

I will adopt the role of a **senior Java instructor with NPTEL course experience** and generate 50 carefully curated MCQs with answers that reflect the most exam-relevant topics based on your syllabus. I will touch on all 12 weeks, ensuring coverage of OOP, core Java syntax, I/O, multithreading, GUI, networking, JDBC, and packages.

Here's your set of **50 exam-focused MCQs with answers**:

Week 1: OOP and Java Overview

1. **Which of the following is not a feature of Java?**
 - a) Object-Oriented
 - b) Platform Dependent
 - c) Secure
 - d) Robust**Answer:** b) Platform Dependent
 2. **Which paradigm does Java follow?**
 - a) Procedural
 - b) Object-Oriented
 - c) Functional
 - d) Logic**Answer:** b) Object-Oriented
 3. **Which method is the entry point of a Java program?**
 - a) start()
 - b) init()
 - c) main()
 - d) run()**Answer:** c) main()
 4. **Bytecode in Java is executed by:**
 - a) JVM
 - b) JDK
 - c) JRE
 - d) Compiler**Answer:** a) JVM
-

Week 2: Java Programming Elements

5. **Which keyword is used to inherit a class in Java?**
 - a) this
 - b) super
 - c) extends
 - d) implements**Answer:** c) extends
 6. **Which of the following is a reserved word in Java?**
 - a) object
 - b) strictfp
 - c) include
 - d) define**Answer:** b) strictfp
 7. **Which data type is used to store a character in Java?**
 - a) char
 - b) character
 - c) String
 - d) byte**Answer:** a) char
 8. **Which operator is used to compare two values in Java?**
 - a) =
 - b) ==
 - c) !=
 - d) :=**Answer:** b) ==
-

Week 3: Input-Output Handling

9. **Which class is used for reading input from the user?**
 - a) Scanner
 - b) Reader
 - c) InputStream
 - d) BufferedReader**Answer:** a) Scanner
10. **Which package contains Java's I/O classes?**
 - a) java.util
 - b) java.io
 - c) java.input
 - d) java.lang

Answer: b) java.io

11. **Method used to read a line from BufferedReader:**

- a) read()
 - b) readLine()
 - c) next()
 - d) getLine()
- Answer:** b) readLine()

12. **Which stream is used for binary data?**

- a) CharacterStream
 - b) ByteStream
 - c) FileStream
 - d) InputStreamReader
- Answer:** b) ByteStream
-

Week 4: Encapsulation

13. **Encapsulation is implemented using:**

- a) Public methods
- b) Private data members and public methods
- c) Protected data members
- d) Static methods

Answer: b) Private data members and public methods

14. **Getters and setters in Java are used to:**

- a) Execute threads
- b) Access and modify private data
- c) Handle exceptions
- d) Create packages

Answer: b) Access and modify private data

Week 5: Inheritance

15. **Java supports:**

- a) Multiple inheritance through classes
- b) Multiple inheritance through interfaces
- c) Multiple inheritance through packages
- d) Multiple inheritance is not supported

Answer: b) Multiple inheritance through interfaces

16. **Keyword used to call superclass constructor:**

- a) super()
- b) this()
- c) parent()
- d) extends()

Answer: a) super()

17. **Polymorphism allows:**

- a) Same method name, different implementations
- b) Multiple classes in one file
- c) Code duplication
- d) Data hiding

Answer: a) Same method name, different implementations

Week 6: Exception Handling

18. **Superclass of all exceptions in Java:**

- a) Throwable
- b) Exception
- c) Error
- d) RuntimeException

Answer: a) Throwable

19. **Which block is always executed?**

- a) try
- b) catch
- c) finally
- d) throw

Answer: c) finally

20. **Which keyword is used to handle an exception?**

- a) throws
- b) try
- c) catch
- d) handle

Answer: c) catch

21. **Checked exceptions are checked at:**

- a) Compile time

- b) Run time
 - c) Both
 - d) None
- Answer:** a) Compile time
-

Week 7: Multithreaded Programming

22. **Which interface must be implemented to create a thread?**
- a) Runnable
 - b) Threadable
 - c) Executable
 - d) Startable
- Answer:** a) Runnable
23. **Which method starts a thread?**
- a) run()
 - b) start()
 - c) init()
 - d) execute()
- Answer:** b) start()
24. **Thread priority in Java ranges from:**
- a) 1 to 5
 - b) 0 to 10
 - c) 1 to 10
 - d) 0 to 5
- Answer:** c) 1 to 10
25. **Method used to pause a thread:**
- a) wait()
 - b) sleep()
 - c) stop()
 - d) pause()
- Answer:** b) sleep()
-

Week 8: Applets and Servlets

26. **Applets run on:**
- a) Server side
 - b) Client side
 - c) Database
 - d) Compiler
- Answer:** b) Client side
27. **Servlets are used for:**
- a) Client-side scripting
 - b) Server-side programming
 - c) GUI design
 - d) Database design
- Answer:** b) Server-side programming
28. **Method called when an applet is terminated:**
- a) destroy()
 - b) stop()
 - c) end()
 - d) finish()
- Answer:** a) destroy()
29. **Servlets are part of which package?**
- a) java.servlet
 - b) javax.servlet
 - c) org.servlet
 - d) net.servlet
- Answer:** b) javax.servlet
-

Week 9: Java Swing and AWT

30. **Swing components are part of:**
- a) java.awt
 - b) javax.swing
 - c) java.swing
 - d) org.swing
- Answer:** b) javax.swing
31. **Which is a lightweight component?**
- a) AWT
 - b) Swing
 - c) Applet
 - d) Frame
- Answer:** b) Swing

32. **AWT stands for:**
a) Abstract Window Toolkit
b) Advanced Window Toolkit
c) Abstract Widget Toolkit
d) Active Window Toolkit
Answer: a) Abstract Window Toolkit
33. **JButton is used to create:**
a) Label
b) Text field
c) Button
d) Checkbox
Answer: c) Button
-

Week 10: Networking with Java

34. **Java class for creating server socket:**
a) ServerSocket
b) Server
c) SocketServer
d) Socket
Answer: a) ServerSocket
35. **Port numbers range from:**
a) 0 to 65535
b) 1 to 1024
c) 0 to 1024
d) 1025 to 65535
Answer: a) 0 to 65535
36. **Which class represents a URL?**
a) java.net.URL
b) java.url.URL
c) java.net.Link
d) java.link.URL
Answer: a) java.net.URL
37. **Sockets are used for:**
a) File handling
b) Network communication
c) GUI
d) Database connectivity
Answer: b) Network communication
-

Week 11: Java ODBC (JDBC)

38. **JDBC stands for:**
a) Java Database Connectivity
b) Java Data Base Client
c) Java Distributed Base Connectivity
d) Java Database Communication
Answer: a) Java Database Connectivity
39. **Which package contains JDBC classes?**
a) java.sql
b) javax.sql
c) java.db
d) java.database
Answer: a) java.sql
40. **DriverManager is used to:**
a) Load database drivers
b) Create GUI
c) Handle exceptions
d) Manage threads
Answer: a) Load database drivers
41. **Method to execute SQL SELECT query:**
a) executeQuery()
b) executeUpdate()
c) execute()
d) select()
Answer: a) executeQuery()
42. **Which interface represents SQL statements?**
a) Connection
b) Statement
c) ResultSet
d) Driver
Answer: b) Statement
-

Week 12: Interface and Packages

43. **A package in Java is a:**
a) Collection of classes and interfaces
b) Set of objects
c) Function library
d) Thread group
Answer: a) Collection of classes and interfaces
44. **Keyword used to declare an interface:**
a) interface
b) implements
c) package
d) class
Answer: a) interface
45. **An interface can contain:**
a) Method definitions
b) Method declarations only
c) Constructors
d) Static blocks
Answer: b) Method declarations only
46. **Default package in every Java program:**
a) java.default
b) java.lang
c) java.base
d) java.core
Answer: b) java.lang
47. **Which keyword is used to access packages in Java?**
a) package
b) include
c) import
d) using
Answer: c) import
-

Mixed & Advanced

48. **Java's garbage collector runs:**
a) Manually
b) Automatically
c) On compile time
d) On object creation
Answer: b) Automatically
49. **Which is not a Java access modifier?**
a) public
b) private
c) protected
d) internal
Answer: d) internal
50. **Which collection class allows duplicate elements?**
a) Set
b) List
c) Map
d) TreeSet
Answer: b) List
-

This covers **all major exam sections**, including syntax, OOP principles, I/O, GUIs, multithreading, networking, JDBC, and packages/interfaces.

Q: Can you also provide a last-minute revision cheat sheet summarizing all key Java concepts covered in this course?

Of course — adopting the role of a **senior Java course mentor**, I'll now provide a **powerful Last-Minute Revision Cheat Sheet** precisely aligned to your NPTEL "Programming in Java" course.
It will be compact yet complete, sorted by week/topic for rapid glancing.

📄 Java Last-Minute Revision Cheat Sheet (NPTEL)

📄 Week 1: OOP & Java Basics

- **OOP Concepts:** Encapsulation, Inheritance, Polymorphism, Abstraction.
 - **Java is:** Platform-independent (because of JVM), Object-Oriented, Robust, Secure.
 - **main() method:**

```
public static void main(String[] args)
```
 - **Compilation:**
.java → (javac) → .class (bytecode) → (JVM) → machine code.
-

⚙️ Week 2: Java Programming Elements

- **Primitive Types:** byte, short, int, long, float, double, char, boolean.
 - **Operators:** Arithmetic (+ - * / %), Relational (== != > <), Logical (&& || !), Assignment (= += -=).
 - **Control Statements:** if, if-else, switch, for, while, do-while.
-

📁 Week 3: Input-Output Handling

- **Scanner for input:**

```
Scanner sc = new Scanner(System.in);
```
 - **BufferedReader for input:**

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```
 - **I/O Streams:**
 - **Byte Streams:** InputStream, OutputStream.
 - **Character Streams:** Reader, Writer.
-

📁 Week 4: Encapsulation

- **Encapsulation = Private Data + Public Getters/Setters.**
- Example:

```
private int age;
public int getAge() { return age; }
public void setAge(int a) { age = a; }
```

📁 Week 5: Inheritance

- **Single inheritance only (class-wise).**
 - **Syntax:** class B extends A { }
 - **super keyword:** Calls parent class constructor or method.
 - **Method Overriding:** Same signature, different class (Inheritance).
-

📁 Week 6: Exception Handling

- **try-catch-finally** block.
 - **Checked Exceptions:** Must be caught (e.g., IOException).
 - **Unchecked Exceptions:** Runtime errors (e.g., NullPointerException).
 - **throw vs throws:**
 - **throw** : manually throw exception.
 - **throws** : declare exception possibility.
-

📁 Week 7: Multithreading

- **Two ways to create threads:**
 - Extending Thread class
 - Implementing Runnable interface.
 - **Thread methods:**
start(), run(), sleep(ms), join(), yield().
 - **Thread Life Cycle:** New → Runnable → Running → Blocked/Waiting → Dead.
-

📁 Week 8: Applets and Servlets

- **Applet methods:** `init()`, `start()`, `stop()`, `destroy()`.
- **Servlet:** Java program on the server-side for web apps.
- **Servlet lifecycle:** `init()` → `service()` → `destroy()`.

📅 Week 9: Java Swing and AWT

- **AWT (heavyweight)** vs **Swing (lightweight, flexible)**.
- **Common Swing Components:** `JFrame`, `JButton`, `JLabel`, `JTextField`, `JPanel`.
- **Event Handling:** `ActionListener`, `MouseListener`, etc.

📅 Week 10: Networking with Java

- **Socket Programming:**
 - Client: `Socket`
 - Server: `ServerSocket`
- **URL Handling:**
`URL url = new URL("http://example.com");`
- **Important Classes:** `Socket`, `ServerSocket`, `DatagramSocket`.

📅 Week 11: Java Database Connectivity (JDBC)

- **JDBC Steps:**
 1. Load Driver → `Class.forName(driverName)`
 2. Connect to DB → `DriverManager.getConnection(url, user, pass)`
 3. Create Statement → `createStatement()`
 4. Execute Query → `executeQuery()` / `executeUpdate()`
 5. Close Connection.
- **Interfaces:** `Connection`, `Statement`, `PreparedStatement`, `ResultSet`.

📅 Week 12: Interfaces and Packages

- **Interfaces:** Only abstract methods (Java 7); default & static methods allowed from Java 8.
- **Syntax:**

```
interface MyInterface {  
    void myMethod();  
}
```

- **Packages:**
 - Group of related classes/interfaces.
 - Import with `import package.className;`
- **Default package:** `java.lang` (imported automatically).

📅 Quick Syntax Snippets

Concept	Syntax Example
Class Definition	<code>class ClassName {</code>
Inheritance	<code>class Child extends Parent {</code>
Interface Implementation	<code>class A implements InterfaceName {</code>
Exception Handling	<code>try { } catch(Exception e) { }</code>
Thread Creation	<code>Thread t = new Thread(obj); t.start();</code>
Package Declaration	<code>package mypack;</code>

☒ Common Java Command Line

```
javac Program.java // compile
java Program       // run
```

☒ Very Last-Minute 5 Points to Remember

- Always close JDBC Connections.
- Use synchronized keyword for thread safety.
- Constructors do not have a return type.
- finalize() method for garbage collection.
- Java does NOT support operator overloading.

Would you also like a **visual mind map (flowchart style)** summarizing this for even faster memory retention? ☒
(It would be perfect to glance at 1 hour before your exam.)

Q: Can you create a "common mistakes" checklist to avoid while answering Java exam questions?

Absolutely — here's a **targeted checklist** designed specifically for Java exams like NPTEL's. This will help you avoid common traps and errors students make during the test.

☒ Java Exam: Common Mistakes Checklist (Avoid These!)

☒ Basics & Syntax

- ☐ Forgetting **semicolon (;)** at the end of statements.
- ☐ Miswriting the `main` method signature:
☒ `public static void main(String[] args)`
☒ `public void main()` or `static public void main()`
- ☐ Using **capital letters incorrectly**: Java is **case-sensitive!** (`System` vs `system`, `String` vs `string`).
- ☐ Confusing `==` (comparison) with `=` (assignment).

☒ OOP & Inheritance

- ☐ Forgetting to use `super()` to call parent class constructor.
- ☐ Thinking Java supports **multiple inheritance with classes** (it doesn't — use interfaces).
- ☐ Overriding methods without `@Override` annotation (not compulsory but prevents mistakes).
- ☐ Not distinguishing between **method overloading** (same class) and **method overriding** (different classes).

☒ Exception Handling

- ☐ Missing `try-catch` block around code that throws **checked exceptions** (e.g., `IOException`).
- ☐ Writing `catch(Exception e)` without handling specific exceptions first.
- ☐ Forgetting **finally** is **always executed**, even after `return`.

☒ Multithreading

- ☐ Confusing `run()` vs `start()`:
☒ Call `start()` to start a new thread,
☒ Calling `run()` just executes in current thread.
- ☐ Forgetting to synchronize shared data between threads (use `synchronized`).

☒ GUI (Swing/AWT)

- ☐ Mixing **AWT** and **Swing** components (can cause issues).
 - ☐ Not using the **Event Dispatch Thread (EDT)** to update GUI components (`SwingUtilities.invokeLater()`).
-

☒ Networking

- ☐ Not closing **sockets** and streams (may cause resource leaks).
 - ☐ Using incorrect port numbers or missing `try-catch` block in networking code.
-

☒ JDBC (Database)

- ☐ Forgetting to close **Connection, Statement, and ResultSet**.
 - ☐ Using **Statement** instead of **PreparedStatement** (less secure).
 - ☐ Not handling `SQLException` properly.
-

☒ Interfaces & Packages

- ☐ Forgetting that **interfaces can't have constructors**.
 - ☐ Using access modifiers incorrectly in interfaces:
 - All methods are **implicitly public and abstract** (Java 7).
 - ☐ Forgetting to **import** packages/classes before using them.
-

⚠ Conceptual Mistakes

- ☐ Thinking **Java supports operator overloading** (it does not).
 - ☐ Believing **everything** in Java is passed by reference —
 - ☒ Primitives = pass by value
 - ☒ Objects = reference is passed by value.
 - ☐ Confusing **checked** and **unchecked** exceptions.
 - ☐ Thinking **final** = **finally** = **finalize()** (they are very different!).
-

☒ During MCQs

- ☐ Not reading all answer options carefully (some have subtle differences).
 - ☐ Ignoring keywords like **"ALWAYS"**, **"NEVER"** in the question.
 - ☐ Missing small syntax errors hidden in code-based MCQs.
 - ☐ Assuming default access without checking package/class relationships.
-

☒ Before Submission

- ☐ Skipping code **compilation mentally** — always "simulate" running the code in your head.
 - ☐ Not double-checking exception-prone areas (file I/O, networking, database).
-

Would you also like me to give you a **"Last 10-Minute Mental Exam Routine"** — a checklist for what exactly to do in the final 10 minutes before you start the exam? ☒
(Super useful for calm + accuracy boost.)

Q: Can you list out **20 tricky syntax traps** in Java that often show up in NPTEL-style MCQs?

Excellent request — here are **20 tricky Java syntax traps** carefully selected to match **NPTEL exam styles**. Many of these are designed to confuse even students who know the concepts but miss small details under pressure.

⚠ 20 Tricky Java Syntax Traps (Common in NPTEL MCQs)

1. ☒ Wrong main method signature

```
static public void main(String args[]) // ❌ valid
public void main(String[] args) // ❌ invalid (must be static)
```

2. ❌ Misusing == VS .equals()

```
String s1 = "abc";
String s2 = new String("abc");
System.out.println(s1 == s2);    // false (reference check)
System.out.println(s1.equals(s2)); // true (content check)
```

3. ❌ Using non-final variables in switch

```
int x = 10;
switch (x) { // ❌ valid
}
```

But:

```
double d = 10.5;
// switch(d) // ❌ invalid: only int, byte, short, char, String allowed
```

4. ❌ Forgetting default constructors when parameterized constructor is present

```
class A {
    A(int x) {}
}
// A obj = new A(); // ❌ compile error (no default constructor)
```

5. ❌ Overriding with wrong method signature

```
class A {
    void show(int x) {}
}
class B extends A {
    void show(double x) {} // ❌ this is overloading, not overriding
}
```

6. ❌ final, finally, finalize() confusion

- **final**: variable/method/class can't change.
 - **finally**: block always executes.
 - **finalize()**: called before GC (deprecated).
-

7. ❌ Multiple catch blocks — order matters

```
try { }
catch (Exception e) { } // ❌ must come after specific exceptions
catch (ArithmeticException e) { } // ❌ compile error (unreachable)
```

8. ❌ Using `throw` without an object

```
throw new ArithmeticException(); // ❌ valid
// throw ArithmeticException; // ❌ invalid
```

9. ❌ Thread `run()` vs `start()` confusion

```
t.start(); // ❌ starts new thread
t.run();   // ❌ just calls method in current thread
```

10. ❌ Array indexes starting at 1 (wrong)

```
int[] arr = new int[5];
arr[0] = 10; // ❌ correct
arr[5] = 20; // ❌ ArrayIndexOutOfBoundsException
```

11. ❌ Access modifier trap: `private` members not visible to subclass

```
class A {
    private int x;
}
class B extends A {
    void show() {
        // System.out.println(x); // ❌ compile error
    }
}
```

12. ❌ `static` methods can't access instance variables

```
class A {
    int x;
    static void show() {
        // System.out.println(x); // ❌ compile error
    }
}
```

13. ❌ Assigning incompatible types

```
int i = 10;
double d = i; // ❌ widening
int j = d;    // ❌ compile error (needs cast)
```

14. ❌ interface methods are public by default

```
interface A {
    void show(); // ❌ implicitly public abstract
}
```

If implemented as:

```
class B implements A {  
    void show() {} // ❌ compile error (must be public)  
}
```

15. ❌ `break / continue` outside loops

```
if (true) {  
    // break; // ❌ compile error  
}
```

16. ❌ `null` can't be assigned to primitive types

```
int x = null; // ❌ compile error  
Integer y = null; // ✅ works (autoboxing)
```

17. ❌ Forgetting to close JDBC objects

```
Connection con = DriverManager.getConnection(...);  
// con.close(); // ❌ must close
```

18. ❌ Wrong package import

```
// import java.util.scanner; // ❌ wrong (case sensitive)  
import java.util.Scanner; // ✅ correct
```

19. ❌ Applet lifecycle methods signature mismatch

```
public void init() { } // ✅ correct  
// public void Init() { } // ❌ wrong (Java is case-sensitive)
```

20. ❌ Misusing `super()` in constructors

```
class A {  
    A(int x) {}  
}  
class B extends A {  
    B() {  
        // super(); // ❌ compile error (no default constructor)  
        super(10); // ✅ correct  
    }  
}
```

These **20 traps** are real MCQ bait in NPTEL, GATE, and other technical exams! Many are meant to check your **attention to syntax, not just concepts**. Memorizing these today will massively improve your score tomorrow.
