Java Programming Final Exam Notebook (Modules 1-5)

Module 1: Java Basics

Java is a case-sensitive, object-oriented programming language. Every Java program starts with a class and a main method.

Syntax Essentials:

- Every statement ends with a semicolon (;)
- Blocks are defined using curly braces {}
- Class names use PascalCase, variable/method names use camelCase

Data Types:

- Primitive: int, float, double, char, boolean, long, short, byte
- Reference: String, arrays, objects

Operators:

- Arithmetic: +, -, *, /, %
- Relational: ==, !=, >, <, >=, <=
- Logical: &&, ||, !
- Assignment: =, +=, -=

I/O:

- Input: Scanner class (e.g., Scanner sc = new Scanner(System.in);)
- Output: System.out.print() and System.out.println()

Module 2: OOP in Java

OOP Principles:

1. Class & Object: Class is a blueprint; object is an instance of a class.
2. Constructor: Special method to initialize objects, same name as class, no return type.
3. Inheritance: Reusing code via 'extends' keyword; subclass inherits superclass properties.
4. Polymorphism:
   - Compile-time (Method Overloading): same method name, different parameters.
   - Runtime (Method Overriding): subclass redefines superclass method.
5. Abstraction: Hides implementation; achieved via abstract classes and interfaces.

6. Encapsulation: Wrapping data using private fields + public getters/setters to protect state.

Module 3: Control Structures & Collections

Control Statements:

- if, else if, else for conditional branching.

- switch for multi-case decision blocks.

Loops:

- for (known count), while (condition-checked first), do-while (condition-checked after).

Collections:

- Arrays: fixed-size, store same type.

- ArrayList: resizable, part of java.util.

- HashSet: unique unordered collection.

- HashMap: key-value storage.

Exception Handling:

- try-catch-finally blocks for error control.

- Checked exceptions (e.g., IOException) must be handled.

- Unchecked exceptions (e.g., ArithmeticException) may be optionally handled.

Module 4: Advanced Java

File I/O:

- FileReader/FileWriter for character streams.

- BufferedReader/BufferedWriter for efficient read/write.

- ObjectOutputStream/ObjectInputStream for object serialization.

Multithreading:

- Thread class or Runnable interface to run tasks in parallel.

- synchronized keyword ensures one thread accesses a block at a time.

Generics:

- Enable type safety with templates like <T>

- Used in collections (e.g., ArrayList<Integer>)

Java Programming Final Exam Notebook (Modules 1-5)

Annotations:

- @Override, @Deprecated, @SuppressWarnings

- Metadata that influences compiler or frameworks

Reflection:

- Allows inspection of classes, methods, fields at runtime using Class<?> APIs.

JDBC:

- Java API to connect to databases.

- Key interfaces: Connection, Statement, PreparedStatement, ResultSet.

Module 5: Frameworks and Architecture

JavaFX:

- GUI framework with Scene, Stage, Nodes

- FXML for layout, CSS for styling

Spring Framework:

- Modular framework built on Dependency Injection (DI) and Inversion of Control (IoC)

- Core annotations: @Component, @Autowired, @Service, @Repository

Spring Boot:

- Simplifies REST API development

- Uses embedded servers and annotations like @RestController, @GetMapping

Maven & Gradle:

- Build and dependency management tools.

- Maven uses XML (pom.xml), Gradle uses Groovy/Kotlin (build.gradle)

JUnit:

- Testing framework using @Test, assertions like assertEquals()

Microservices:

Java Programming Final Exam Notebook (Modules 1-5)

- Architecture of small, independent services

- Each service runs and scales independently

- Communicates via REST APIs, supported by Spring Boot + Spring Cloud