Interview Questions

JAVA-112: Session 7 - JUnit

Answering interview questions is crucial in your journey of applied learning. Review them to ensure your understanding of important topics covered in the previous session and to prepare yourself for upcoming challenges. Remember that it's important to answer these questions on your own before viewing the solution. The solutions are hyperlinked to community posts on our platform.

**Note: The questions below have been sourced from previous interviews**

Questions

1. [What is JUnit? Why is it important?](https://crio.do/learn/crio-community/topic/what-is-junit-why-is-it-important/214221)
2. [What is the classpath in Java?](https://crio.do/learn/crio-community/topic/what-is-the-classpath-in-java/214222)
3. [What is the default value of the classpath?](https://crio.do/learn/crio-community/topic/what-is-the-default-value-of-the-classpath/214223)
4. [Describe the order in which the JVM searches for classes specified in the classpath.](https://crio.do/learn/crio-community/topic/describe-the-order-in-which-the-jvm-searches-for-classes-specified-in-the-classpath/214224)
5. [What are the potential issues with setting the classpath incorrectly, and how can you troubleshoot them?](https://crio.do/learn/crio-community/topic/what-are-the-potential-issues-with-setting-the-classpath-incorrectly-and-how-can-you-troubleshoot-them/214225)
6. [What are Build Tools? Why do we need them?](https://crio.do/learn/crio-community/topic/what-are-build-tools-why-do-we-need-them/214226)
7. [What is unit testing? Why is it important?](https://www.crio.do/learn/crio-community/topic/what-is-unit-testing-why-is-it-important/214227/)
8. [Differentiate between JUnit annotations and assertions.](https://crio.do/learn/crio-community/topic/differentiate-between-junit-annotations-and-assertions-discuss-their-usecases/214228)

**1. What is JUnit? Why is it important?**

* **JUnit** is a popular **unit testing framework** for Java applications.
* It provides annotations and assertions that make it easy to write repeatable tests.
* **Importance:**
  + Ensures **code correctness** by verifying expected behavior.
  + Helps detect **bugs early** in the development cycle.
  + Promotes **test-driven development (TDD)**.
  + Encourages **modular design**, since code must be testable.

**2. What is the classpath in Java?**

* The **classpath** is a parameter in Java that tells the **Java Virtual Machine (JVM)** and **Java compiler** where to look for user-defined classes and packages.
* It includes:
  + The **current directory** (by default).
  + External JAR files.
  + Other directories containing compiled .class files.

**3. What is the default value of the classpath?**

* By default, the **classpath** is set to the **current directory (.)**.
* That means if you run java MyClass, the JVM looks for MyClass.class in the current folder unless specified otherwise.

**4. Describe the order in which the JVM searches for classes specified in the classpath.**

The JVM searches for classes in the following order:

1. **Bootstrap classloader** → Loads core Java classes from the rt.jar (Java Runtime Environment).
2. **Extension classloader** → Loads classes from the jre/lib/ext directory.
3. **System/Application classloader** → Loads classes from the locations specified in the **classpath** environment variable or -cp/-classpath command-line option.

**5. What are the potential issues with setting the classpath incorrectly, and how can you troubleshoot them?**

* **Potential Issues:**
  + ClassNotFoundException → JVM cannot find the specified class.
  + NoClassDefFoundError → Class was found at compile-time but missing at runtime.
  + Wrong version of a JAR may cause **compatibility issues**.
* **Troubleshooting:**
  + Check if the required JARs are included in the classpath.
  + Use the **-verbose:class** option to see which classes are loaded.
  + Make sure paths are separated correctly (; on Windows, : on Linux/Mac).
  + Verify spelling and case sensitivity of directory names.

**6. What are Build Tools? Why do we need them?**

* **Build tools** (like **Maven, Gradle, Ant**) automate the process of building, packaging, and deploying Java projects.
* **Need for build tools:**
  + Manage dependencies (external JARs).
  + Compile code and run unit tests automatically.
  + Create executable JARs/WARs.
  + Improve project consistency across environments.

**7. What is unit testing? Why is it important?**

* **Unit testing** is the process of testing individual components (methods, classes) of an application in isolation.
* **Importance:**
  + Detects bugs early.
  + Reduces debugging time.
  + Helps in **refactoring safely** (tests confirm behavior hasn’t changed).
  + Forms the foundation for **continuous integration (CI/CD)**.

**8. Differentiate between JUnit annotations and assertions.**

* **Annotations (@)** → Used to **control test execution**.
  + @Test – Marks a method as a test method.
  + @Before / @BeforeEach – Runs before each test.
  + @After / @AfterEach – Runs after each test.
  + @BeforeAll, @AfterAll – Run once before/after all tests.
* **Assertions (Assert methods)** → Used to **validate test results**.
  + assertEquals(expected, actual) – Checks if values match.
  + assertTrue(condition) – Verifies if condition is true.
  + assertNotNull(object) – Ensures object is not null.
  + fail() – Marks test as failed explicitly.

👉 **Summary:**

* **Annotations** → Control *when and how* tests run.
* **Assertions** → Verify *what is correct* in the test result.