Module 6 – Core Java

1. Introduction to Java

1) History of Java ?

Ans:-

→ Before Java : "OAK" :1991 : James Goasling : Console app :single user

→Renamed "Java" from "OAK" : 1995 : James Goasling : multiuser : console app & Desktop app

→ WORA: Write Once Run Anywhere

→Java Having 3 category

→1) J2SE(Core Java) :Java 2 Standard Edition

→2) J2EE(Adv .Java) : Java 2 Enterprise Edition

→3) Framework : (Hibernate, Spring , Spring Boot)

→4) Services : Web Services , Micro Services

2) Features of Java :

Ans:-

→ 1)simple
→2)OO
→3)interpreter : JVM : bytecode (class file) to machine code
→ 4)Robust :powerful
→5)secure
→6)dynamic
→7)high performance :10x
→8)multithreading :
→9)platform independent :
→10) portable

3) Understanding JVM, JRE, and JDK ?

Ans:-

→ JDK : Java Development Kit
→JRE : Java Runtime Env.
→JVM : Java Virtual Machine
→JIT : Just In Time

4) Setting up the Java environment and IDE (e.g., Eclipse, IntelliJ)?

Ans:-

→ Eclipse Setup :
→ Download and install Eclipse from the official website.
→ Launch Eclipse and create a new Java project.
→ Right-click on the project and select "Properties."
→ In the "Properties" window, select "Java Build Path" and add the JDK to the build path.

→ IntelliJ IDEA Setup :
→ Download and install IntelliJ IDEA from the official website.
→ Launch IntelliJ IDEA and create a new Java project.
→ In the "Project Structure" window, select "SDKs" and add the JDK.
→In the "Project Structure" window, select "Project" and set the project SDK to the JDK.

5) Java Program Structure (Packages, Classes, Methods)

Ans:-

→ Packages :

→ packages are used to organize related classes and interfaces.

→ The package declaration is the first statement in a Java source file. It specifies the package to which the classes and interfaces in the file belong.

→ A package is essentially a directory that contains a group of related classes and interfaces.

→ Classes :

→ classes are the basic building blocks of a program. A class is a blueprint or a template that defines the properties and behavior of an object.

→ A class declaration typically includes the public access modifier, the class keyword, and the name of the class.

→ The class body contains the members of the class, including fields, methods, and constructors.

→ Methods :

→ methods are blocks of code that perform a specific task. Methods are used to encapsulate logic and promote code reusability.

→ A method declaration typically includes the return type, method name, parameter list, and method body.

→There are two main types of methods in Java: instance methods and static methods.

Lab Exercise:

1) Install JDK and set up environment variables.

Ans:-

→ Installing JDK

→ 1. Download JDK: Go to the Oracle website (https://www.oracle.com/java/technologies/javase-downloads.html) and download the latest version of JDK for your operating system (Windows, macOS, or Linux).

→ 2. Run the Installer: Run the downloaded installer and follow the installation prompts.

→ 3. Choose Installation Location: Choose the installation location for JDK. The default location is usually C:\Program Files\Java\jdk-<version> on Windows or

/Library/Java/JavaVirtualMachines/jdk-<version>.jdk on macOS.

→ 4. Install JDK: The installer will install JDK and its components, including the Java Runtime Environment (JRE).

→ Setting Up Environment Variables

→ Windows

→ 1. Right-click on Computer: Right-click on the Computer icon and select Properties.

→ 2. Advanced System Settings: Click on Advanced system settings on the left side.

→ 3. Environment Variables: Click on Environment Variables.

→ 4. New System Variable: Under System Variables, scroll down and find the Path variable, then click New.

→ 5. Add JDK Bin Directory: Add the path to the JDK bin directory (usually C:\Program Files\Java\jdk-<version>\bin).

→ 6. Update Path Variable: Update the Path variable by adding the JDK bin directory to the end of the existing value, separated by a semicolon.

→ 7. OK: Click OK to close all the windows.

2) Compile and run the program using command-line tools (javac, java).

Ans:-

→ Step 1: Open a Terminal or Command Prompt

→ Open a terminal or command prompt on your computer.

→ Step 2: Navigate to the Program Directory

→ Navigate to the directory where your Java program is saved. You can use the cd command to change directories.

→ Step 3: Compile the Program using javac

→ Compile your Java program using the javac command. The basic syntax is:

→ bash

→ javac ProgramName.java

→ Replace ProgramName.java with the name of your Java program.

→ Step 4: Run the Program using java

→ Run your Java program using the java command. The basic syntax is:

→ bash

→ java Program Name

→ Replace Program Name with the name of your Java program (without the .java extension).

2. Data Types, Variables, and Operators

1) Primitive Data Types in Java (int, float, char, etc.)

Ans:-

→ Primitive data types are the basic data types that are built into the Java programming language.

→ Int: A 32-bit signed integer, ranging from.

→ Long: A 64-bit signed integer, ranging from.

→ Float: A 32-bit floating-point number, ranging from approximately.

→ Char: A single 16-bit Unicode character, ranging from.

→ Double: A 64-bit floating-point number, ranging from approximately.

2) Variable Declaration and Initialization?

Ans:-

→ Variable Declaration :
→ Java, a variable is declared using the following syntax :
→ type variable Name;

→Variable Initialization :
→ A variable can be initialized using the assignment operator (=):
→ type variable Name = value;

3) Operators: Arithmetic, Relational, Logical, Assignment, Unary, and Bitwise?

Ans:-

→ Arithmetic Operators :

→ Arithmetic operators are used to perform mathematical operations.

→ (addition) +

→ (subtraction) -

→ (multiplication) *

→ (division) /

→ (modulus) %

→ Relational Operators :

→ Relational operators are used to compare values.

→ == (equal to)

→ != (not equal to)

→ > (greater than)

→ < (less than)

→ >=(greater than to equal to)

→<= (less than to equal to)

→ Logical Operators :

→ Logical operators are used to combine conditions

→ AND (logical and)

→OR (logical or)

→NOT (logical not)

→Assignment Operators Assignment operators are used to assign values to variables.

→:= or = (assignment)

→+= (addition assignment)

→-= (subtraction assignment)

→*= (multiplication assignment)

→/= (division assignment)

→Unary Operators Unary operators operate on a single operand.

→

- (unary plus)

→

- (unary minus)

→NOT (logical not)

→Bitwise Operators Bitwise operators perform operations on bits.

→& (bitwise and)

→| (bitwise or)

→^ (bitwise x or)

→~ (bitwise not)

→<< (left shift)

→(right shift)

4) Type Conversion and Type Casting?

Ans:-

→Type Conversion :--

→Type conversion is the process of changing the data type of a value from one type to another.

→Type Casting :--

→Type casting is a specific type of type conversion that changes the way a value is interpreted without changing its underlying data.

3. Control Flow Statements

1) If-Else Statements?

Ans:-

→ if condition is true then your if block will be execute otherwise else block will be executed.

→ If-Else statements are a fundamental control structure in programming that allow you to execute different blocks of code based on conditions or decisions.

2) Switch Case Statements?

Ans:-

→ Switch Case statements are a control structure in programming that allows you to execute different blocks of code based on the value of a variable or expression.

3) Loops (For, While, Do-While)?

Ans:-

→ Loops are control structures in programming that allow you to execute a block of code repeatedly for a specified number of iterations.

→ For Loop: Used to iterate over a specific number of iterations.

→ While Loop: Used to iterate while a certain condition is true.

→ Do-While Loop: Used to iterate at least once and then continue while a certain condition is true.

4) Break and Continue Keyword?

Ans:-

→ BREAK Keyword:--

→ The BREAK keyword is used to exit a loop prematurely. When BREAK is encountered, the loop is terminated, and the program continues executing the code after the loop.

→ CONTINUE Keyword:--

→ The CONTINUE keyword is used to skip the current iteration of a loop and move on to the next iteration.

4. Classes and Objects

1) Defining a Class and Object in Java?

Ans:-

→**class** : is an collection of data member(variables) and member function(methods, process) with its behaviors.

→**object** : is a instance of an class

→ :when you create class variables also called..

→ :its uses new keyword and class constructor to create object

→ :access whole properties of an class except private

2) Constructors and Overloading

Ans:-

→ Constructors:-

→ A constructor is a special method in a class that is used to initialize objects when they are created. It has the same name as the class and does not have a return type, not even void.

→ Overloading:--

→ Method overloading is a feature in Java that allows a class to have more than one method with the same method name, but with different parameter lists.

3) Object Creation, Accessing Members of the Class

Ans:-

→ Object Creation:--

→ To create an object, you need to use the new keyword followed by the class name and parentheses.

→ Accessing Members of the Class :--

→ To access members of a class, you need to use the dot notation.

4) this Keyword?

Ans:-

→ Accessing class members: this is used to access class members, such as fields and methods, especially when there's ambiguity between class members and local variables.

→ Passing the current object as an argument: this can be passed as an argument to another method.

→ Returning the current object: this can be returned from a method.

→ Chaining method calls: this can be used to chain method calls.

→ Accessing inner classes: this can be used to access members of an inner class.

5. Methods in Java

1) Defining Methods

Ans:-

→ Method Definition:

→ A method is a block of code that performs a specific task. It's defined inside a class and can

be called multiple times from different parts of the program.

2) Method Parameters and Return Types

Ans:-

→ Method Parameters

→ Method parameters are the variables declared in the method signature that receive the values passed to the method when it's called.

→ Return Types

→ Return types specify the data type of the value returned by the method.

→ Types of Return Types

→ 1. Primitive Types: int, double, Boolean, etc.

→ 2. Reference Types: String, Array, Object, etc.

→ 3. Void: Indicates that the method does not return any value.

3) Method Overloading

Ans:-

→ Method overloading is a feature in Java that allows multiple methods with the same name to be defined, as long as they have different parameter lists.

4) Static Methods and Variables

Ans:-

→ Static Methods:--

→ Static methods belong to a class, rather than an instance of the class. They can be called without creating an instance of the class.

→ Static Variables:--

→ Static variables are shared by all instances of a class. They are initialized only once, when the class is loaded.

6. Object-Oriented Programming (OOPs) Concepts :

1) Basics of OOP: Encapsulation, Inheritance, Polymorphism, Abstraction?

Ans:-

→ Encapsulation : wrapping up of data into single unit i.e. :data hiding

→ private your data member and member function.

→ Inheritance : properties of parent class extends into child class

→ :properties of superclass extends into subclass

→ :main purpose is : Reusability , extendibility

→ :to used "extends" keyword through create inheritance

→ :always called last child class to create object with access the properties of parent class except private

→ Polymorphism : ability to take one name having many forms or different forms.

→ Abstraction :

→ Abstraction is the concept of showing only the necessary information to the outside world while hiding the implementation details.

2) Inheritance: Single, Multilevel, Hierarchical

Ans:-

→ 1) single : only one parent having only one child

→ 2) multilevel : single inheritance having one another child

→ 3) hierarchical : one parent having 2 or more child

3) Method Overriding and Dynamic Method Dispatch

Ans:-

→ method overriding(run time) : the whole signature of the method should be same in super

class as well as in subclass but its behaviors (body part of the method) are different.

→ method overloading(compile time) : the two or more method name should be same in a single class but its behaviors(data types, arguments) are different .

7. Constructors and Destructors

1) Constructor Types (Default, Parameterized)

Ans:-

→ 1)default : no any argument in constructor

→ 2) parameterized: may have one or more argument in constructor

2) Copy Constructor (Emulated in Java)

Ans:-

→ A copy constructor is a special constructor in C++ that creates a copy of an existing object. It's used to initialize an object from another object of the same class.

3) Constructor Overloading

Ans:-

→ Constructor overloading is a technique in Java where multiple constructors with different parameter lists can be defined for a class.

4) Object Life Cycle and Garbage Collection

Ans:-

→ Object Life Cycle

→ The object life cycle refers to the stages an object goes through from its creation to its destruction.

→ Garbage Collection

→ Garbage collection is the process of automatically reclaiming memory occupied by objects that are no longer in use.

8. Arrays and Strings

1) One-Dimensional and Multidimensional Arrays

Ans:-

→ 1)one dimensional : [] : at a time only one loop will be use

→ 2) two or more dimensional : [][] or [][][] : loop with in loop will be used.

2) String Handling in Java: String Class, String Buffer, StringBuilder

Ans:-

→ String Class

→ The String class in Java is an immutable class, meaning that once a String object is created, its value cannot be changed.

→ String Buffer Class

→ The String Buffer class in Java is a mutable class, meaning that its value can be changed after it's created.

→ StringBuilder Class

→ The StringBuilder class in Java is a mutable class, similar to String Buffer, but it's not thread-safe.

3) Array of Objects

Ans:-

→ An array of objects is a collection of objects of the same class stored in a single array variable.

4) String Methods (length, Charat, substring, etc.)

Ans:-

→ length() Returns the length of the string.

→ Charat() Returns the character at the specified index.

→ substring() Returns a substring of the original string.

→ index Of() Returns the index of the first occurrence of the specified character or substring.

→ listened() Returns the index of the last occurrence of the specified character or substring.

9. Inheritance and Polymorphism

1) Inheritance Types and Benefits

Ans:-

→ Single Inheritance: A child class inherits from a single parent class.

→ Multilevel Inheritance: A child class inherits from a parent class, which in turn inherits from another parent class.

→ Hierarchical Inheritance: Multiple child classes inherit from a single parent class.

→ Multiple Inheritance: A child class inherits from multiple parent classes (not supported in Java).

→ Hybrid Inheritance: A combination of multiple inheritance types.

→ Code Reusability: Inheritance promotes code reusability by allowing child classes to inherit properties and behavior from parent classes.

→ Improved Code Organization: Inheritance helps to organize code in a more structured and maintainable way.

→ Easier Maintenance: Inheritance makes it easier to maintain code by allowing changes to be made at the parent class level.

→ Increased Flexibility: Inheritance allows child classes to add new properties and behavior or override existing ones.

→ Better Error Handling: Inheritance enables better error handling by allowing child classes to handle errors in a more specific way.

2) Method Overriding

Ans:-

→ method overriding(run time) : the whole signature of the method should be same in super class as well as in subclass but its behaviors (body part of the method) are different

3) Dynamic Binding (Run-Time Polymorphism)

Ans:-

→ Dynamic binding, also known as run-time polymorphism, is a process in which the actual method to be invoked is determined at runtime, rather than at compile time.

4) Super Keyword and Method Hiding

Ans:-

→ super : when your super class variable name and subclass variable name

→ are same then you used superclass variable want to used value into subclass at that time

→ used super keyword with variable.

→ Method Hiding :

→ Method hiding occurs when a subclass provides a method with the same name, return type, and parameter list as a method in its superclass.

→ The subclass method hides the superclass method.

10. Interfaces and Abstract Classes

1) Abstract Classes and Methods

Ans:-

→ Abstract Classes :

→ An abstract class is a class that cannot be instantiated on its own and is intended to be inherited by other classes.

→ Abstract classes are used to provide a partial implementation of a class, leaving some methods to be implemented by subclasses.

→ Abstract Methods :

→ An abstract method is a method declared in an abstract class without an implementation. Abstract methods must be implemented by any non-abstract subclass.

2) Interfaces: Multiple Inheritance in Java

Ans:-

→ interface in Java is an abstract class that cannot be instantiated on its own and is used to define a contract or a set of methods that must be implemented by any class that implements it.

3) Implementing Multiple Interfaces

Ans:-

→ a class can implement multiple interfaces by separating the interface names with commas.

11. Packages and Access Modifiers

1) Java Packages: Built-in and User-Defined Packages

Ans:-

→ A Java package is a collection of related classes, interfaces, and other types that are organized together to provide a single unit of functionality.

→ Built-in Java Packages :

→ Java provides several built-in packages that contain classes and interfaces for various purposes.

→ User-Defined Packages :

→ You can create your own packages to organize your classes and interfaces. To create a user-defined package.

2) Access Modifiers: Private, Default, Protected, Public

Ans:-

→ Access modifiers are keywords used to specify the accessibility of classes, methods, and variables in Java.

→ They determine the scope of access, meaning who can access the class, method, or variable.

→ Private: Accessible only within the same class.

→ Default (no modifier): Accessible within the same package.

→ Protected: Accessible within the same package and by subclasses in other packages.

→ Public: Accessible from anywhere.

3) Importing Packages and Classpath

Ans:-

→ Importing Packages :

→ You can import packages to use classes, interfaces, and other types from those packages

without having to qualify them with their fully qualified name.

→ Classpath :

→ The class path is a parameter that tells the Java Virtual Machine (JVM) where to find the classes it needs to run a Java program.

12. Exception Handling

1) Types of Exceptions: Checked and Unchecked

Ans:-

→ Checked Exceptions: These are exceptions that are checked at compile-time.

→ The Java compiler checks if the code handles these exceptions or not.

→ If not, the compiler throws an error.

→ Unchecked Exceptions: These are exceptions that are not checked at compile-time.

→ They are also known as runtime exceptions.

2) try, catch, finally, throw, throws

Ans:-

→ Try :

→ The try keyword is used to define a block of code that may throw an exception.

→ Catch :

→ The catch keyword is used to define a block of code that handles an exception thrown by the try block.

→ Finally :

→ The finally keyword is used to define a block of code that is executed regardless of whether an exception is thrown or not.

→ Throw :

→ The throw keyword is used to throw an exception explicitly.

→ Throws :

→ The throws keyword is used to declare that a method may throw an exception.

3) Custom Exception Classes

Ans:-

→ A custom exception class is a user-defined exception class that extends the built-in Exception class or one of its subclasses.

→ Custom exception classes are used to create specific exceptions that can be thrown and caught in a program.

13. Multithreading

1) Introduction to Threads

Ans:-

→ Thread : its an light weight process or processor

→ :its totally depends on process

→ :its self class

→ :its derived from java. Lang package

→ :each and every program must have a thread i.e. main()

→ :when you start your code for executing.

2) Creating Threads by Extending Thread Class or Implementing Runnable Interface

Ans:-

→ Creating Threads

→ In Java, threads can be created in two ways:

→ 1. Extending the Thread Class: By extending the Thread class and overriding the run() method.

→ 2. Implementing the Runnable Interface: By implementing the Runnable interface and providing an implementation for the run() method.

3) Thread Life Cycle

Ans:-

→ newborn state : when you create object

→ runnable state: start()

→ running state: run()

→ blocked state: suspend()=>resume(), sleep(MS),wait()=>notify()

→ dead state: stop()

4) Synchronization and Inter-thread Communication

Ans:-

→ Synchronization :

→ Synchronization is the process of controlling the access of multiple threads to shared resources.

→ It ensures that only one thread can access the shared resource at a time, preventing data inconsistency and thread interference.

→ Inter-thread Communication :

→ Inter-thread communication is the process of exchanging data between threads.

→ It ensures that threads can communicate with each other to achieve a common goal.

14. File Handling

1) Introduction to File I/O in Java (java.io package)

Ans:-

→ File IO : File Input Output:

→ :so now I want to start permanent to store data into some where

→ :File it self class

→ :but you can create any type of file like .jpg .txt, .docs, .xlsx

→ :its derived from java.io package

→ : having stream

2) File Reader and File Writer Classes

Ans:-

→ FineReader Class

→ The FineReader class is a convenience class for reading character files. It provides a way to

read characters from a file using a variety of methods.

→ File Writer Class

→ The File Writer class is a convenience class for writing character files. It provides a way to write characters to a file using a variety of methods.

3) Buffered Reader and Buffered Writer

Ans:-

→ Buffered Reader Class :

→ The Buffered Reader class is a subclass of the Reader class. It provides a way to read text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

→ Buffered Writer Class :

→ The Buffered Writer class is a subclass of the Writer class. It provides a way to write text to a character-output stream, buffering characters so

as to provide for the efficient writing of single characters, arrays, and strings.

4) Serialization and Deserialization

Ans :-

→ Serialization is the process of converting an object's state into a byte stream. This allows the object to be saved to a file, sent over a network, or stored in a database.

→ Deserialization is the process of reconstructing an object from a byte stream. This allows the object to be restored to its original state.

15. Collections Framework

1) Introduction to Collections Framework

Ans:-

→ The Collections Framework is a set of classes and interfaces in Java that provide a unified

architecture for representing and manipulating collections of data.

→ It provides a standard way of storing and manipulating collections of objects.

2) List, Set, Map, and Queue Interfaces

Ans:-

→ List Interface :

→ The List interface is an ordered collection of elements that allows duplicates. It provides methods for accessing and manipulating elements in the list.

→ Set Interface :

→ The Set interface is an unordered collection of unique elements. It provides methods for accessing and manipulating elements in the set.

→ Map Interface :

→ The Map interface is an unordered collection of key-value pairs. It provides methods for accessing and manipulating key-value pairs in the map.

→ Queue Interface :

→ The Queue interface is an ordered collection of elements that allows duplicates. It provides methods for accessing and manipulating elements in the queue.

3) ArrayList, LinkedList, HashSet, Tree Set, HashMap, Tree Map

Ans:-

→ ArrayList : auto implemented List interface

→ :its represent like dynamic array

→ :automatically shrink and grow both

→ :Default size is 0

→ :duplicate values are allow

→ :when you add some value in it , so same way display on console

→ :add() and remove()

→ :default symbol is "[]"

→ 2) HashSet :auto implemented Set interface

→ :its represent like dynamic array

→ :automatically shrink and grow both
→ :Default size is 0
→ :duplicate values are not allow
→ :all the value has an Haskey
→ :all Haskey convert into hashcode
→ :all the value display via hash code wise
→ :add() and remove()
→ :default symbol is "[]"
→ 3) HashMap: auto implemented Map interface
→ :its represent like dynamic array
→ :automatically shrink and grow both
→ :Default size is 0
→ :its work look like in pair<k,v>
→ :duplicate pair are not allow, if key is same so value override
→ :all the pair has an Hashey
→ :all Hashey convert into hashcode
→ :all the pair display via Hash code wise
→ :put(),get() and remove()
→ :default symbol is "{}"
→ LinkedList : auto implemented List interface

→ :its represent like dynamic array/ doubly linked list

→ :automatically shrink and grow both

→ :Default size is 0

→ :duplicate values are allow

→ :when you add some value in it , so same way display on console

→ :add() and remove(),add first() and remove first(),add last() and remove last()

→ :default symbol is "[]"

→ Tree Set :

→ A Tree Set is a Navigable Set implementation based on a Tree Map. The elements are ordered using their natural ordering, or by a Comparator provided at set creation time, depending on which constructor is used.

→ Tree Map :

→ A Tree Map is a Navigable Map implementation based on a Tree Map. The map is sorted according to the natural ordering of its keys, or by a Comparator provided at map

creation time, depending on which constructor is used.

4) Iterators and List Iterators

Ans:-

→ Iterator :

→ An iterator is an object that allows you to traverse a collection of elements, such as a list, set, or map. It provides a way to access each element in the collection one at a time.

→ List Iterator :

→ A list iterator is a special type of iterator that allows you to traverse a list of elements in both forward and backward directions. It also provides methods to add and remove elements from the list.

16. Java Input/Output (I/O)

1) Streams in Java (Input Stream, Output Stream)

Ans:-

→ a stream is a sequence of data that can be read from or written to. Streams are used to transfer data between a program and a file, network, or other input/output device.

→ Input Stream: An Input Stream is used to read data from a source, such as a file or network connection.

→ Output Stream: An Output Stream is used to write data to a destination, such as a file or network connection.

2) Reading and Writing Data Using Streams

Ans:-

→ Reading Data Using Streams

→ To read data from a stream:

→ 1. Create a Stream: Create a stream object to read data from the source.

→ 2. Read Data: Use the read() method to read data from the stream.

→ 3. Process Data: Process the data as needed.

→ 4. Close the Stream: Close the stream to release system resources.

→ Writing Data Using Streams

→ To write data to a stream:

→ 1. Create a Stream: Create a stream object to write data to the destination.

→ 2. Write Data: Use the write() method to write data to the stream.

→ 3. Flush the Stream: Use the flush() method to ensure that all buffered data is written to the destination.

→ 4. Close the Stream: Close the stream to release system resources.

3) Handling File I/O Operations

Ans:-

→ Handling File I/O Operations

→ To handle file I/O operations:

→ 1. Import necessary classes: Import the necessary classes, such as File, FileInputStream, FileOutputStream, etc.

→ 2. Create a File object: Create a File object to represent the file you want to perform I/O operations on.

→ 3. Check if the file exists: Check if the file exists before performing I/O operations.

→ 4. Perform I/O operations: Perform the desired I/O operation, such as reading, writing, appending, or deleting.

→ 5. Handle exceptions: Handle any exceptions that may occur during I/O operations.

→ 6. Close resources: Close any resources, such as streams or files, after completing I/O operations.