ACADEMIC PLANNER

For

***“*OBJECT ORIENTED PROGRAMMING THROUGH JAVA*”***

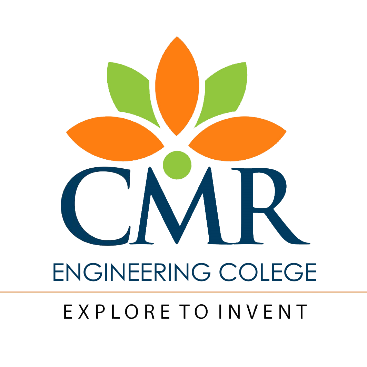
**Presented by**

**D SIVA RAJA KUMAR**

**Assistant Professor**

Department of

**CSE(AI&ML)**

****

**CMR ENGINEERING COLLEGE**

(Approved by AICTE-NewDelhi, Affiliated to J.N.T.U, Hyderabad)

Kandlakoya(v),Medchal Road,Hyderabad-501 401,Telangana State, India .Website: www.cmrec.ac.in

**(AY:2025-26)**

**ACADEMIC PLANNER**

**Subject: Object Oriented Programming through Java**

**S.NO CONTENT**

**(1) - Preamble/Introduction**

**(2) - Prerequisites**

**(3) - Objectives and Outcomes**

**(4) - Syllabus**

**1.R22-CMREC(CS304PC)**

**2.GATE**

**3.IES**

**(5) - List of Expert Details**(Local/National/International with Contact details/Profile link/Blogs/their research Contribution towards the subject)

**(6) - Journals with min 5 ref paper for literature study**

**(7) - Subject -Lesson plan**

**(8) - Suggested Books (**Prescribed and References)

**(9) - Websites for self-learning** **resources like**

*www.geeksforgeeks.org, www.schools.com,* [*Coursera*](https://www.theeducationmagazine.com/word-art/best-educational-websites/#Coursera)*,* [*edX*](https://www.theeducationmagazine.com/word-art/best-educational-websites/#edX)*,* [*Udemy*](https://www.theeducationmagazine.com/word-art/best-educational-websites/#Udemy)*,* [*Khan Academy*](https://www.theeducationmagazine.com/word-art/best-educational-websites/#Khan Academy), NPTEL etc along Registration procedures*)*

**(10) - Question Banks 1.JNTUH/Model papers**

**2.GATE**

**(11) - Two case study presentations with Project / Product/ Model /prototypes/ Industrial applications. (12) - Assignment Question/Innovative Assignments sets.**

**(13) - List of topics for students Seminars with Guidelines**

**(14) - STEP/Course material in softcopy**

**(15) - Expert Lectures with topics &Schedules(if any)**

1. **Introduction:**

Java is a programming language and a platform independent. Java is a high-level, robust, object-oriented, and secure programming language developed at Sun Microsystems Inc (which is now the subsidiary of Oracle) by James Gosling in 1991, The first publicly available version of Java (Java 1.0) was released in 1995. Later on, Java was acquired by Oracle Corporation. It is a platform-independent programming language because in Java, programs are compiled into byte code and that byte code is platform-independent. The byte code is executed by the Java Virtual Machine

1. **Prerequisites**
   * Students should have some knowledge of basic programming concepts like C and C++.
2. **OBJECTIVES & OUTCOMES**
3. **Course Objectives**

* To introduce object-oriented programming concepts.
* To understand OOP Concepts and apply them in solving problems
* To introduce the principles of Inheritance and Polymorphism and demonstrate how they relate to the design of the abstract classes.
* To introduce the implementation of Packages and Interfaces
* To introduce the concept of Exception Handling and Multithreading
* To introduce the design of a Graphical User Interface using applets and swing controls.

1. **Course Outcomes**
   1. Demonstrate basic programming constructs like control structures, constructors, string handling, and garbage collection.
   2. Explore and apply concepts related to member access rules, abstract classes, and interfaces.
   3. Use multithreading concepts to develop inter-process communication.
   4. Understand the process of graphical user interface design and implementation using AWT or Swing.
   5. Develop applets that interact abundantly with the client environment and deploy them on the server.

**4. SYLLABUS**

**4.1 CMREC Autonomous Syllabus**

**OBJECT ORIENTED PROGRAMMING THROUGH JAVA**

**UNIT-I**

**Object oriented thinking and Java Basics**- Need for OOP paradigm, summary of OOP concepts, coping with complexity, abstraction mechanisms. A way of viewing world – Agents, responsibility, messages, methods, History of Java, Java buzzwords, data types, variables, scope and life time of variables, arrays, operators, expressions, control statements, type conversion and casting, simple java program, concepts of classes, objects, constructors, methods, access control, this keyword, garbage collection, overloading methods and constructors, method binding, inheritance, overriding and exceptions, parameter passing, recursion, nested and inner classes, exploring string class.

**UNIT-II**

**Inheritance, Packages and Interfaces**–Hierarchical abstractions, Base class object, subclass, subtype, substitutability, forms of inheritance specialization, specification, construction, extension, limitation, combination, benefits of inheritance, costs of inheritance. Member access rules, super uses, using final with inheritance, polymorphism- method overriding, abstract classes, the Object class. Defining, Creating and Accessing a Package, Understanding CLASSPATH, importing packages, differences between classes and interfaces, defining an interface, implementing interface, applying interfaces, variables in interface and extending interfaces. Exploring java.io.

**UNIT-III**

**Exception handling and Multithreading**—Concepts of exception handling, benefits of exception handling, Termination or resumptive models, exception hierarchy, usage of try, catch, throw, throws and finally, built in exceptions, creating own exception subclasses. String handling, Exploring java.util. Differences between multithreading and multitasking, thread lifecycle, creating threads, thread priorities, synchronizing threads, inter thread communication, thread groups, daemon threads. Enumerations, auto boxing, annotations, generics.

**UNIT-IV**

**Event Handling:** Events, Event sources, Event classes, Event Listeners, Delegation event model, handling mouse and keyboard events, Adapter classes. The AWT class hierarchy, user interface components-labels, button, canvas, scroll bars, text components, checkbox, check box groups, choices, lists panels– scroll pane, dialogs, menu bar, graphics, layout manager –layout manager types–border, grid, flow, card and grid bag.

**UNIT-V**

**Applets** – Concepts of Applets, differences between applets and applications, life cycle of an applet, types of applets, creating applets, passing parameters to applets. Swing – Introduction, limitations of AWT, MVC architecture, components, containers, exploring swing-JApplet, JFrame and JComponent, Icons and Labels, text fields, buttons–The JButton class, Checkboxes, Radio buttons, Combo boxes, Tabbed Panes, Scroll Panes, Trees and Tables.

**4.2 GATE Syllabus**

NA (Note: Basic Programming knowledge is required to learn)

**4.3 IES**

NA

1. **Expert Details**
   * **International**

**James Gosling**

Engineer at Amazon Web Services (AWS) and the original developer of Java, Blog: [**nighthacks.com/roller/jag**](http://nighthacks.com/roller/jag)

* + **National**

# Dr. Uday Khedker

|  |  |
| --- | --- |
| |  | | --- | | Professor, [Department of Computer Science & Engg.,](http://www.cse.iitb.ac.in/) [IIT Bombay](http://www.iitb.ac.in/).  Contact No : 912572 0290 | |

**Mr**. **Shubham Giri**

Software Architect , Amdocs,Pune

Contact No-7869521631

* + **Local**

**Dr. D Nagaraj**

Professor, HOD, Department of CSE, Sri Venkatesa Perumal College of Engineering (SVPP), Puttur, Chittor.

Contact No-8015380054, Email id: nagarajd2005@gmail.com

1. **JOURNALS**

1. **Thread-Sensitive Data Race Detection for Java**

**Description:** In this paper we present StaTS, a precise static data-race detection mechanism for Java. It analyzes applications in four phases. The first one is a novel points-to analysis that includes approximations of threads and execution contexts. The second phase uses the results of the points-to analysis to compute which fields are accessed by which threads, while considering the locks held by the threads. The third phase carries out a context-sensitive static happens-before analysis to rule out accesses in execution contexts that can never be executed in parallel. The final phase builds upon the results of the first three to determine conflicting accesses and report them to the user. Our proof-of-concept implementation does not scale for large programs, which is why it can optionally limit the number of points-to relations it considers, based on sampling. Nevertheless, our evaluation shows that, even with sampling enabled for large programs, StaTS detects more data races than existing approaches. In terms of execution time, the analysis without sampling takes in the order of seconds for smaller programs. For larger ones and with sampling enabled, analysis takes minutes, thus being practically usable in nightly build environments in all cases.

**Citations:1**

Ref: https://doi.org/10.1109/APSEC53868.2021.00011

## ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an undergraduate Java Programming Course

**Description:** In this study, we assess the efficacy of employing the ChatGPT language model to generate solutions for coding exercises within an undergraduate Java programming course. ChatGPT, a large-scale, deep learning-driven natural language processing model, is capable of producing programming code based on textual input. Our evaluation involves analyzing ChatGPT-generated solutions for 80 diverse programming exercises and comparing them to the correct solutions. Our findings indicate that ChatGPT accurately generates Java programming solutions, which are characterized by high readability and well-structured organization. Additionally, the model can produce alternative, memory-efficient solutions. However, as a natural language processing model, ChatGPT struggles with coding exercises containing non-textual descriptions or class files, leading to invalid solutions. In conclusion, ChatGPT holds potential as a valuable tool for students seeking to overcome programming challenges and explore alternative approaches to solving coding problems. By understanding its limitations, educators can design coding exercises that minimize the potential for misuse as a cheating aid while maintaining their validity as assessment tools

**Citations:13**

**Ref:**- DOI:10.1145/3587102.3588794

## Code Writing Problems for Basic Object-Oriented Programming Study in Java Programming Learning Assistant System

**Description:** Nowadays, Java has been extensively used in industries as a reliable and portable object-oriented programming (OOP) language. To support novice students in learning Java programming, we have developed a web-based system called Java Programming Learning Assistant System (JPLAS). JPLAS offers the Code Writing Problem (CWP) among several types of exercise problems. A CWP instance requests a student to write a source code that will pass the provided test code in the assignment where the accuracy of the code is verified by running them on JUnit. In Java programming, every student should master basic OOP concepts, which can be challenging for novice students. This paper presents CWP instances for studying six basic OOP concepts including encapsulation, polymorphism, constructor, inheritance, abstract, and interface in JPLAS. Our preliminary evaluations of them confirm their validity.

**Citations:1**

**Ref:-** DOI:10.1145/3587102.3588794

## Novice Use of the Java Programming Language

**Description:** Java is a popular programming language for use in computing education, but it is difficult to get a wide picture of the issues that it presents for novices, most studies look only at the types or frequency of errors.

**Citations:7**

**Ref:-** DOI: 10.1145/3551393

## A Proposal of Grammar-Concept Understanding Problem in Java Programming Learning Assistant System

**Description:** Java is a popular programming language for use in computing education, but it is difficult to get a wide picture of the issues that it presents for novices, most studies look only at the types or frequency of errors.

**Citations:21**

**Ref:-** DOI:10.12720/jait.12.4.342-350

1. **Subject Lesson Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **Topic (syllabus)** | **Sub-Topic** | **NO. OF LECTURES REQUIRED** | **Suggested Books** | **Teaching Methods** |
| 1 | **Unit – 1** |  |  |  |  |
| 2 | **Object-Oriented Thinking** | Need for OOP paradigm, summary of OOP concepts. | 1 | T1, T2 | BB / PPT |
| 3 |  | coping with complexity, abstraction mechanisms. | 1 | T1, T2 | BB / PPT |
| 4 |  | A way of viewing world Agents and Communities, messages and methods, and Responsibilities | 1 | T1, T2 | BB / PPT |
| 5 |  | History of Java, Java buzzwords | 1 | T1, T2 | BB / PPT |
| 6 |  | data types, variables, scope and life time of variables | 1 | T1, T2 | BB / PPT |
| 7 |  | arrays, operators | 1 | T1, T2 | BB / PPT |
| 8 |  | expressions, control statements | 1 | T1, T2 | BB / PPT |
| 9 |  | type conversion and casting, simple java program | 1 | T1, T2 | BB / PPT |
| 10 |  | concepts of classes, objects | 1 | T1, T2 | BB / PPT |
| 11 |  | constructors, methods, |
| 12 |  | access control, this keyword, | 1 | T1, T2 | BB / PPT |
| 13 |  | garbage collection, overloading methods | 1 | T1, T2 | BB / PPT |
| 14 |  | constructors, method binding, inheritance | 1 | T1, T2 | BB / PPT |
| 15 |  | overriding and exceptions, parameter passing | 1 | T1, T2 | BB / PPT |
| 16 |  | recursion, nested and inner classes, exploring string class | 1 | T1, T2 | BB / PPT |
| 17 |  | **Total classes required for UNIT - 1** | 14 |  |  |
| 18 | **UNIT – 2** |  |  |  |  |
| 19 | **Inheritance** | Inheritance concept, Inheritance basics | 1 | T1, T2 | BB / PPT |
| 20 |  | Member access, Constructors | 1 | T1, T2 | BB / PPT |
| 21 |  | Creating Multilevel hierarchy |
| 22 |  | super uses, using final with inheritance | 1 | T1, T2 | BB / PPT |
| 23 |  | Polymorphism | 1 | T1, T2 | BB / PPT |
| 24 |  | method overriding, abstract classes, Object class | 1 | T1, T2 | BB / PPT |
| 25 |  | forms of inheritance- specialization, specification, construction, extension, limitation, combination | 1 | T1, T2,R1 | BB / PPT |
| 26 |  | benefits of inheritance and costs of inheritance | T1, T2 | BB / PPT |
| 27 | **Packages** | Defining a Package | 1 | T1, T2 | BB / PPT |
| 28 |  | Understanding CLASSPATH | T1, T2 | BB / PPT |
| 29 |  | Access protection | T1, T2,R3 | BB / PPT |
| 30 |  | Creating a Package, importing packages | 1 |  |  |
| 31 | **Interfaces** | defining an interface | 1 | T1, T2 | BB / PPT |
| implementing interfaces | T1, T2 | BB / PPT |
| 32 |  | Difference between class and interface | 1 | T1, T2 | BB / PPT |
| 33 |  | Nested interfaces, applying interfaces | T1, T2 | BB / PPT |
| 34 |  | variables in interfaces, extending interfaces | 1 | T1, T2,R4 | BB / PPT |
| 36 |  | Exploring Java.io.\*; | 1 | T1, T2,R4 | BB / PPT |
| 35 |  | **Total classes required for UNIT - 2** | 12 |  |  |
| 46 | **UNIT – 3** |  |  |  |  |
| 47 | **Exception handling** | Fundamentals of exception handling | 1 | T1, T2 | BB / PPT |
| 48 |  | Exception types | T1, T2 | BB / PPT |
| 49 |  | Termination or resumptive models | 1 | T1, T2 | BB / PPT |
| 50 |  | Uncaught exceptions | T1, T2 | BB / PPT |
| 51 |  | using try and catch | 1 | T1, T2 | BB / PPT |
| 52 |  | multiple catch clauses | T1, T2 | BB / PPT |
| 53 |  | nested try statements | 1 | T1, T2 | BB / PPT |
| 54 |  | throw, throws and finally | T1, T2 | BB / PPT |
| 55 |  | built- in exceptions | 1 | T1, T2 | BB / PPT |
| 56 |  | creating own exception sub classes | 1 | T1, T2 | BB / PPT |
|  |  | String handling, Explore Java.util | 1 | T1, T2 | BB / PPT |
| 57 | **Multithreading** | Differences between thread-based multitasking and process-based multitasking | 1 | T1, T2 | BB / PPT |
| 59 |  | Thread Life cycle | 1 | T1, T2 | BB / PPT |
| creating threads |
| 60 |  | thread priorities | 1 | T1, T2 | BB / PPT |
| 61 |  | synchronizing threads | T1, T2 | BB / PPT |
| 62 |  | inter thread communication | T1, T2 | BB / PPT |
|  |  | Thread groups, deamon threads,enumarations,autoboxing | 1 | T1, T2 | BB / PPT |
|  |  | Annatations,generics | 1 | T1, T2 | BB / PPT |
| 63 |  | **Total classes required for UNIT - 3** | 12 |  |  |
| 77 | **UNIT – 4** |  |  |  |  |
| 83 | **Event Handling** | Events, Event sources, Event Listeners, and Event classes | 1 | T1, T2 | BB / PPT |
| 84 |  | The Delegation event model | T1, T2 | BB / PPT |
|  |  | Handling mouse and keyboard events | 1 | T1, T2 | BB / PPT |
| 85 |  | Adapter classes | 1 | T1, T2 | BB / PPT |
| 86 |  | AWT class hierarchy | 1 | T1, T2 | BB / PPT |
| 87 |  | User interface components-labels, buttons, canvas, scrollbar | T1, T2 | BB / PPT |
| 88 |  | T1, T2 | BB / PPT |
|  |  | Txt components, checkbox, check box groups | 1 | T1, T2 | BB / PPT |
|  |  | choices, lists panels– scroll pane, | 1 | T1, T2 | BB / PPT |
|  |  | dialogs, menu bar, graphics | 1 | T1, T2 | BB / PPT |
|  |  | layout manager –layout manager types–border | 1 | T1, T2 | BB / PPT |
|  |  | grid, flow, card and grid bag. | 1 | T1, T2 | BB / PPT |
|  |  | **Total classes required for UNIT - 4** | 10 |  |  |
|  | **UNIT 5** |  |  |  |  |
| 89 | **Applets** | Concepts of Applets, Life cycle of Applet | 1 | T1, T2 | BB / PPT |
| 91 |  | Difference between Applets and Applications | 1 | T1, T2 | BB / PPT |
| 92 |  | passing parameters to applets |
| 93 |  | Creating a Swing Applet | 1 | T1, T2 | BB / PPT |
| 94 |  | Painting in Swing, A Paint example | 1 | T1, T2 | BB / PPT |
| 95 |  | Introduction and limitations of AWT | 1 | T1, T2 | BB / PPT |
|  |  | MVC architecture | 1 | T1, T2 | BB / PPT |
| 96 |  | Components | 1 | T1, T2 | BB / PPT |
| 97 |  | Containers | 2 | T1, T2 | BB / PPT |
| 99 |  | JCheck Box | T1, T2 | BB / PPT |
| 100 |  | JRadio Button | T1, T2 | BB / PPT |
| 101 |  | JTabbed Pane | T1, T2 | BB / PPT |
| 102 |  | JScroll Pane | T1, T2 | BB / PPT |
| 103 |  | JList | T1, T2 | BB / PPT |
| 104 |  | JCombo Box | T1, T2 | BB / PPT |
| 105 |  | Swing Menus | T1, T2 | BB / PPT |
| 106 |  | Dialogs | T1, T2 | BB / PPT |
| 107 |  | **Total classes required for UNIT - 5** | 9 | T1, T2 | BB / PPT |
| 108 |  | **Total classes required** | **57** |  |  |

**8. Suggested Books**

**Text Books**

1. Java The complete reference, 9th edition, Herbert Schildt, McGraw Hill Education (India) Pvt. Ltd.[T1]
2. Understanding Object-Oriented Programming with Java, updated edition, T. Budd, Pearson Education. [T2]

**Reference Books**

1. An Introduction to programming and OO design using Java, J. Nino and F.A. Hosch, John Wiley & sons [R1]
2. Introduction to Java programming, Y. Daniel Liang, Pearson Education. [R2]
3. Object Oriented Programming through Java, P. Radha Krishna, University Press. [R3]
4. Programming in Java, S. Malhotra, S. Chudhary, 2nd edition, Oxford Univ. Press. [R4]
5. Java Programming and Object-oriented Application Development, R. A. Johnson, Cengage Learning. [R5]
6. Core and Advanced Java, Black Book, Steven Holzner [R5]

**9.WEBSITES For Self Learning Resources**

* 1. <https://nptel.ac.in/courses/106/105/106105191/>
  2. <https://www.tutorialspoint.com/java/index.htm>
  3. <https://www.w3schools.com/java/default.asp>
  4. <https://www.programiz.com/java-programming>
  5. <https://www.guru99.com/java-tutorial.html>

**10. Question Banks**

****

**Short Answer Question**

1. Java is platform-independent and portable. - Justify
2. Java is distributed language – Justify
3. Java is dynamic and extensible language - Justify
4. What is JVM (Java Virtual Machine)?
5. List out Java development tools and explain any one from it
6. List out Java API and define any one of it
7. Draw the Java program structure
8. Differences between print() and println() in java.
9. Write down the rules for variable naming.
10. What are symbolic constants? Explain with example.
11. What is meaning of automatic type conversion?
12. List out the types of operators used in Java
13. How can you use ternary operator?
14. Define the break and continue statements.
15. List out the features of Java.
16. List the primitive and non-primitive data types used in Java.
17. Write the syntax of any one of the following loops – while, do...while, for loop.
18. Write various types of inheritance.
19. Define method overloading.
20. Define fields and methods of a class in java.
21. Define constructor. How do we invoke constructor in java?
22. Define method overriding.
23. Define Inheritance. How it is created in Java?
24. What is the use of auto boxing in java? Explain.
25. Define interface. Write the syntax for implementing an interface in a class.
26. What is the meaning of the *static* keyword?
27. What are the restrictions with static methods?
28. Differentiate between Classes and Interface.
29. Define the final variable and final methods.
30. Define class. How does it accomplish data hiding?
31. What is the meaning of the *abstract* keyword?
32. What are the conditions for using super() method.
33. Give examples of the Run-time error.
34. List out the different types of exception.
35. Define the terms: try, catch.
36. Define the term - stream, reader stream classes, writer stream classes.
37. Explain any two string methods.
38. List the java API packages.
39. List the methods of Reader or Writer class.
40. Define the term – Exception and Exception Handling.
41. What is event delegation?
42. List out methods of MouseListener.
43. Define (1) Event (2) Event Source (3) Event Class (4) Event Listener
44. Explain methods (1) setBackground( ) (2) setForeground( )
45. Write difference between java applet program and java application program.
46. List down methods for KeyEvent class and ItemEvent class.
47. What is synchronization and why is it important?
48. What is the purpose of String Tokenizer class? Explain.
49. What is an adapter class? Explain with an example.
50. What is a Collection Class? Give an example.

**Long Answer Question**

1. What is meant by byte code? Briefly explain how java is platform independent.
2. Explain the significance of public, private, and protected access specifiers in inheritance.
3. Explain different parts of a java program with an appropriate example.
4. How does polymorphism promote extensibility? Explain with example.
5. Explain the process of defining and creating package with suitable example.
6. Give an example where interface can be used to support multiple inheritance.
7. What is the accessibility of public method or field inside a non-public class or interface? Explain.
8. Describe the process of importing and accessing a package with suitable examples.
9. Differentiate between checked and unchecked exceptions with examples.
10. Write a program to create four threads using Runnable interface.
11. What are different ways to handle exceptions? Explain.
12. How many ways are possible in java to create multiple threaded programs? Discuss the difference between them.
13. Differentiate between ArrayList and Vector? Why ArrayList is faster than Vector? Explain.
14. How a Hashtable can change the iterator? Explain.
15. Explain the Bit Set and Calendar classes in detail?
16. Discuss the differences between HashList and HashMap, Set and List?
17. List and explain different types of Layout managers with suitable examples.
18. How to move/drag a component placed in a swing container? Explain.
19. Discuss about different applet display methods in brief.
20. What are the various components of a swing? Explain.
21. Write the significance of JVM.
22. How to implement polymorphism in java? Explain briefly.
23. What is an array? How do you declare an array in java? Give examples.
24. How to design and implement interface in java? Give an example.
25. Write about the methods available in the Character Streams? Discuss.
26. Distinguish between Byte Stream class and Character stream classes.
27. What is an exception? How the exceptions are handled in java?
28. Write a java program that illustrate the application of multiple catch statements.
29. Differentiate between multiprocessing and multi-threading. What is to be done to implement these in a Program?
30. Write a program that creates two threads. First thread prints numbers from 1 to 100 and other thread prints numbers from 100 to 1.
31. What are the similarities between ArrayList and Vector? Explain.
32. What is the difference between Iterator and ListIterator? Explain different ways to iterate over a list.
33. What is Comparable and Comparator interface? Differentiate between them.
34. What is the difference between init() and start() methods in Applet? When will each be executed?
35. Create an applet to draw a Cube and Circle shapes.
36. Write a program to create a frame window that responds to mount clicks.
37. Describe different levels of access protection available in java.
38. List the primitive data types available in java and explain.
39. How can you extend one interface by the other interface? Discuss.
40. Discuss about CLASSPATH environmental variable.
41. Write a java program that demonstrates how certain exception types are not allowed to be throw.
42. What is the importance of hashCode() and equals() method.
43. What is an applet? Explain the life cycle of an applet with a neat sketch.
44. Write a program to create a frame window that responds to key strokes.

**11.CASE STUDY**

**11.1 Blockchain-Based Voting System using Java**

The Blockchain-Based Voting System ensures secure, transparent elections by recording votes on a blockchain. Using Spring Boot for backend management, Web3j for blockchain integration, and cryptographic techniques for security, the system allows voters to cast and verify votes. Real-time results and voter anonymity are maintained, enhancing election integrity. In this java programming is used to develop the backend server that handles the core functionalities of the voting system, such as voter registration, authentication, and vote submission.

**11.2 Smart Agriculture Monitoring System using Java**

The Smart Agriculture Monitoring System leverages IoT sensors to collect real-time environmental data, such as soil moisture, temperature, and humidity. Using a Spring Boot backend for processing, Kafka for stream handling, and machine learning models for predictive analytics, the system optimizes irrigation schedules and monitors crop health. A user-friendly dashboard provides real-time data visualization and alerts, helping farmers make informed decisions to enhance crop yield and efficiency, In this java is used to develop the backend server that handles data collection from IoT sensors, processes this data, and stores it in a database. The backend also manages user authentication, sensor registration, and data analytics.

**12.Assignment and Innovative Assignment Questions**

* **Assignment – 1**

1. What is meant by byte code? Briefly explain how Java is platform independent?
2. Explain the significance of public, protected and private access specifiers in inheritance.
3. Write the significance of Java Virtual Machine.
4. How do we implement polymorphism in JAVA? Explain briefly.
5. How to design and implement an interface in java? Give an example.

* **Assignment – 2**

1. What is an Exception? How is an Exception handled in Java?
2. Differentiate between multiprocessing and multi-threading. What is to be done to implement these in a Program?
3. Explain the process of defining and creating a package with suitable examples.
4. What is Java Collections Framework? List out some benefits of Collections framework and explain.
5. What is an applet? Explain the life cycle of Applet with a neat sketch.

**important Question Sets of each Unit**

**Unit – 1**

1. What is Java? Explain the features of Java.
2. Describe the Java environment
3. Explain the structure of Java program
4. Explain the data types available in Java
5. Explain type casting with example
6. Explain the scope of variable
7. List out the decision making statements available in Java. Explain with example
8. List out the looping statements available in Java. Explain with example
9. Write various types of inheritance
10. Define inheritance. Describe different forms of inheritance.

**Unit – 2**

1. Explain the process of defining and creating package with suitable example.
2. Describe the various forms of implementing interface. Give an example of JAVA code for each case.
3. When do we declare a method or class abstract? Discuss with one Example.
4. Write short note on method overloading and method overriding.
5. Define the term - stream, reader stream classes, writer stream classes.

**Unit – 3**

1. Give examples of the Run-time error
2. Explain Arithmetic Exception with an example.
3. What is exception? Explain the syntax of try block and catch block with an example.
4. Describe the try and catch statements in detail.
5. Differentiate between multiprocessing and multi-threading. What is to be done to implement these in a Program?
6. Write a program that creates two threads. First thread prints numbers from 1 to 100 and other thread prints numbers from 100 to 1.

**Unit – 4**

1. Discuss the differences between HashList and HashMap, Set and List?
2. What are the similarities between ArrayList and Vector? Explain.
3. What is the difference between Iterator and ListIterator? Explain different ways to iterate over a list.
4. What is Comparable and Comparator interface? Differentiate between them.

**Unit – 5**

1. Explain Applet life cycle in detail.
2. Write short note on following components.
   1. Label
   2. TextField
   3. TextArea
   4. List
   5. Choice
   6. Button
   7. Checkbox
3. Define (1) Event (2) Event Source (3) Event Class (4) Event Listener

|  |
| --- |
| 1. Write difference between java applet program and java application program. |
| 1. List down methods for KeyEvent class and ItemEvent class |

**13. List of Topics for Students Seminar**

* Inheritance in java
* Exception handling in java
* Interfaces in java
* Packages in java
* Collections in java
* AWT and Swings

**14.** [**STEP MATERIAL**](file:///C:\Users\Azhar\Downloads\JAVA%20PROGRAMMING_STEP_MATERIAL.docx)—



**15. EXPERT LECTURE SCHEDULE:**

**Java Programming Mastery: From Beginner to Expert Level**–

The tentative period is the month of September last week or October First week of 2025.