Java Core Study Guide

Java

**What is Java? What are the benefits of Java?**

Java is a high-level, object-oriented programming language used to develop mobile and desktop applications.

* High level means that it closely resembles human language, so its easy to learn and interpret

**Benefits**:

1. Platform independent – means that complied java code is capable of running on any operating system.
   * Compiler turns written code into byte code that’s able to be read by the OS in use. **[1]**
2. Automatic memory management – java contains a tool known as a garbage collector that looks at the heap to determine which objects are in use and which objects are not in use so that it can clean up or remove the objects not in use.
   * This means that we do not have to implement management logic in our application. **[2]**
3. Java has a rich open-source community and is supported by Oracle
   * This means that there is a plethora of documentation available to teach you about java syntax, tools, and libraries.
   * Open-source tutorials and code examples available to learn from
   * A lot of documentation for java is created and posted by a Oracle, one of the largest software development companies in the country.
4. Multithreading capability – Process of executing multiple threads/ processes simultaneously.
   * Allows multiple parts of a program to run simultaneously instead of in the order that it is written. **[3]**

**What is the difference between the JDK, JRE, and JVM?**

* **JDK – Java development kit**
  + Software development environment used to develop java applications.
  + Provides interface to help guide or assist programmers while writing code.
  + Contains the compiler, debugger, and JRE.
* **JRE – Java runtime environment**
  + Needed to run java programs
  + Contains all of the runtime libraries used to and core classes that code will be calling to execute
  + Also contains the JVM
* **JVM – java virtual machine**
  + Provides runtime environment to execute java bytecode.
  + Specific to operating system in use.
  + Reads compiled java code and translates it into machine code that can be read and executed by the OS in use.
* **Java Compilation Process**
  + Javac or java compiler complies java code into bytecode.
  + The JVM then takes that java bytecode and into machine code that is native to the OS in use.
  + Machine code, which is a set of instructions readable by a computers CPU, are then executed. **[4]**

**What are the primitive datatypes?**

* whole numbers: byte, short, int, long
  + int – 32 bit
  + long 64 bit
* floating points: float, double
  + float – 32 bit
  + double – 64 bit
* character: char – 16 bit
* Boolean
* Use Case:
  + Normally, 32-bit. Why? Maximize storage.
  + 64- bit. Creating a variable with a large value.

**What is the difference between an object and a class?**

* A **class** is a template for the creation of an object. Template meaning it defines the qualities and functionality of a specific type of object.
* An **object** is an instance of a class. Many objects or instances can be created from a class because they typically contain the same elements.
* For example, we have a class called house. It contains information that describes what a house is. With java, we can use that house class to create instances of specific types of houses. These instances can then store their own descriptions using the house class.

**What is a constructor?**

* A method that declares how an object is to be created/instantiated and initialized.
  + What class is the object being created from, and what values are going to be initially assigned to it.

Different types of constructors

* Default constructor – automatically created if there is no constructor create for a class
* No-arg constructor – does not accept any parameters
* Parameterized constructor – takes in parameter values

Constructor chaining

* Can be done in the same class using the **this()** keyword.
* Can use **super()** to call constructor from parent/base class. **[5]**

**What are the scopes of a variable in java?**

1. Method scope

* variable lives within the scope of the methods
* include the parameters and variables declared within that scope

1. Instance/object scope
2. Static/class scope
3. Loop scope

**What are arrays and how would you create an array in java?**

* An Array is a way to store a group of elements with the same data type in sequential order.
* To declare an array, or create a reference we can use the code: ‘datatype[] arrayName;’ or ‘dataType arrayName[];’
* Then to give memory to the create we can use the code ‘arrayName = new arrayType[];’
* To declare the size of the array we could use: ‘int[] var = new int[5];’
* To create an array we could use: ‘int[] var = {1, 2, 3, 4, 5};’

**What are packages and imports?**

* Packages are a container to group related types of files like java classes, interfaces, and enumerations.
* Used for better organization and also resolves name conflicts.
  + If two classes have the same name but they are in different packages, then there will not be a collision whenever some code is ran.
* The imports keyword can be used to allow a java class to refer to other classes, and can also be used to import libraries that include tools used to create java applications.

**What are Strings?**

Strings are a sequence or array of characters.

Important String Methods:

* toCharArray() – turns string into character array
* length() – gets the length of the string
* concat() – used to add a string to another string
* .equals() compares two strings
* charAt() – returns the character at a specified index

Git

**What is version Control?**

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later

**What is the difference between Git and Github?**

* Git is a version control system that allows you to manage and keep track of your codes history.
* Github is a cloud-based service that allows you to create and manage repositories.

**List the git commands you know and what they do?**

* **Git add .** – adds an entire directory to the staging area
  + staging is essentially a preview of what is to be committed
* **git status** – displays the state of the working directory and staging area
  + shows you which changes have been staged, and what is being currently tracked by Git.
* **Git commit -m “message” –** snapshot of changes made, used for version control
* **Git push** – sends changes that were committed to remote repository.

**How would you prevent a file from being tracked by git?**

You could create a .gitignore file which tell Git which files or folders to ignore.

**What is the git workflow for editing code and saving changes?**

When editing code you want to go ahead an use git add to add files or directories to a staging area. You can then use git status to view the status of the staging area. Once your satisfied with the changes made, then you can use git commit to commit the changes, thus creating a snapshot of the changes. Then you can finally use git push to send that code to a remote repository.

**What is a commit?**

A commit is essentially a way to capture the state of a project currently staged changes at a point in time. Every time you make a change to the state of the project you can commit your changes.

**What is a branch? What are some common branching strategies?**

A branch is a version of a main repos

References

[1] - <https://www.geeksforgeeks.org/java-platform-independent/>

[2]- <https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html>

[3] - <https://www.tutorialcup.com/java/multithreading-java.htm>

[4] - <https://www.programiz.com/java-programming/jvm-jre-jdk>

[5] - <https://www.geeksforgeeks.org/constructor-chaining-java-examples/>