1. **What is the Java Virtual Machine? What is Bytecode?**

Java Virtual Machine (JVM) is a virual machine that runs within the system. It interprets the bytecode that Java-code creates when executed. Bytecode is a kind of compiled programcode. What makes bytecode unique is that it doesn't (usally) rely on a machine architecture.

2. **What is the Java Classpath?**

The classpath provides the JVM where to find imported classes. Otherwise it would have to look in all your folders to find a class which would be very time consuming.

3. **How do you compile and run your java program without the help of an Integrated Development Environment (IDE) (e.g., an IDE like Eclipse)?**

With the help of the cmd. This will still require a JDK though. How to do it via the cmd : http://www.skylit.com/javamethods/faqs/javaindos.html

4. **What is a JAR file?**

It's a package file format that can contain several Java classes and any metadata/resources that comes with them. It's used to have easy distribution of a Java application.

5. **How do you declare the starting point of a Java application?**

A class that contains the public static void main(String args[]) function. This function name is case sensitive.

6. **What is a package? Why is important to declare classes inside packages?**

A package is similar to a folder. You use and create them to have order and structure in your Java application. For example if you're making a game you could have a Player package that would contain sub-packages, resources, classes etc. related to the Player.

7. **What is an *interface*? Why is it important to not change them?**

An interface declares structure rules an inherited class *must* follow. It's important not to change them because if you do you will have to do that change to all classes that inherit the interface.

8. **Which visibility levels are available in Java? What is the default visibility for classes, methods, and fields?**

Public, protected and private. With no modifier the default visibility is within the class, the package and any sub-classes.

9. **In the context of Java, what is an Exception? And what is an Error?**

Exception's are used and occurs for "softer" error handling, example: wrong user input, file not found/read correctly etc. Exception's should be handled within the program. Error's are much more servere and should not be handled within the program. If and error happens then in most cases the application should crash instead of trying to handle the error.

10. **<u>What happened if your program terminates with an *OutOfMemoryError*, or *NoClassDefFoundError*or*NullPointerException*?</u>**

OutOfMemoryError is thrown when the JVM is out of memory and no more could be made with the garbadge collector.
(src: https://docs.oracle.com/javase/7/docs/api/java/lang/OutOfMemoryError.html)
NoClassDefFoundError is thrown when the JVM can't find a definition of a class.
(src: https://docs.oracle.com/javase/7/docs/api/java/lang/NoClassDefFoundError.html)
NullPointerException is thrown when the value null is used in a case where an object value is required.
(src: https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html)

11. **<u>How do you handle Exceptions in your program?</u>**

By surrounding any code that could throw an Exception within a try statement and handle any exception thrown in the following catch statement/statemens.

12. **<u>Why is it important to test your code/application/product, before you deliver it to your customer/boss/teacher?</u>**

Without testing you can't carry confidence in a good quality/ working product that you deliver to your customer/boss/teacher. Even though a function might have worked when you first wrote it, later implemention might have affected it in some way (parameters etc.).

13. **<u>What is JavaDoc? How do you write documentation with it?</u>**

JavaDoc is a tool for generating HTML format of comments in Java source code. The comments have to follow a certain architecture. Different architectures exist for different type of documentation, for example varibles and constructors. Alot of different tags also exist to describe diffrent aspects (@author, @return, @param etc.).