The 80 Top Java Interview Questions and Answers

Question 1: What is Java?

Answer: Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

Question 2: Explain the JDK, JRE, and JVM.

Answer:

- **JVM (Java Virtual Machine):** The JVM is an engine that provides a runtime environment to drive the Java Code or applications. It converts Java bytecode into machine language.
- **JRE (Java Runtime Environment):** JRE is a part of software that is designed to run other software. It contains the set of libraries + other files that JVM uses at runtime.
- **JDK (Java Development Kit):** The JDK is a software development environment used for developing Java applications and applets. It physically exists. It contains JRE + development tools.

Question 3: What are variables in Java?

Answer: Variables are containers for storing data values. In Java, each variable must be declared with a data type that designates the type and quantity of value it can hold. Java is a statically typed language, meaning variables must be defined before they are used.

Question 4: What is typecasting in Java?

Answer: Typecasting is the process of converting a variable from one type to another. In Java, there are two types of casting:

- **Widening Casting (Implicit):** automatic type conversion from a smaller to a larger type
- **Narrowing Casting (Explicit):** needs explicit conversion to convert a larger type to a smaller type

Question 5: How do you declare an array in Java?

Answer: An array is a container object that holds a fixed number of values of a single type. The declaration of an array in Java is as follows:

```
1
```

```
2
```

```
int[] myIntArray = new int[10]; // declares an array of integers
```

```
String[] myStringArray = new String[50]; // declares an array of strings
```

Question 6: Explain the `main` method in Java.

Answer: The `main` method is the entry point for any Java program. It must be public, static, return no value (void), and accept a String array as a parameter. It signature is:

```
1
```

```
2
```

```
3
```

```
public static void main(String[] args) {
```

```
    // code to be executed
```

```
}
```

Question 7: What are literals in Java?

Answer: Literals refer to the fixed values assigned to variables in Java. For example, `100`, `-90`, `3.14F`, `'A'`, and `"Hello"` are all literals.

Question 8: What is a constructor in Java?

Answer: A constructor in Java is a block of code similar to a method that's called when an instance of an object is created. Unlike methods, constructors have no explicit return type and have the same name as the class itself.

Question 9: Explain method overloading in Java.

Answer: Method overloading is a feature that allows a class to have more than one method having the same name, if their parameter lists are different. It is related to compile-time (or static) polymorphism.

Question 10: What is a package in Java?

Answer: A package in Java is a namespace that organizes a set of related classes and interfaces. Conceptually you can think of packages as being similar to different folders on your computer.

Question 11: What is Object-Oriented Programming?

Answer: Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data in the form of fields (often known as attributes or properties) and code in the form of procedures (often known as methods).

Question 12: What are the main principles of OOP?

Answer:

- **Encapsulation:** The binding (or wrapping) of data and methods that operate on the data into a single unit called a ‘class’. It also means hiding data (i.e., private variables) from direct access.
- **Abstraction:** Hiding the complex implementation details of an operation while exposing a simple interface.
- **Inheritance:** Allows a new class to inherit properties and methods of an existing class.
- **Polymorphism:** The ability of different classes to provide a unique interface by exposing a method that can behave differently.

Question 13: What is inheritance?

Answer: Inheritance in Java is a mechanism where one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object-Oriented programming systems).

Question 14: What is an interface?

Answer: An interface in Java is a reference type, similar to a class, that can contain only constants, method signatures, default methods, static methods, and nested types. Interfaces cannot contain instance fields. The methods in interfaces are abstract by default.

Question 15: Explain the difference between abstract classes and interfaces.

Answer:

- **Abstract Class:** Can have both abstract and non-abstract methods. Abstract classes are used to provide a base for subclasses to extend and implement the abstract methods.
- **Interface:** Typically contains abstract methods only. Starting from Java 8, it can also contain default and static methods. Interfaces are used to implement abstraction.

Question 16: What is polymorphism?

Answer: Polymorphism in Java is the ability of an object to take on many forms. Most commonly, it is when a parent class reference is used to refer to a child class object.

Question 17: Explain method overriding.

Answer: Method overriding, in object-oriented programming, is a language feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its superclasses or parent classes. The method that is overridden is called an overridden method.

Question 18: What is the "super" keyword?

Answer: The `super` keyword in Java is a reference variable that is used to refer to parent class objects. The keyword can be used to call superclass methods and to access the superclass constructor.

Question 19: What are getters and setters in Java?

Answer: Getters and setters are methods that get and set the value of a private variable. For example:

```
1
2
3
4
5
6
7
8
9
```

```
public class Data {
    private int num;
    public int getNum() {
        return num;
    }
    public void setNum(int num) {
        this.num = num;
    }
}
```

Question 20: What is static in Java?

Answer: The keyword `static` means that the particular member belongs to a type itself, rather than to an instance of that type. This means that only one instance of that static member is created which is shared across all instances of the class.

Question 21: What is exception handling?

Answer: Exception handling is a mechanism to handle runtime errors such as `ClassNotFoundException`, `IOException`, `SQLException`, `RemoteException`, etc. The core advantage of exception handling is to maintain the normal flow of the application.

Question 22: What is a try-catch block?

Answer: `try` is the start of the block, and `catch` is used to handle the exception. It must be followed by either finally or another catch block.

Question 23: What is the finally block?

Answer: The `finally` block in Java is used to place important code such as clean-up code, e.g., closing the file, database connections, etc. The finally block executes whether an exception is handled or not.

Question 24: What is an exception?

Answer: An exception is a problem that arises during the execution of a program. Exceptions are caught and handled in Java with try-catch blocks.

Question 25: What are checked exceptions?

Answer: Checked exceptions are checked at compile-time. It means if a method is throwing a checked exception, then it should handle the exception using the `try-catch` block, or it should declare the exception using the `throws` keyword, e.g., IOException, SQLException, etc.

Question 26: What are unchecked exceptions?

Answer: Unchecked exceptions are not checked at compiled time. It means if your program is throwing an unchecked exception, and even if you didn't handle/declare that exception, the program won't give a compilation error. Most of the time, these are programming errors, e.g., Logic errors or improper use of an API. Examples are `ArithmeticException`, `NullPointerException`, `ArrayIndexOutOfBoundsException`, etc.

Question 27: What is the throws keyword?

Answer: The `throws` keyword is used to declare an exception. It gives information to the programmer that there may be an exception, so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.

Question 28: What is the difference between throw and throws?

Answer: `throw` keyword is used to explicitly throw an exception. `throws` is used to declare an exception. Checked exceptions cannot be propagated with `throws`.

Question 29: What is throw used for?

Answer: The `throw` keyword is used within a method. It is used to throw an exception explicitly.

Question 30: Explain custom exceptions.

Answer: Sometimes, it is necessary to create a custom exception. This is done by extending the Exception class.

Question 31: What is the String API?

Answer: The String API is a set of classes and methods that operate on strings, including manipulating characters, comparing strings, searching strings, extracting substrings, and creating copies of strings with alterations.

Question 32: Explain "String", "StringBuilder", and "StringBuffer".

Answer:

- **`String`:** Immutable sequence of characters.
- **`StringBuilder`:** Mutable sequence of characters. Not thread-safe.
- **`StringBuffer`:** Mutable sequence of characters. Thread-safe.

Question 33: What is an immutable object?

Answer: An immutable object is an object whose state cannot be modified after it is created. Strings are a good example of immutable objects.

Question 34: What are wrapper classes?

Answer: Wrapper classes convert the Java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they "wrap" the primitive data type into an object of that class. Examples include `Integer`, `Character`, `Double` etc.

Question 35: Explain autoboxing and unboxing.

Answer: Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an `int` to an `Integer`, a `double` to a `Double`, etc. Unboxing is the reverse process, where the object is converted back to a primitive type.

Question 36: What is the "Collections" API?

Answer: The Collections API is a set of classes and interfaces that support operations on collections of objects. These classes and interfaces are grouped under java.util package. The collections framework provides both interfaces that define various collections and classes that implement them.

Question 37: What are generics?

Answer: Generics enable types (classes and interfaces) to be parameters when defining classes, interfaces, and methods. Much like the more familiar formal parameters used in method declarations, type parameters provide a way for you to reuse the same code with different inputs. The inputs to formal parameters are values, while the inputs to type parameters are types.

Question 38: What is the "Iterator" interface?

Answer: The `Iterator` interface provides methods to iterate over any Collection. It works similarly to a "looping" mechanism, and it allows you to traverse through a collection and remove elements from the collection selectively if desired.

Question 39: Explain the difference between "Iterator" and "ListIterator".

Answer:

- **`Iterator`:** Enables you to traverse through a collection in the forward direction only, for obtaining or removing elements.
- **`ListIterator`:** Extends `Iterator` to allow bidirectional traversal of the list and also to modify elements.

Question 40: What is a "Map"?

Answer: A `Map` is an object that maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value. Examples include `HashMap, TreeMap,` and `LinkedHashMap.`

Question 41: What is concurrency?

Answer: Concurrency is the ability to run several programs or several parts of a program in parallel. It helps to perform many tasks in a shorter period of time.

Question 42: What is a thread?

Answer: A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.

Question 43: What is the difference between a process and a thread?

Answer:

- **Process:** A program in execution. Independent set of variables, own execution stack.
- **Thread:** Subset of Process. Threads share the same process's variables and execution stack.

Question 44: What is multithreading?

Answer: Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU.

Question 45: What is the "synchronized" keyword?

Answer: The `synchronized` keyword is used to control the access of multiple threads to any shared resource. It can be used to define a method or a block of code that can only be accessed by one thread at a time.

Question 46: What is deadlock?

Answer: Deadlock is a part of multithreading. It occurs when two or more threads get blocked forever, waiting for each other. Deadlock occurs due to improper thread synchronization.

Question 47: What are the states of a thread?

Answer: A thread can be in one of the following states:

- **New:** The thread is in a new state if you create an instance of the Thread class but before the invocation of the start() method.
- **Runnable:** The thread is ready to run, but it's not running.
- **Blocked:** The thread is waiting for a monitor lock.
- **Waiting:** The thread is waiting indefinitely for another thread to perform a particular action.
- **Timed Waiting:** The thread is waiting for another thread to perform a specific action for up to a specified waiting time.

- **Terminated:** A thread that has exited is in this state.

Question 48: What is the "volatile" keyword?

Answer: The `volatile` keyword is used to indicate that a variable's value will be modified by different threads. Declaring a volatile Java variable means:

- The value of this variable will never be cached thread-locally: all reads and writes will go straight to "main memory";
- Access to the variable acts as though it is enclosed in a synchronized block, synchronized on itself.

Question 49: What is thread safety?

Answer: Thread safety means that a method or class instance can be used by multiple threads at the same time without any problem. An example of thread safety is a synchronized method.

Question 50: How do you create a thread in Java?

Answer: There are two ways to create a thread in Java:

By extending the Thread class:

```
1
2
3
4
5
```

```
public class MyThread extends Thread {
    public void run() {
        System.out.println("Thread running");
    }
}
```

By implementing the Runnable interface:

```
1
2
3
4
5
```

```
public class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Runnable running");
    }
}
```

Question 51: What is the Stream API in Java?

Answer: The Stream API in Java provides a new abstraction called Stream, which allows you to process data in a declarative way. It supports operations like map, filter, limit, reduce, find, match, and sort, on collections of objects.

Question 52: What are the benefits of using Streams?

Answer: Streams can make the code more concise and readable. They can simplify code to perform bulk operations sequentially or parallelly. Streams don't store data and, instead, operate on the source data structures (e.g., collections) directly.

Question 53: How do you obtain a Stream from a List?

Answer:

```
1
```

```
2
```

```
List<String> myList = Arrays.asList("a1", "a2", "b1", "c2", "c1");
```

```
Stream<String> myStream = myList.stream();
```

Question 54: What is the difference between 'map' and 'flatMap' in Streams?

Answer:

- **`map`:** It transforms each element in the stream using the given function. It is a **one-to-one** mapping.
- **`flatMap`:** It helps in converting one type of `stream` element into another type and flattens the `Stream` of `Streams` into a `Stream`.

Question 55: What is the "filter" method in a "Stream"?

Answer: The `filter` method is used to evaluate each element in a stream using a predicate. If the predicate evaluates to `true`, the element is included in the resulting stream.

Question 56: How do you sort a "Stream"?

Answer:

```
1

2

3

4
```

```
List<String> myList = Arrays.asList("a1", "a2", "b1", "c2", "c1");

List<String> sortedList = myList.stream()

    .sorted()

    .collect(Collectors.toList());
```

Question 57: What are terminal operations on "Streams"?

Answer: Terminal operations produce a result from a stream pipeline. Terminal operations include operations like `forEach`, `reduce`, `collect`, and `sum`.

Question 58: What is the "collect" method in "Streams"?

Answer: The `collect` method is a terminal operation that transforms the elements of a stream into a different kind of result, e.g., a `List`, a `Map`, or even an `Integer`.

Question 59: What is the "forEach" operation in "Streams"?

Answer: The `forEach` operation is used to iterate over each element of the stream. The `forEach` operation is a terminal operation which means it consumes the stream and can't be used afterwards.

Question 60: Explain the "reduce" operation in "Streams".

Answer: The `reduce` operation combines all elements of the stream into a single result by applying a binary operator. This operation takes two parameters: an initial value, and a binary operator function.

Question 61: What is JDBC?

Answer: JDBC (Java Database Connectivity) is an API that enables Java programs to execute SQL statements. This allows Java applications to interact with any SQL-compliant database.

Question 62: What are the core components of JDBC?

Answer: The core components of JDBC include DriverManager, Driver, Connection, Statement, ResultSet, and SQLException.

Question 63: How do you connect to a database in Java?

Answer:

```
Connection conn = DriverManager.getConnection("jdbc:subprotocol:subname", "user",
"password");
```

Question 64: What are "Statement" and "PreparedStatement"?

Answer:

- **Statement:** Used to execute a simple SQL query with no parameters.
- **PreparedStatement:** Used for executing SQL statements multiple times or when you need to bind parameters to the query.

Question 65: How do you execute a query in JDBC?

Answer:

```
1
```

```
2
```

```
Statement stmt = conn.createStatement();
```

```
ResultSet rs = stmt.executeQuery("SELECT * FROM myTable");
```

Question 66: What is "ResultSet" in JDBC?

Answer: `ResultSet` is a table of data representing a database result set, which is generated by executing a statement that queries the database.

Question 67: What do you mean by batch processing in JDBC?

Answer: Batch processing in JDBC is used to execute multiple SQL statements as a single batch, which reduces the number of communication rounds between the application and the database server.

Question 68: Explain the types of JDBC drivers.

Answer: There are four types of JDBC drivers:

- Type 1: JDBC-ODBC bridge
- Type 2: Native-API/partly Java driver
- Type 3: Net-protocol/all-Java driver
- Type 4: Native-protocol/all-Java driver

Question 69: How do you handle SQL exceptions?

Answer:

```
1

2

3

4

5
```

```
try {

    // code that may throw SQLException

} catch (SQLException ex) {

    // handle exception

}
```

Question 70: What is Connection Pooling?

Answer: Connection pooling is a technique used to improve performance in applications that need to make calls to a database by reusing the connections instead of creating a new one each time.

Question 71: What is the "List" interface in Java?

Answer: The List interface is part of the Java Collections Framework and it represents an ordered collection (also known as a sequence). The user can access elements by their integer index (position in the list), and search for elements in the list.

Question 72: How do you iterate over a "List"?

Answer:

```
1

2

3

4
```

```
List<String> list = Arrays.asList("apple", "banana", "cherry");

for(String fruit : list) {

    System.out.println(fruit);

}
```

Question 73: What is the difference between "ArrayList" and "LinkedList"?

Answer:

- **ArrayList:** Resizable-array implementation of the List interface. Best for storing and accessing data.
- **LinkedList:** Doubly-linked list implementation of the List interface. Better for operations that require frequent addition and removal of elements from any part of the list.

Question 74: How do you remove elements from a List?

Answer: Elements can be removed from a list using the `remove()` method. It can be called using an element or using an index:

```
1
```

```
2
```

```
list.remove("apple");  // removes "apple" from list
```

```
list.remove(0);       // removes the first element
```

Question 75: What are "Vector" and "Stack" classes?

Answer:

- **Vector:** Similar to `ArrayList`, but it is synchronized.
- **Stack:** Extends `Vector` with five operations that allow a vector to be treated as a stack.

Question 76: How do you convert a List to an array?

Answer:

```
String[] array = list.toArray(new String[list.size()]);
```

Question 77: Compare Iterator and "ListIterator".

Answer:

- **Iterator:** Can traverse the list in the forward direction only.
- **ListIterator:** Can traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

Question 78: What is the 'subList' method in List?

Answer: The `subList` method creates a view of the portion of this list between the specified `fromIndex`, inclusive, and `toIndex`, exclusive.

Question 79: How can you synchronize a "List"?

Answer:

```
List<String> syncList = Collections.synchronizedList(new ArrayList<String>());
```

Question 80: What are the benefits of using "Generics" with "List"?

Answer: Generics provide type safety. They ensure that you insert only the specified type of objects into your list. This reduces bugs and eliminates the need for typecasting.

Q: Do I need prior programming experience to learn Java?
A: While prior programming experience is beneficial, it is not necessary. Beginners can learn Java with the help of various resources like books, online courses, and practice.

Q: How important is it to understand object-oriented programming for Java?
A: Very important. Java is fundamentally object-oriented. Understanding OOP concepts, such as classes, objects, inheritance, polymorphism, etc., is crucial.

Q: What is the best way to practice Java programming?
A: The best way to practice is by coding regularly and solving problems on platforms like LeetCode, HackerRank, and Codecademy. Working on projects can also help in understanding how to apply Java concepts in real-world scenarios.

Q: Can I learn Java without installing any software?
A: Yes, there are several online IDEs like **Repl.it** and **JDoodle** that allows you to write and execute Java code without installing anything.

Q: What are common mistakes beginners make when learning Java?
A: Common mistakes include not learning the basics thoroughly, not practicing code regularly, and not understanding how to debug issues. Avoid these to improve your learning experience.