<tr><td>Course Objectives</td><td colspan="10">1. Understand the core concepts of Java programming, including data types, control statements, object-oriented principles, and the Java Collection Framework.
2. Gain proficiency in HTML, CSS, and JavaScript to develop and design dynamic & interactive web pages.
3. Learn the fundamentals of React, JSX syntax, how to construct applications, manage components, style interfaces, maximize performance, use Hooks to implement stateful logic, connect Firebase for CRUD and real-time data, and query React effectively.
4. Master the use of Node.JS for creating robust web applications, including concepts of RESTful APIs and Node Express.
5. Understand NoSQL database concepts and learn how to build and manage MongoDB databases.</td></tr>
<tr><td>Course Outcome</td><td colspan="8">CO Statement (MAX 5 OUTCOMES)</td><td colspan="2">Bloom’s Taxonomy Level</td></tr>
<tr><td>1</td><td colspan="8">Students will apply fundamental programming concepts and techniques to effectively tackle various computational problems</td><td colspan="2">BTL3</td></tr>
<tr><td>2</td><td colspan="8">Students will evaluate and create web pages and interactive web applications using contemporary JavaScript approaches, demonstrating proficiency in web development.</td><td colspan="2">BTL5</td></tr>
</table>

3	Students will analyze and utilize modern JavaScript techniques to develop dynamic React applications, allowing them to create, style, and manage interactive web applications.	BTL4
4	Students will design and create reliable online applications by implementing RESTful API development, MVC concepts, Node.js, and Express	BTL5
5	Students will apply NoSQL principles to manage MongoDB databases, set up environments, ensure user access, and integrate MongoDB into applications for effective querying and data management.	BTL3

Course Syllabus

Module	**Introduction to Programming**	**Contact Hours:12**
Module -I	Basic of Programming-Loops and Functions-Arrays-Strings and 2D Arrays. Problem Solving Techniques-Object Oriented Programming. Java Coding Standards-Package –Interface-Abstract Class-Exception Handling.	
Module -II	**Simple Web Development**	**Contact Hours:12**
	Getting started with Basics-Introduction to HTML-Forms. Introduction to CSS-Styling with CSS-CSS Selectors-**JavaScript-**Fundamentals of JavaScript-Loops-Arrays-Function & Objects-Working of JS-Function of JS-OOP in JS-ES6 in JS,	
Module -III	**React JS**	**Contact Hours:12**
	Introduction to React - React Render HTML - React Components - Styling in React-Component Lifecycle Methods - and Single Page Applications React Forms - Flow Architecture and Introduction to Redux More Redux and Client-Server Communication	
Module -IV	**NODE.JS and EXPRESS**	**Contact Hours:12**
	Node.js basics - Local and Export Modules - Node Package Manager - Node.js web server - Node.js File system - Node Inspector - Node.js Event Emitter - Frameworks for Node.js - Express.js Web App - Serving static Resource - Node.js Data Access - Express REST APIs - REST - Resource Based - HTTP Methods as Actions - JSON- Express - Routing - Handler Function - Middleware - The List API - Automatic Server Restart - Testing - The Create API - Using the List API - Using the Create API- Error Handling - Template Engine.	
Module -V	**MongoDB**	**Contact Hours:12**
	Query API, Database Creation, Understanding NoSQL and MongoDB – Building MongoDB Environment – User accounts – Access control – Administering databases – Managing collections – Connecting to MongoDB – simple applications	

Practical Syllabus		
Experiment No.	**Name**	**Tool/ Library Used**
1.	1. To implement program with Command-line arguments, Inheritance 2. To implement program with Polymorphism, interface, abstract classes and exception handling using java	**Java compiler (ubuntu)**
2.	1. To create a simple HTML page that includes your name, a photo, a short biography, and links to your favourite websites. 2. To create an HTML page with a list of products. Use different CSS selectors (class, ID, attribute) to style each product differently. 3. To create voting eligibility form using HTML, CSS and JavaScript (JS).	**Visual studio code or Windows 10 (Eclipse)**
3.	1. To create a dynamic greeting message based on a user's input using React Render, HTML. 2. To build your Profile Card and display that profile information using React components. 3. To create login registration form and handle form submission by using React	**Visual studio code or Windows 10 (Eclipse)**
4.	1. To create a simple chat application where messages are sent and received using Node.js events Emitter. 2. To create a RESTful API with different HTTP methods (GET, POST, PUT, DELETE) 3. Write a program to display CRUD application using HTML with Databases	**Visual studio code or Windows 10 (Eclipse) and MongoDB**
5.	1. To create database table for simple e-commerce application such as Products, Users, and Orders etc. 2. To create a simple user registration form using HTML and JavaScript. Use MongoDB to store user data. Implement basic validation for the registration form. 3. To build a simple web application using Node.js and Express that connects to a MongoDB database. Implement CRUD operations for managing user profiles.	**Visual studio code or Windows 10 (Eclipse) and MongoDB**
	A Mini Project: **Problem Statement:** Develop a web-based task management system where users can create, manage, and collaborate on tasks in real-time. The application will include features like user authentication, task creation and assignment, real-time updates, and data persistence using MongoDB. It will showcase frontend development with ReactJS, backend	**User Preferences**

	development with Spring Boot, and database management with MongoDB	

Self-Study topics for Advanced Learners
1. Explore advanced state management techniques in ReactJS using Redux or Context API to manage complex state across components in the task management system. 2. Study advanced schema design techniques in MongoDB to optimize data storage and retrieval for tasks, users, and notifications in the task management system 3. Deepen your understanding of functional programming principles in JavaScript, applying concepts like higher-order functions and immutability to enhance code quality and maintainability.

Textbooks	
1.	Herbert Schildt, Java the Complete Reference, Ninth Edition, Oracle Press, McGraw Hill Education-2014. **[Module-I]**
2	Mardan, Azat, Azat Mardan, and Corrigan. Full Stack JavaScript. Apress, 2018**[Module-II, III, V]**
3	Craig Walls, Spring in Action, Sixth Edition, Manning shelter Island. **[Module-IV]**
4	Brad Dayley, Brendan Dayley, Caleb Dayley, 'Node.js, MongoDB and Angular Web Development', Addison-Wesley, Second Edition, 2018 **[Module-II, III, V]**
Reference books	
1.	Northwood, Chris. The full stack developer: your essential guide to the everyday skills expected of a modern full stack web developer. Apress, 2018.
2.	Dean, John. Web programming with HTML5, CSS, and JavaScript. Jones and Bartlett Learning, 2018.
3.	Brown, Ethan. Web development with node and express: leveraging the JavaScript stack. O'Reilly Media, 2019
4.	MongoDB: The Definitive Guide by Shannon Bradshaw, Eoin Brazil, Kristina Chodorow.
5.	Learning React: A Hands-On Guide to Building Web Applications Using React and Redux by Kirupa Chinnathambi
E Books	
1.	https://www.pourzad.com/Programming/Learn%20Java%20for%20Web%20Development.pdf
2	https://mu.ac.in/wp-content/uploads/2022/09/Core-JAVA.pdf

Coursera Credit Course:	
1. Coursera/NPTEL/SWAYAM courses linked with credit transfer:	
1.	https://www.coursera.org/learn/capstone-react-app/ **(13-hrs)**
2.	https://www.coursera.org/learn/introduction-to-web-development-with-html-css-javacript **(13-hrs)**
3.	https://www.coursera.org/learn/developing-frontend-apps-with-react?specialization=ibm-full-stack-cloud-developer (**14-hrs)**
4	https://infyspringboard.onwingspan.com/web/en/login (**36-hrs)**
2. MOOC Courses for optional learning (Advanced learners):	
1.	https://www.coursera.org/learn/introduction-to-cloud?specialization=ibm-full-stack-cloud-developer
2.	Meta Front-End Developer Professional Certificate | Coursera
Online Resources	
1.	https://www.tutorialspoint.com/the_full_stack_web_development/index.asp
2.	https://nareshit.com/courses/full-stack-java-online-training
3.	https://intellipaat.com/full-stack-web-developer-mean-stack-certification-training/?utm_source=google&utm_medium=display&utm_campaign=p_performancemax_generic_in_dec_2023&gad_source=1&gclid=CjwKCAjw7NmzBhBLEiwAxrHQ-ftR2D__tf9z9Fr30ox_DoOM7-Giihp1fWMnZMkQP_Y-QiShQR69hBoCDc8QAvD_BwE
Case Studies	
1.	Develop a real-time collaborative document editing platform similar to Google Docs, leveraging Java for back-end development and modern front-end frameworks for a seamless user experience
2.	Develop a health monitoring system using Java Full Stack technologies, integrating with IoT devices to collect and analyze health data for proactive healthcare management
3.	Build a content recommendation and streaming platform similar to Netflix, utilizing Java Full Stack technologies to enhance user engagement and provide a personalized viewing experience

CO-PO-PSO Mapping

Course Outcomes	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	2	2	1	-	-	-	1	-	2	1	1	2	2	-	1
CO2	2	2	1	-	1	1	1	-	1	1	1	2	2	1	1
CO3	2	2	1	-	2	1	1	-	1	1	1	2	2	2	-
CO4	3	3	2	-	3	2	1	-	2	2	1	2	1	-	-
CO5	3	3	3	1	3	2	2	-	1	2	2	2	2	2	-

3-Strong Contribution (SC) **2-Moderate Contribution(MC)** **1-Weak Contribution(WC)**

CO-SO MAPPING:

Course Outcomes	Student Outcomes (EAC)							Student Outcomes (CAC)				
	SO1	SO2	SO3	SO4	SO5	SO6	SO7	SO1	SO2	SO3	SO4	SO5
CO1	1	-	-	-	-	-	1	-	1	-	-	-
CO2	1	1	1	-	-	2	2	2	1	-	-	-
CO3	1	1	1	-	-	2	1	1	2	-	-	-
CO4	1	-	-	-	-	-	-	-	-	-	-	-
CO5	1	1	1	-	-	-	-	2	2	-	-	-

3-Strong Contribution (SC) **2-Moderate Contribution(MC)** **1-Weak Contribution(WC)**

List of Experiments

Sl. No.	Name of Experiment	Tools Required for Conduction	Page No.
1	To implement a program with Command-line arguments and Inheritance	Java Compiler (Ubuntu), Eclipse (Windows)	
2	To implement a program with Polymorphism, Interface, Abstract Classes, and Exception Handling	Java Compiler (Ubuntu), Eclipse (Windows)	
3	To create a simple HTML page that includes your name, a photo, a short biography, and favourite links	Visual Studio Code, Windows 10	
4	To create an HTML page with a list of products styled using different CSS selectors	Visual Studio Code, Windows 10	
5	To create a voting eligibility form using HTML, CSS, and JavaScript	Visual Studio Code, Windows 10	
6	To create a dynamic greeting message based on a user's input using React	Visual Studio Code, Windows 10	
7	To build your Profile Card and display it using React components	Visual Studio Code, Windows 10	
8	To create a login registration form and handle form submission using React	Visual Studio Code, Windows 10	
9	To create a simple chat application where messages are sent and received using Node.js Event Emitters	Visual Studio Code, Windows 10	
10	To create a RESTful API with different HTTP methods (GET, POST, PUT, DELETE)	Visual Studio Code, Windows 10	
11	To display CRUD operations using HTML with a database	Visual Studio Code, MongoDB, Windows 10	
12	To create a database table for an e-commerce application (Products, Users, Orders, etc.)	Visual Studio Code, MongoDB, Windows 10	
13	To create a simple user registration form using HTML and JavaScript and store data in MongoDB	Visual Studio Code, MongoDB, Windows 10	
14	To build a web application using Node.js and Express that connects to MongoDB and implements CRUD	Visual Studio Code, MongoDB, Windows 10	

Exp. No.: 1a **To Implement a Program with Command-line Arguments and Inheritance** **Date:**

Aim:

To write a Java program that demonstrates the use of command-line arguments and inheritance.

Requirements:

- Java Compiler (Ubuntu)
- Eclipse (Windows)

Suggested Reading (Theoretical Background):

Refer to **Module I**, Chapter **2.1 and 2.3 (T1)** for understanding the concepts of command-line arguments and inheritance in Java.

Algorithm:

Part A: Using Command-line Arguments

1. Start.
2. Accept input values as command-line arguments.
3. Convert the input values to appropriate data types (e.g., integers or doubles).
4. Perform the necessary operation (e.g., sum of numbers).
5. Display the result.
6. End.

Part B: Demonstrating Inheritance

1. Start.
2. Create a base class with attributes and methods.
3. Create a derived class that inherits from the base class.
4. Add specific methods or attributes to the derived class.
5. Create objects of both classes and demonstrate the inheritance hierarchy.
6. End.

Sample Code:

```
public class CommandLineExample {
   public static void main(String[] args) {
```

```
        if (args.length < 2) {
            System.out.println("Please provide at least two numbers as arguments.");
            return;
        }
        int num1 = Integer.parseInt(args[0]);
        int num2 = Integer.parseInt(args[1]);
        System.out.println("Sum of " + num1 + " and " + num2 + " is: " + (num1 + num2));
    }
}

// Part B: Inheritance
class Base {
    void display() {
        System.out.println("This is the base class.");
    }
}

class Derived extends Base {
    void show() {
        System.out.println("This is the derived class.");
    }
}

public class InheritanceExample {
    public static void main(String[] args) {
        Derived obj = new Derived();
        obj.display();
        obj.show();
    }
}
```

Sample Output/Result:

Part A Output:

Input: java CommandLineExample 10 20
Output: Sum of 10 and 20 is: 30

Part B Output:

This is the base class.

This is the derived class.

Inferences:

- **Initialization:**
    - Command-line arguments are initialized as strings in the args array and converted to the required data type.
    - Inheritance enables code reuse and establishes a parent-child relationship between classes.
- **Input:**
    - Command-line arguments are passed during program execution.
    - Inheritance requires defining and initializing objects for the respective classes.
- **Calculation:**
    - Basic arithmetic operations can be performed using input values.
    - Derived classes extend the functionality of base classes through additional methods.
- **Output:**
    - Outputs results based on operations performed on command-line inputs.
    - Displays messages from methods in base and derived classes.

Viva Questions:

1. What is the importance of command-line arguments in Java?
2. Explain the concept of method overriding in inheritance with an example.
3. How would you handle invalid or missing command-line arguments in your program?

Exp. No.: 1b To implement a program with Polymorphism, Interface, Abstract Classes, and Exception Handling **Date:**

Aim:

To write a Java program that demonstrates the use of polymorphism, interface, abstract classes, and exception handling.

Requirements:

- Java Compiler (Ubuntu)
- Eclipse

Suggested Reading (Theoretical Background):

Refer to **Module I**, Chapter **3.2, 3.4, and 3.6 (T1)** for understanding polymorphism, interface, abstract classes, and exception handling in Java.

Algorithm:

Part A: Polymorphism with Interface and Abstract Classes

1. Define an interface with abstract methods.
2. Create an abstract class that implements the interface partially.
3. Derive concrete classes from the abstract class to implement remaining methods.
4. Demonstrate polymorphism using method overriding.

Part B: Exception Handling

1. Write a try block to handle a specific operation that may throw an exception (e.g., division by zero).
2. Use a catch block to handle the exception.
3. Include a finally block to execute cleanup code, if necessary.

Sample Code:

```
// Part A: Polymorphism with Interface and Abstract Classes

interface Shape {
  void draw();
  double calculateArea();
}
```

```
abstract class Polygon implements Shape {
  int sides;

  Polygon(int sides) {
    this.sides = sides;
  }

  void displaySides() {
    System.out.println("This shape has " + sides + " sides.");
  }
}

class Rectangle extends Polygon {
  private double length, width;

  Rectangle(double length, double width) {
    super(4); // Rectangle has 4 sides
    this.length = length;
    this.width = width;
  }

  @Override
  public void draw() {
    System.out.println("Drawing a rectangle.");
  }

  @Override
  public double calculateArea() {
    return length * width;
  }
}
```

```
class Circle implements Shape {
    private double radius;

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

    @Override
    public void draw() {
        System.out.println("Drawing a circle.");
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}

// Part B: Exception Handling


public class ExceptionHandlingExample {
    public static void main(String[] args) {
        // Demonstrating polymorphism
        Shape rect = new Rectangle(5.0, 3.0);
        rect.draw();
        System.out.println("Area of Rectangle: " + rect.calculateArea());

        Shape circ = new Circle(4.0);
        circ.draw();
```

```
        System.out.println("Area of Circle: " + circ.calculateArea());

        // Demonstrating exception handling
        try {
            int num1 = 10, num2 = 0;
            System.out.println("Result: " + (num1 / num2));
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        } finally {
            System.out.println("Execution completed.");
        }
    }
}
```

Sample Output/Result:

Polymorphism Output:

Drawing a rectangle.

Area of Rectangle: 15.0

Drawing a circle.

Area of Circle: 50.26548245743669

Exception Handling Output:

Error: Division by zero is not allowed.

Execution completed.

Inferences:

- **Polymorphism:**
    - Interfaces and abstract classes facilitate code reusability and flexibility.

- Method overriding demonstrates dynamic polymorphism.

- **Exception Handling:**
  - Prevents program termination during runtime errors.
  - Ensures proper resource cleanup using the finally block.

Viva Questions:

1. How does Java achieve polymorphism using interfaces and abstract classes?
2. What is the significance of the finally block in exception handling?
3. Can an abstract class implement multiple interfaces? Justify your answer.