# OOP with Java Important Questions PDF

The most important OOP with Java questions are the features of Java and the JVM, OOP principles, constructors and the this and static keywords, inheritance and method overriding, abstract classes versus interfaces, packages and access modifiers, exception handling with custom exceptions, multithreading and synchronisation, string handling, the Collections framework (ArrayList, HashMap, HashSet), file I/O, and newer topics such as lambda expressions and streams.

## Key takeaways

- Write a short, correct program for every long answer where it fits. Java papers reward working code.
- Abstract class versus interface, overloading versus overriding, and exception handling are asked very often.
- Multithreading questions usually ask for both theory (thread life cycle) and a synchronised program.
- These are practice questions, not past papers; check your own syllabus and official previous papers.

## About these important questions

This page lists the most important Object Oriented Programming with Java questions for B.Tech CSE and IT exams, arranged unit by unit. The units follow the standard syllabus that most Indian technical universities share, so the order may differ slightly from yours.

**Please note:** these are practice questions based on common syllabus topics. They are not verified past papers, and we do not claim that any question will appear in your exam. Check your own university syllabus first, then solve the official previous year papers. AKTU students can download them from the AKTU old question paper archive.

## Unit 1: Introduction to Java and OOP

Covers Java features, the JDK, JRE and JVM, data types, operators, control statements and arrays.

**Short questions (2 marks)**

- Why is Java platform independent?
- Differentiate between JDK, JRE and JVM.
- What is bytecode?

- List the four pillars of OOP.

**Long questions**

1. Explain the features of Java and the architecture of the JVM with a diagram.
2. Explain encapsulation, abstraction, inheritance and polymorphism with Java examples.
3. Explain type casting, and widening versus narrowing conversion.
4. Write a program to find the largest element in an array and to multiply two matrices.
5. Explain command-line arguments with a program.

## Unit 2: Classes, Inheritance, Interfaces and Packages

Covers classes and objects, constructors, inheritance, polymorphism, interfaces and packages.

**Short questions (2 marks)**

- Differentiate between method overloading and method overriding.
- What is the use of the super keyword?
- What is a final class?
- Why does Java not support multiple inheritance with classes?

**Long questions**

1. Explain constructors and constructor overloading with a program, and the role of the this keyword.
2. Explain the types of inheritance with programs.
3. Compare abstract classes and interfaces, and write a program that uses both.
4. Explain dynamic method dispatch (runtime polymorphism) with a program.
5. Explain packages, how to create and import one, and the four access modifiers in a table.
6. Explain the static keyword: static variables, methods and blocks.

## Unit 3: Exception Handling and Multithreading

Covers exceptions, the thread life cycle, synchronisation and inter-thread communication.

**Short questions (2 marks)**

- Differentiate between checked and unchecked exceptions.
- What is the difference between throw and throws?
- Differentiate between a process and a thread.

**Long questions**

1. Explain try, catch, finally, throw and throws with a program, including a custom exception.
2. Explain the exception hierarchy in Java with a diagram.
3. Explain the thread life cycle with a state diagram.
4. Create threads by extending Thread and by implementing Runnable, and compare the two ways.
5. Explain synchronisation with a synchronised method or block, using a bank account example.

6. Explain inter-thread communication using wait(), notify() and notifyAll() with producer-consumer.

## Unit 4: Strings, I/O and Java New Features

Covers String, StringBuffer and StringBuilder, file handling, and newer language features. Universities differ on which new features they include.

**Short questions (2 marks)**

- Why are strings immutable in Java?
- Differentiate between StringBuffer and StringBuilder.
- What is a functional interface?

**Long questions**

1. Explain important String methods with a program.
2. Write a program to read from and write to a file using FileReader, FileWriter or buffered streams.
3. Explain byte streams and character streams.
4. Explain lambda expressions and method references with examples.
5. Explain the Stream API with filter, map and collect examples.
6. Explain default and static methods in interfaces, and the try-with-resources statement.

## Unit 5: Collections Framework and JDBC

Covers the Collections framework, generics and database connectivity. Some syllabi add Spring or servlets here.

**Short questions (2 marks)**

- Differentiate between ArrayList and LinkedList.
- Differentiate between HashMap and Hashtable.
- What is an Iterator?

**Long questions**

1. Explain the hierarchy of the Collections framework with a diagram.
2. Explain List, Set and Map with programs using ArrayList, HashSet and HashMap.
3. Compare HashSet, LinkedHashSet and TreeSet.
4. Explain Comparable and Comparator with a sorting program.
5. Explain generics and why they improve type safety.
6. Explain the steps to connect to a database using JDBC, with a program that runs a SELECT query.

## Sample answer outline: Abstract class versus interface

This question is asked in many forms. A full answer has four parts:

1. **Define both.** An abstract class is declared with the abstract keyword and may have abstract and concrete methods, constructors and instance variables. An interface defines a contract; its

variables are public, static and final, and since Java 8 it can also have default and static methods.

2. **Give a comparison table.** Rows for keyword (abstract versus interface), methods, variables, constructors, inheritance (a class extends one abstract class but can implement many interfaces) and access modifiers.
3. **Write a short program.** For example, an abstract class Shape with an abstract method area() and a concrete method display(), a Drawable interface with draw(), and a Circle class that extends Shape and implements Drawable.
4. **Say when to use each.** Use an abstract class when related classes share code and state. Use an interface when unrelated classes must share a capability, or to achieve multiple inheritance of type.

Keep the program short and make sure it compiles. Comment the key lines.

## Marks weightage and how to write answers

A common pattern, used by AKTU and many other universities, has a compulsory short-answer section that covers every unit, followed by long-answer questions with a choice inside each unit. Marks per question change between universities and years, so confirm the scheme on a recent official paper.

- **Short answers (about 2 marks):** a crisp definition, one example or diagram, and nothing more. Three to five lines is enough.
- **Long answers (7 to 10 marks):** start with a definition, add a labelled diagram or table, explain the steps, and finish with an example, advantages or a comparison.
- Write programs with correct class names, a main method and sample output. Missing semicolons and braces cost marks.
- When comparing two concepts, use a table, then a small code example.

## A 7-day revision plan for OOP with Java

1. **Days 1 to 5:** one unit a day. Read your notes, then write the long answers for that unit from memory and check them against your notes.
2. **Day 6:** write from memory the programs for custom exceptions, synchronised threads, file I/O, a HashMap example and JDBC.
3. **Day 7:** solve one full official previous year paper in three hours, then check which units cost you time.

Keep a single sheet of formulas, definitions and diagrams for the last hour before the exam.

## Frequently asked questions

### Which OOP with Java topics are most important?

OOP principles, constructors, inheritance and polymorphism, abstract classes and interfaces, exception handling, multithreading, string handling, collections and JDBC are asked most often.

### Do I need to write full programs in a Java exam?

For long answers, yes where the question asks for a program. Write a complete, short program with a main method. For theory questions, a small code snippet is enough.

### Is Java still useful for jobs?

Yes. Java is widely used for back-end development, Android and enterprise software, and OOP concepts carry over to most other languages.

### Are these OOP with Java questions from previous year papers?

No. They are practice questions built from topics that appear in most B.Tech Object Oriented Programming with Java syllabi. Use them with your own university's official previous papers, for example the AKTU archive at https://aktu.ac.in/old-question-paper.html.

### Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

## Sources and further reading

- AKTU: old question papers (official archive)
- Oracle: The Java Tutorials
- NPTEL: online courses from IITs (search for Programming in Java)

## About this PDF

These are practice questions for revision, prepared by the Uncodemy trainer team from common B.Tech syllabus topics. They are not past papers, a certificate or a promise of marks. Spot an error? Write to info@uncodemy.com and the team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550