

# Java Session 1 ->10

take the below session and give the short important theory that covers all the points that i want to learn and all that covers the mcq also because i want to prepare for the mcq purpose exam also

Session 1: Introduction to Java

Lecture: □ Introduction to java

□ Features of java

□ JVM Architecture

□ JDK and its usage

□ Structure of java class □ Working with data types: Primitive data types

Below is **short, exam-oriented theory** for **Session 1: Introduction to Java**, written in **easy language, covers all points**, and is **perfect for MCQ preparation**.

You can revise this one page before exams. 

## Session 1 – Introduction to Java (MCQ Ready Notes)

### 1. Introduction to Java

- Java is a **high-level, object-oriented, and platform-independent** programming language.
- Developed by **James Gosling** at **Sun Microsystems** (now Oracle).
- Java programs are first compiled into **bytecode**, which runs on **JVM**.
- Java follows "**Write Once, Run Anywhere (WORA)**" concept.

#### MCQ Points

- Java is **not purely object-oriented** (because of primitive data types).
- Java source file extension → **.java**
- Compiled file extension → **.class**

### 2. Features of Java

1. **Simple** – Easy syntax, no pointers.
2. **Object-Oriented** – Uses class, object, inheritance, polymorphism.
3. **Platform Independent** – Bytecode runs on any OS with JVM.
4. **Secure** – No direct memory access, uses bytecode verification.
5. **Robust** – Strong memory management, exception handling.
6. **Multithreaded** – Supports multiple threads.
7. **Portable** – Same bytecode on all platforms.

8. **High Performance** – Uses JIT compiler.
9. **Distributed** – Supports networking (RMI, sockets).

### 📌 MCQ Trap

- Platform independent because of **JVM**, not OS.
  - Java does **not use pointers**.
- 

## 3. JVM Architecture

**JVM (Java Virtual Machine)** runs Java bytecode.

### Main Components

#### 1. Class Loader

- Loads `.class` file into memory.
- Types: Bootstrap, Extension, Application.

#### 2. Memory Areas

- **Method Area** – Class data, methods.
- **Heap** – Objects and instance variables.
- **Stack** – Method calls, local variables.
- **PC Register** – Keeps current instruction.
- **Native Method Stack** – Native code.

#### 3. Execution Engine

- **Interpreter** – Executes bytecode line by line.
- **JIT Compiler** – Improves performance.

#### 4. Garbage Collector

- Removes unused objects automatically.

### 📌 MCQ Points

- Objects are stored in **Heap memory**.
  - Local variables are stored in **Stack**.
  - JVM is **platform dependent**, Java is platform independent.
- 

## 4. JDK and Its Usage

- **JDK (Java Development Kit)** is used to **develop Java applications**.
- JDK includes:
  - **JRE** (Java Runtime Environment)

- **JVM**
- Development tools

## Important Tools

- **javac** – Compiler
- **java** – Runs program
- **javadoc** – Documentation
- **jar** – Packaging

### 📌 MCQ Difference

| Term | Purpose     |
|------|-------------|
| JDK  | Development |
| JRE  | Running     |
| JVM  | Execution   |

## 5. Structure of Java Class

Basic structure:

```
java

class ClassName {
    public static void main(String[] args) {
        // statements
    }
}
```

### Explanation

- **class** – Keyword to create class
- **main()** – Entry point of program
- **public** – Accessible everywhere
- **static** – No object needed
- **void** – No return value
- **String[] args** – Command line arguments

### 📌 MCQ Points

- Program execution starts from **main() method**.
- File name must match **class name** (public class).

## 6. Primitive Data Types in Java

Java has **8 primitive data types.**

| Data Type | Size    | Example                       |
|-----------|---------|-------------------------------|
| byte      | 1 byte  | <code>byte a = 10;</code>     |
| short     | 2 bytes | <code>short b = 100;</code>   |
| int       | 4 bytes | <code>int c = 1000;</code>    |
| long      | 8 bytes | <code>long d = 100L;</code>   |
| float     | 4 bytes | <code>float e = 10.5f;</code> |
| double    | 8 bytes | <code>double f = 20.5;</code> |
| char      | 2 bytes | <code>char g = 'A';</code>    |
| boolean   | 1 bit   | <code>true / false</code>     |

### 📌 MCQ Traps

- Default integer type → **int**
- Default decimal type → **double**
- **char** uses **Unicode**
- **boolean** size is **not fixed in bytes**

## Important MCQ One-Liners

- Java is **compiled and interpreted**.
- Java does **not support multiple inheritance using classes**.
- JVM converts bytecode into **machine code**.
- Java uses **automatic garbage collection**.
- Entry point of Java program → **main()**

If you want, I can also give:

- Only MCQs (with answers)
- One-page revision sheet
- Previous year MCQ patterns
- Tricky MCQ questions

Just tell me 

Sessions 2 & 3: Basic programming concepts

Lecture: □ Java Tokens □ Declaring variables and methods □ Data type compatibility □ Operators □

Below is a **short, exam-oriented theory** for **Sessions 2 & 3: Basic Programming Concepts**, written in **easy language, covers all topics**, and is **perfect for MCQ-based exams**.

You can revise this fully for objective questions. 

## Sessions 2 & 3 – Basic Programming Concepts (MCQ Ready Notes)

### 1. Java Tokens

**Tokens** are the **smallest units** of a Java program.

#### Types of Tokens

1. **Keywords** – Reserved words (`int, class, if, for`)
2. **Identifiers** – Names of variables, methods, classes
3. **Literals** – Fixed values (`10, 3.14, 'A'`)
4. **Operators** – Symbols for operations (`+, =, >`)
5. **Separators** – Symbols (`;, , (), {}, []`)

#### MCQ Points

- Keywords **cannot be used as identifiers**
- Java has **50+ keywords**
- Identifiers are **case-sensitive**

### 2. Declaring Variables and Methods

#### Variable Declaration

```
java
int a = 10;
```

- `int` → data type
- `a` → variable name
- `10` → value

#### Types of Variables

1. **Local variable** – Inside method

2. **Instance variable** – Inside class, outside method

3. **Static variable** – Declared using **static**

## Method Declaration

```
java

public int add(int x, int y) {
    return x + y;
}
```

### MCQ Points

- Local variables have **no default value**
- Instance variables have **default values**
- Method must have **return type**

## 3. Data Type Compatibility

- Determines whether a value can be stored in a variable.
- Java supports **automatic type conversion** (widening).

### Widening (Implicit)

```
java

int a = 10;
double b = a;
```

### Narrowing (Explicit)

```
java

double a = 10.5;
int b = (int)a;
```

### MCQ Points

- Widening is **automatic**
- Narrowing requires **type casting**
- boolean is **not compatible** with other data types

## 4. Operators in Java

### Types of Operators

1. **Arithmetic** – + - \* / %

2. **Relational** - `>` `<` `>=` `<=` `==` `!=`

3. **Logical** - `&&` `||` `!`

4. **Assignment** - `=` `+=` `-=`

5. **Unary** - `++` `--`

6. **Bitwise** - `&` `|` `^`

7. **Ternary** - `?:`

Example:

```
java
int max = (a > b) ? a : b;
```

### 📌 MCQ Traps

- `==` compares **value**, not object
- `%` gives **remainder**
- `&&` is **short-circuit operator**

## 5. Control Statements

Used to **control the flow** of program execution.

### Types

#### Decision Making

- `if`
- `if-else`
- `switch`

#### Looping

- `for`
- `while`
- `do-while`

#### Jumping

- `break`
- `continue`
- `return`

### 📌 MCQ Points

- `switch` works with `int, char, String`
  - `do-while` executes **at least once**
  - `break` exits loop
  - `continue` skips iteration
- 

## 6. Arrays in Java

Array stores **multiple values of same data type.**

### 1-D Array

```
java
int[] a = {1, 2, 3};
```

### Multidimensional Array

```
java
int[][] a = {
    {1, 2},
    {3, 4}
};
```

### Important Properties

- Index starts from **0**
- Size is **fixed**
- Stored in **Heap memory**

### MCQ Traps

- Array index out of bound → `ArrayIndexOutOfBoundsException`
- `length` is a **property**, not method
- Multidimensional arrays are **arrays of arrays**

### Important MCQ One-Liners

- Java is **case sensitive**
- Default value of instance int → **0**
- Logical operators return **boolean**
- `for` loop is entry-controlled loop
- Arrays are **objects** in Java

## ✓ Quick MCQ Revision Keywords

Tokens • Identifiers • Widening • Narrowing • Type Casting • Short Circuit • Entry Controlled Loop • Heap Memory • Array Index

If you want next:

- ♦ Only MCQs with answers
- ♦ Tricky MCQs
- ♦ Previous year MCQs
- ♦ Session 4 notes

Just tell me 

Lecture: □ Introduction to OOP  
 □ Classes and Objects □ OOP principles □ Encapsulation, Abstraction, Inheritance and Polymorphism

Below is **short, crisp, MCQ-oriented theory for OOP in Java**, written in **easy language** and covering **all asked points**.

This is **perfect for objective / MCQ exams** and quick revision. ✓

## Introduction to OOP (MCQ Ready Notes)

### 1. Introduction to OOP

**OOP (Object-Oriented Programming)** is a programming approach based on **objects**, not functions.

- Object = **Real-world entity**
- Focuses on **data security, reusability, and flexibility**
- Java follows **OOP model** (partially)

#### MCQ Points

- Java is **not 100% OOP** (uses primitive data types)
- OOP improves **code reuse** and **Maintainability**

### 2. Classes and Objects

#### Class

- Blueprint or template.
- Contains **variables (data)** and **methods (functions)**.

```
java

class Car {
    int speed;
    void drive() {
        System.out.println("Driving");
    }
}
```

## Object

- Instance of a class.
- Created using **new** keyword.

```
java

Car c = new Car();
```

### 📌 MCQ Traps

- Class does **not occupy memory**
- Object **occupies memory**
- One class → **multiple objects**

## 3. OOP Principles

There are **4 main OOP principles**:

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

### 📌 MCQ

- OOP pillars = **4**

## 4. Encapsulation

**Wrapping data and methods into a single unit (class) and protecting data.**

### How in Java

- Use **private variables**

- Access using **getters and setters**

```
java

class Student {
    private int marks;

    public int getMarks() {
        return marks;
    }
}
```

### 📌 MCQ Points

- Encapsulation = **Data hiding**
- Achieved using **access modifiers**
- Improves **security**

## 5. Abstraction

Hiding internal implementation and showing **only essential features**.

### Achieved Using

1. **Abstract class**
2. **Interface**

```
java

interface Animal {
    void sound();
}
```

### 📌 MCQ Traps

- Abstraction focuses on **what**, not **how**
- Interface supports **100% abstraction**
- Abstract class can have **abstract and non-abstract methods**

## 6. Inheritance

**One class acquiring properties of another class.**

```
java

class A {
    int x;
}
class B extends A {
    int y;
}
```

## Types

- Single
- Multilevel
- Hierarchical

### 📌 MCQ Points

- Java does **not support multiple inheritance using classes**
- Achieved using **extends** keyword
- Parent class = **Superclass**

## 7. Polymorphism

**One name, many forms.**

### Types

1. **Compile-time** – Method overloading
2. **Run-time** – Method overriding

```
java

class A {
    void show() {}
}
class B extends A {
    void show() {}
}
```

### 📌 MCQ Traps

- Overloading → same method name, **different parameters**
- Overriding → same method name, **same signature**
- Runtime polymorphism uses **dynamic binding**

## Important MCQ One-Liners

- **new** keyword creates object
- Object is **runtime entity**
- Encapsulation improves **security**
- Inheritance improves **reusability**
- Polymorphism improves **flexibility**
- Interface uses **implements** keyword

# Quick MCQ Keywords

Object • Class • Data Hiding • Dynamic Binding • Method Overloading • Method Overriding • Superclass • Subclass

If you want next:

- **Only OOP MCQs with answers**
- **Tricky interview MCQs**
- **OOP comparison table**
- **Next Java session notes**

Just tell me 

Session 5:

Lecture: □ Static variables and methods □ Accessing static variables and methods of different class  
 □ Introduction to reference data types □ Reference variables and methods □ Difference between reference data types and primitive data types □ Difference between reference variable and static variable

Below is **short, clear, MCQ-oriented theory** for **Session 5**, written in **easy language** and covering **all listed points**.

This is ideal for **objective / MCQ exams** and fast revision. 

## Session 5 – Static Members & Reference Data Types (MCQ Ready Notes)

### 1. Static Variables

- Declared using **static keyword**.
- Belongs to **class**, not object.
- **Single copy** shared by all objects.
- Stored in **Method Area**.

```
java

class Test {
    static int count = 0;
}
```

#### MCQ Points

- Static variables are created **once**
- Can be accessed using **class name**
- Static variables get **default values**

## 2. Static Methods

- Declared using **static keyword**.
- Can access **only static data directly**.
- Cannot use **this or super**.

```
java

class Test {
    static void show() {
        System.out.println("Static method");
    }
}
```

### MCQ Traps

- Static method **cannot access non-static members directly**
- **main()** method is **static**

## 3. Accessing Static Variables and Methods of Different Class

```
java

class A {
    static int x = 10;
    static void display() {
        System.out.println(x);
    }
}

class B {
    public static void main(String[] args) {
        System.out.println(A.x);
        A.display();
    }
}
```

### MCQ Points

- Access using **ClassName.member**
- No object creation needed

## 4. Introduction to Reference Data Types

- Reference data types store **address of object**, not value.

- Examples:
  - Class objects
  - Arrays
  - String
  - Interfaces

```
java
Student s = new Student();
```

### 📌 MCQ Points

- Reference variables store **memory address**
- Objects are stored in **Heap memory**

## 5. Reference Variables and Methods

- Reference variable refers to an object.
- Used to access **instance variables and methods**.

```
java
Car c = new Car();
c.drive();
```

### 📌 MCQ Traps

- Reference variable does **not store object**
- Multiple references can point to **same object**

## 6. Reference Data Types vs Primitive Data Types

| Primitive Data Types | Reference Data Types   |
|----------------------|------------------------|
| Store actual value   | Store object address   |
| Fixed size           | Size depends on object |
| Not objects          | Objects                |
| Stored in stack      | Object in heap         |
| Cannot be null       | Can be null            |

### 📌 MCQ Key

- **int, float** → primitive
- **String, array** → reference

## 7. Reference Variable vs Static Variable

| Reference Variable    | Static Variable          |
|-----------------------|--------------------------|
| Refers to object      | Belongs to class         |
| Created at runtime    | Created at class loading |
| Multiple copies       | Single copy              |
| Access via object     | Access via class name    |
| Points to heap object | Stored in method area    |

### 📌 MCQ Trap

- Static variable is **shared**
- Reference variable is **individual**

## Important MCQ One-Liners

- Static members belong to **class**
- Reference variable holds **address**
- Objects are created using **new**
- Static methods cannot use **this**
- Multiple objects share static variable

## Quick MCQ Keywords

Static • Class Level • Heap Memory • Method Area • Reference Variable • Object Address • Shared Data

If you want next:

- Only MCQs with answers (Session 5)
- Tricky MCQs
- Comparison tables only
- Next session notes

Just tell me 

Session 6:

Lecture: □ Constructors, initializing reference variables using constructors. □ Pass by value v/s pass

by reference. □ Re-assigning a reference variable. □ Passing reference variable to method  
 □ Initializing reference variable of different class □ Heap memory and stack memory

Below is **short, clear, MCQ-oriented theory** for **Session 6**, written in **easy language** and covering **all listed topics**.

This is **perfect for objective / MCQ exams** and quick revision. ✓

## Session 6 – Constructors & Reference Handling (MCQ Ready Notes)

### 1. Constructors

- A **constructor** is a **special method** used to **initialize objects**.
- Constructor name = **class name**.
- No return type (not even **void**).
- Called **automatically** when object is created.

```
java

class Student {
    int id;
    Student(int i) {
        id = i;
    }
}
```

#### 📌 MCQ Points

- Constructor is **not inherited**
- Can be **overloaded**
- If no constructor → **default constructor** provided by compiler

### 2. Initializing Reference Variables Using Constructors

- Reference variable gets linked to object created by constructor.

```
java

Student s = new Student(101);
```

#### 📌 MCQ Key

- **new** allocates memory in **heap**
- Reference variable is stored in **stack**

## 3. Pass by Value vs Pass by Reference (Java Concept)

### Important Rule

👉 Java is always pass by value

### Primitive Type

- Copy of value is passed
- Original value does **not change**

```
java

void change(int x) {
    x = 50;
}
```

### Reference Type

- Copy of reference is passed
- Object data **can change**

```
java

void update(Student s) {
    s.id = 200;
}
```

### 📌 MCQ Trap

- Java does **not support true pass by reference**
- Reference value (address) is passed by value

## 4. Re-assigning a Reference Variable

- Reference variable can point to **new object**.

```
java

Student s1 = new Student(1);
s1 = new Student(2);
```

### 📌 MCQ Points

- Old object becomes **eligible for garbage collection**
- Reference can be **null**

## 5. Passing Reference Variable to Method

- Object reference is passed to method.
- Changes inside method affect original object.

```
java

void modify(Student s) {
    s.id = 300;
}
```

### MCQ Key

- Object state can be modified
- Reference itself is **not replaced**

## 6. Initializing Reference Variable of Different Class

```
java

class A {
    int x = 10;
}

class B {
    public static void main(String[] args) {
        A obj = new A();
        System.out.println(obj.x);
    }
}
```

### MCQ Points

- Use **new** keyword
- Access members using **dot operator**

## 7. Heap Memory and Stack Memory

### Heap Memory

- Stores **objects**
- Large size
- Shared among threads
- Garbage collected

### Stack Memory

- Stores **local variables and references**
- Faster

- Thread-safe
- No garbage collection

### MCQ Table

| Stack             | Heap            |
|-------------------|-----------------|
| Stores references | Stores objects  |
| Faster            | Slower          |
| LIFO              | Dynamic         |
| Thread-safe       | Not thread-safe |

## Important MCQ One-Liners

- Constructors initialize objects
- `new` creates object in heap
- Java is pass by value
- Reference variable stores address
- Reassigned object → eligible for GC

## Quick MCQ Keywords

Constructor • new keyword • Heap Memory • Stack Memory • Pass by Value • Reference Copy • Garbage Collection

If you want next:

- Only MCQs with answers (Session 6)
- Tricky MCQs
- Memory diagram explanation
- Next session notes

Just tell me 

### Session 7:

Lecture: □ Inheritance: single & multilevel □ Inheritance: Hierarchical □ Association, Aggregation and Composition

□ Polymorphism: Compile time and runtime polymorphism

ACTS, Pune

PG-DAC Page 3 of 6

□ Rules of overriding and overloading of methods □ super and this keyword

Below is **short, crisp, MCQ-oriented theory** for **Session 7**, written in **easy language** and covering **all listed topics**.

This is **perfect for objective / MCQ exams** and fast revision. 

## Session 7 – Inheritance, Polymorphism & Relationships (MCQ Ready Notes)

### 1. Inheritance

Inheritance means **one class acquires properties of another class**.

- Keyword used: **extends**
- Parent class → **Superclass**
- Child class → **Subclass**

```
java
class A { }
class B extends A { }
```

#### MCQ Points

- Improves **code reusability**
- Java supports inheritance using **classes and interfaces**

### 2. Types of Inheritance

#### a) Single Inheritance

- One parent → one child

```
java
class A { }
class B extends A { }
```

#### b) Multilevel Inheritance

- Chain of inheritance

```
java
class A { }
class B extends A { }
```

```
class C extends B { }
```

### 📌 MCQ

- Java supports **multilevel inheritance**

## c) Hierarchical Inheritance

- One parent → multiple children

```
java

class A { }
class B extends A { }
class C extends A { }
```

### 📌 MCQ Trap

- Java does **not support multiple inheritance using classes**

## 3. Association, Aggregation & Composition

### Association

- General relationship** between two classes.
- Objects are **independent**.

Example: Teacher — Student

### Aggregation

- Weak “has-a” relationship**
- Child can exist **without parent**

Example: Department has Teachers

### Composition

- Strong “has-a” relationship**
- Child **cannot exist without parent**

Example: House has Rooms

### 📌 MCQ Table

| Type        | Relationship | Dependency |
|-------------|--------------|------------|
| Association | Uses         | No         |
| Aggregation | Has-a (weak) | Partial    |

| Type        | Relationship   | Dependency |
|-------------|----------------|------------|
| Composition | Has-a (strong) | Full       |

## 4. Polymorphism

Polymorphism means **one method, many forms**.

### a) Compile-Time Polymorphism

- Achieved using **method overloading**
- Happens at **compile time**

```
java
void add(int a);
void add(int a, int b);
```



- Overloading → same name, different parameters

### b) Runtime Polymorphism

- Achieved using **method overriding**
- Happens at **runtime**
- Requires **inheritance**

```
java
class A {
    void show() {}
}
class B extends A {
    void show() {}
}
```



- Runtime polymorphism uses **dynamic binding**

## 5. Rules of Method Overloading

- Same method name
- Different **number/type/order** of parameters
- Can change return type (with different parameters)

- Happens in **same class**

### 📌 MCQ Trap

- Overloading does **not depend on return type alone**

## 6. Rules of Method Overriding

- Same method name
- Same parameters
- Must use **inheritance**
- Access level cannot be reduced
- **final, static, private** methods **cannot be overridden**

### 📌 MCQ Key

- Overriding happens at **runtime**

## 7. super Keyword

Used to refer to **parent class**.

### Uses

- Access parent variable
- Call parent method
- Call parent constructor

```
java
```

```
super.method();
super();
```

### 📌 MCQ

- **super()** must be **first statement** in constructor

## 8. this Keyword

Used to refer to **current object**.

### Uses

- Resolve naming conflict
- Call current class method

- Call constructor

```
java

this.x = x;
this();
```

### 📌 MCQ

- `this()` calls current class constructor

## Important MCQ One-Liners

- Java supports **single, multilevel, hierarchical inheritance**
- Multiple inheritance is supported using **interfaces**
- Compile-time polymorphism → overloading
- Runtime polymorphism → overriding
- `super` refers to parent
- `this` refers to current object

## Quick MCQ Keywords

Inheritance • Overloading • Overriding • Dynamic Binding • Has-a Relationship • `super` • `this` • Polymorphism

If you want next:

- Only MCQs with answers (Session 7)
- Tricky inheritance MCQs
- Diagram-based explanations
- Next session notes

Just tell me 

Session 8:

Lecture: □ Upcasting and downcasting of a reference variable  
 □ Abstract class and abstract methods □ Interface (implementing multiple interfaces)  
 Sessions 9 & 10:  
 Lecture: □ Final variables, final methods and final class □ Functional interface  
 □ New interface features (Java 8 & 11) □ Lambda Expression  
 □ Inner Class (Regular, Method local, Anonymous & static inner class) □ Enum

Below is **short, exam-oriented, MCQ-ready theory** for **Session 8, 9 & 10**, written in **simple language**, covering **all listed points** clearly.

This is ideal for **PG-DAC / MCQ exams** and quick revision. ✓

## Session 8 – Casting, Abstract Class & Interface (MCQ Ready Notes)

### 1. Upcasting and Downcasting of Reference Variable

#### Upcasting

- Child object referred by **parent reference**
- Happens **automatically**
- Safe

```
java
Parent p = new Child();
```

#### 📌 MCQ Points

- Supports **runtime polymorphism**
- Only parent methods accessible

#### Downcasting

- Parent reference converted to **child reference**
- Requires **explicit casting**
- May cause **ClassCastException**

```
java
Child c = (Child) p;
```

#### 📌 MCQ Trap

- Downcasting works **only if object is child type**

### 2. Abstract Class and Abstract Methods

- Declared using **abstract keyword**
- Cannot create object
- Can have:
  - Abstract methods

- Non-abstract methods
- Constructors
- Variables

```
java

abstract class Shape {
    abstract void draw();
}
```

### 📌 MCQ Points

- Abstract method has **no body**
- Child class must **implement all abstract methods**
- Abstract class can have **0-100% abstraction**

## 3. Interface (Multiple Interfaces)

- Achieves **100% abstraction**
- Supports **multiple inheritance**
- Uses **implements keyword**

```
java

interface A { void m1(); }
interface B { void m2(); }

class C implements A, B {
    public void m1() {}
    public void m2() {}
}
```

### 📌 MCQ Traps

- Interface methods are **public by default**
- Variables are **public static final**
- Cannot create object of interface

## Sessions 9 & 10 – Advanced Java Concepts (MCQ Ready Notes)

## 4. Final Keyword

### Final Variable

- Value **cannot change**

```
java
final int x = 10;
```

## Final Method

- Cannot be **overridden**

## Final Class

- Cannot be **inherited**

```
java
final class Test {}
```

### 📌 MCQ Points

- **String** class is **final**
- Final methods prevent overriding

## 5. Functional Interface

- Interface with **only one abstract method**
- Used with **Lambda expressions**
- Annotation: **@FunctionalInterface**

```
java
@FunctionalInterface
interface A {
    void show();
}
```

### 📌 MCQ Trap

- Can have **default and static methods**

## 6. New Interface Features (Java 8 & 11)

### Java 8

- **Default methods**
- **Static methods**
- Lambda support

```
java

default void msg() {}
static void show() {}
```

## Java 11

- **Private methods** in interface

```
java

private void helper() {}
```

### 📌 MCQ Key

- Default methods support **backward compatibility**

## 7. Lambda Expression

- Short form of **anonymous function**
- Used with **functional interfaces**

```
java

() -> System.out.println("Hello");
```

### 📌 MCQ Points

- Reduces code
- Introduced in **Java 8**
- No method name, no return type

## 8. Inner Classes

Classes inside another class.

### Types

1. **Regular (Non-static) Inner Class**
2. **Method Local Inner Class**
3. **Anonymous Inner Class**
4. **Static Inner Class**

### 📌 MCQ Traps

- Inner class can access **private members**

- Static inner class cannot access non-static data

## 9. Enum

- Used to define **fixed constants**
- Type-safe

```
java

enum Day {
    MON, TUE, WED
}
```

### 📌 MCQ Points

- Enum constants are **public static final**
- Enum extends **java.lang.Enum**
- Enum constructor is **private**

## Important MCQ One-Liners

- Upcasting is implicit
- Downcasting is explicit
- Abstract class cannot be instantiated
- Interface supports multiple inheritance
- Lambda uses ->
- Functional interface has one abstract method
- Enum constants are fixed

## Quick MCQ Keywords

Upcasting • Downcasting • Abstract Class • Interface • Final • Lambda • Functional Interface • Inner Class  
• Enum

If you want next:

- Only MCQs with answers (Sessions 8-10)
- Tricky PG-DAC MCQs
- Comparison tables
- Full Java MCQ revision pack

Just tell me 

can you give me the mcq on session 1to 10

Yes 

Below are **MCQs covering Session 1 to Session 10, exam-oriented, PG-DAC level**, and **answers given at the end** for fast revision.

## Java MCQs (Session 1 – Session 10)

### ◆ SESSION 1: Introduction to Java

**1. Java is called platform independent because of**

- A. Compiler
- B. JDK
- C. JVM
- D. OS

**2. Java source code file extension is**

- A. .class
- B. .java
- C. .exe
- D. .jar

**3. Which memory area stores objects in JVM?**

- A. Stack
- B. Heap
- C. Method Area
- D. Register

**4. Which component converts bytecode to machine code?**

- A. JDK
- B. JRE
- C. JVM
- D. OS

### ◆ SESSION 2 & 3: Basic Programming

**5. Which of the following is NOT a Java token?**

- A. Keyword
- B. Identifier
- C. Comment
- D. Operator

**6. Which variable has no default value?**

- A. Instance variable
- B. Static variable
- C. Local variable
- D. Global variable

**7. Automatic type conversion is called**

- A. Narrowing
- B. Casting
- C. Widening
- D. Boxing

**8. Which loop executes at least once?**

- A. for
- B. while
- C. do-while
- D. switch

**9. Array index in Java starts from**

- A. 1
- B. -1
- C. 0
- D. Depends on size

## ◆ SESSION 4: OOP Basics

**10. Java is not 100% object-oriented because of**

- A. Classes
- B. Objects
- C. Primitive data types
- D. Interfaces

**11. Which keyword is used to create object?**

- A. create
- B. new
- C. alloc
- D. malloc

**12. Which OOP principle focuses on data hiding?**

- A. Polymorphism
- B. Inheritance
- C. Abstraction
- D. Encapsulation

## ◆ SESSION 5: Static & Reference Types

**13. Static variables belong to**

- A. Object

- B. Method
- C. Class
- D. Constructor

**14. Static members are stored in**

- A. Heap
- B. Stack
- C. Method Area
- D. Register

**15. Reference variable stores**

- A. Value
- B. Object
- C. Address of object
- D. Method

---

**◆ SESSION 6: Constructors & Memory****16. Constructor name must be same as**

- A. Method
- B. Variable
- C. Class
- D. Object

**17. Java follows**

- A. Pass by reference
- B. Pass by pointer
- C. Pass by value
- D. Pass by address

**18. Object is stored in**

- A. Stack
- B. Heap
- C. Register
- D. Method Area

---

**◆ SESSION 7: Inheritance & Polymorphism****19. Java supports which inheritance using classes?**

- A. Multiple
- B. Hybrid
- C. Multilevel
- D. All

**20. Method overloading occurs at**

- A. Runtime
- B. Compile time

- C. Execution time
- D. JVM load time

### **21. Which methods cannot be overridden?**

- A. Abstract
- B. Public
- C. Final
- D. Protected

### **22. super keyword is used to**

- A. Create object
- B. Refer current class
- C. Refer parent class
- D. Destroy object

---

## ◆ **SESSION 8: Abstract & Interface**

### **23. Abstract class can have**

- A. Only abstract methods
- B. Only concrete methods
- C. Both abstract and concrete methods
- D. No methods

### **24. Interface supports**

- A. Single inheritance
- B. Multiple inheritance
- C. Multilevel inheritance
- D. No inheritance

### **25. Upcasting is**

- A. Explicit
- B. Unsafe
- C. Implicit
- D. Not allowed

---

## ◆ **SESSION 9 & 10: Advanced Java**

### **26. Final variable means**

- A. Can be overridden
- B. Can be changed
- C. Cannot be changed
- D. Cannot be accessed

### **27. Functional interface contains**

- A. Zero abstract methods
- B. One abstract method
- C. Two abstract methods
- D. Multiple abstract methods

**28. Lambda expression introduced in**

- A. Java 6
- B. Java 7
- C. Java 8
- D. Java 11

**29. Which inner class has no name?**

- A. Static inner class
- B. Method local class
- C. Anonymous class
- D. Regular inner class

**30. Enum constants are**

- A. Dynamic
- B. User-defined
- C. Fixed
- D. Variable

 **ANSWER KEY**

1. C
2. B
3. B
4. C
5. C
6. C
7. C
8. C
9. C
10. C
11. B
12. D
13. C
14. C
15. C
16. C
17. C
18. B
19. C
20. B

21. C
22. C
23. C
24. B
25. C
26. C
27. B
28. C
29. C
30. C

---

## 🔥 Want more?

I can give you:

- 📋 50 more MCQs (Tricky level)
- 🧠 Previous PG-DAC style MCQs
- ⚡ One-day Java MCQ revision sheet
- 💬 Only wrong-option explanation MCQs

Just tell me 😊

50 more

Perfect 👍

Here are **50 MORE Java MCQs** covering **Session 1 to Session 10, PG-DAC / MCQ-exam oriented**, with **answers at the end**.

Try solving first, then check answers. 🔥

## ✓ Java MCQs – Set 2 (50 Questions)

### ◆ Session 1: Java Basics

#### 1. Who developed Java?

- A. Microsoft
- B. Apple
- C. Sun Microsystems
- D. IBM

**2. Java bytecode is executed by**

- A. Compiler
- B. JVM
- C. JDK
- D. OS

**3. JIT compiler is part of**

- A. JDK
- B. JRE
- C. JVM
- D. OS

**4. Which Java feature provides security?**

- A. Pointers
- B. Bytecode verification
- C. Multiple inheritance
- D. Preprocessor

**5. Which is platform dependent?**

- A. Java language
- B. Bytecode
- C. JVM
- D. JDK

## ◆ Session 2 & 3: Programming Fundamentals

**6. Which is a valid identifier?**

- A. 1num
- B. num-1
- C. \_num
- D. int

**7. Which operator has highest precedence?**

- A. +
- B. \*
- C. =
- D. ++

**8. Which keyword is used to stop loop execution?**

- A. exit
- B. stop
- C. break
- D. end

**9. Which is NOT a logical operator?**

- A. &&
- B. ||
- C. !
- D. &

**10. Which exception occurs when array index is invalid?**

- A. NullPointerException
  - B. ArrayIndexOutOfBoundsException
  - C. ArithmeticException
  - D. ClassCastException
- 

**◆ Session 4: OOP Concepts****11. Object is created at**

- A. Compile time
- B. Runtime
- C. Load time
- D. Design time

**12. Which supports data hiding?**

- A. Abstraction
- B. Polymorphism
- C. Encapsulation
- D. Inheritance

**13. Which keyword is used to inherit a class?**

- A. implements
- B. inherits
- C. extends
- D. super

**14. Which OOP principle improves reusability?**

- A. Encapsulation
- B. Inheritance
- C. Polymorphism
- D. Abstraction

**15. Which cannot be achieved without inheritance?**

- A. Encapsulation
  - B. Abstraction
  - C. Runtime polymorphism
  - D. Compile-time polymorphism
- 

**◆ Session 5: Static & Reference****16. Static block executes**

- A. After object creation
- B. Before main()
- C. After main()
- D. On method call

**17. How many copies of static variable exist?**

- A. One per object

- B. One per class
- C. One per method
- D. Multiple

**18. Reference variable is created in**

- A. Heap
- B. Stack
- C. Method area
- D. Register

**19. Which can be null?**

- A. int
- B. double
- C. char
- D. Reference variable

**20. Static method can access**

- A. Only instance members
- B. Only static members
- C. Both
- D. Neither

## ◆ Session 6: Constructors & Memory

**21. Constructor is used to**

- A. Destroy object
- B. Initialize object
- C. Access data
- D. Call method

**22. If no constructor is defined, Java provides**

- A. Parameterized constructor
- B. Copy constructor
- C. Default constructor
- D. Static constructor

**23. Stack memory stores**

- A. Objects
- B. Methods only
- C. References & local variables
- D. Static variables

**24. Which object becomes eligible for GC?**

- A. Referenced object
- B. Null object
- C. Unreferenced object
- D. Static object

**25. Passing object to method passes**

- A. Object itself

- B. Reference copy
  - C. Memory address directly
  - D. Pointer
- 

## ◆ Session 7: Inheritance & Polymorphism

### 26. Java supports multiple inheritance using

- A. Classes
- B. Abstract classes
- C. Interfaces
- D. Constructors

### 27. Method overriding requires

- A. Same return type only
- B. Same method name
- C. Same signature
- D. All

### 28. Dynamic method dispatch occurs in

- A. Overloading
- B. Overriding
- C. Encapsulation
- D. Abstraction

### 29. Which keyword prevents inheritance?

- A. static
- B. abstract
- C. final
- D. private

### 30. this keyword refers to

- A. Parent class
  - B. Current object
  - C. Static data
  - D. Interface
- 

## ◆ Session 8: Abstract & Interface

### 31. Abstract method must be

- A. private
- B. protected
- C. public
- D. static

### 32. Interface variables are

- A. Instance variables
- B. Local variables

- C. public static final
- D. private

**33. Downcasting is**

- A. Automatic
- B. Implicit
- C. Explicit
- D. Safe

**34. Which causes ClassCastException?**

- A. Upcasting
- B. Valid downcasting
- C. Invalid downcasting
- D. Object creation

**35. Which cannot have constructor?**

- A. Abstract class
- B. Interface
- C. Enum
- D. Class

## ◆ Session 9 & 10: Advanced Java

**36. Final method can be**

- A. Overloaded
- B. Overridden
- C. Hidden
- D. Inherited only

**37. Functional interface annotation is**

- A. @Override
- B. @FunctionalInterface
- C. @Interface
- D. @Lambda

**38. Default methods were introduced in**

- A. Java 6
- B. Java 7
- C. Java 8
- D. Java 11

**39. Lambda expression reduces**

- A. Memory
- B. Security
- C. Code length
- D. Execution time

**40. Anonymous class is used for**

- A. Multiple objects
- B. Single use

- C. Reusability
  - D. Inheritance
- 

## ◆ Mixed Concepts

### 41. Which is not allowed in enum?

- A. Variables
- B. Methods
- C. Constructors
- D. Inheritance

### 42. Enum constructors are

- A. public
- B. protected
- C. private
- D. static

### 43. Static inner class can access

- A. Non-static members
- B. Static members only
- C. All members
- D. Private only

### 44. Which is thread-safe?

- A. Heap
- B. Stack
- C. Method area
- D. Object

### 45. Which is true about String?

- A. Mutable
- B. Final
- C. Static
- D. Abstract

### 46. JVM stands for

- A. Java Virtual Machine
- B. Java Variable Machine
- C. Java Verified Machine
- D. Java Visual Model

### 47. Which exception is unchecked?

- A. IOException
- B. SQLException
- C. ArithmeticException
- D. ClassNotFoundException

### 48. Garbage collection is done by

- A. JDK
- B. JRE

- C. JVM
- D. OS

**49. Method overloading is resolved at**

- A. Runtime
- B. Compile time
- C. Execution time
- D. JVM startup

**50. Which supports backward compatibility?**

- A. Lambda
- B. Default methods
- C. Enum
- D. Final keyword

 **ANSWER KEY (Set 2)**

1. C
2. B
3. C
4. B
5. C
6. C
7. D
8. C
9. D
10. B
11. B
12. C
13. C
14. B
15. C
16. B
17. B
18. B
19. D
20. B
21. B
22. C
23. C

- 24. C
  - 25. B
  - 26. C
  - 27. D
  - 28. B
  - 29. C
  - 30. B
  - 31. C
  - 32. C
  - 33. C
  - 34. C
  - 35. B
  - 36. A
  - 37. B
  - 38. C
  - 39. C
  - 40. B
  - 41. D
  - 42. C
  - 43. B
  - 44. B
  - 45. B
  - 46. A
  - 47. C
  - 48. C
  - 49. B
  - 50. B
- 

### 🔥 Next Options

-  **100-question mock test (timed)**
-  **Only tricky MCQs**
-  **Session-wise MCQ PDFs**
-  **Wrong-option explanation MCQs**

Tell me what you want next 😊