**Difference between Java 7 and Java 8**

**Lambada expression: -** Java 8 brings the most anticipated feature for the programming language called Lambda Expressions, a new language feature which allows users to code local functions as method argument.

### **Specialized API for Date and Time manipulation: -** Java 8 brings its own new specialized API for Date and Time manipulation. **New and improved JavaScript engine, Nashorn :-** which allows developers to run the script on a JVM. The idea was to implement a lightweight JavaScript runtime in the programming language with a native JVM **What Methods Does the Object Class Have? https://dzone.com/articles/top-10-java-interview-questions-that-i-recently-fa**

* Clone
* Equals
* Finalize
* Getclass
* Hashcode
* Equals
* Tostring
* Wait
* Notify
* NotifyAll

### **Why Is the String Object Immutable in Java?**

* **String Pool** is possible only because String is immutable in Java.

This way, Java Runtime saves a lot of Java heap space, because different String variables can refer to the same String variable in the pool.

If String is not immutable, then String interning would not have been possible, because if any variable would have changed the value, it would have been reflected in other variables.

* **Security**
* **Strings are implicitly thread safe.**
* **Java Class Loader Use it**
* **Hashcode –** Fast if used as hashmap

### **What Is the Difference Between Final, Finally, and Finalize?**

* The *final* keyword is used in several contexts to define an entity that can only be assigned once.
* **The Java finally block** is a block that is used *to execute important code,* such as closing connection, stream, etc. The Java finally block is always executed, whether the exception is handled or not. Java finally block follows the try or catch block.
* This is a **method** that the GarbageCollector always calls just **before** the deletion/destroying the object, which is eligible for Garbage Collection to perform **clean-up activity**.

### **How Can You Make a Class Immutable?**

1. **Final** Declare the class as final so it can’t be extended.
2. **All fields private: -**Make all fields private so that direct access is not allowed.
3. **Do not provide setter method: -** Don’t provide setter methods for variables
4. Make all **mutable fields final** so that it’s value can be assigned only once.
5. Initialize all the fields via a constructor performing a deep copy.
6. Perform cloning of objects in the getter methods to return a copy rather than returning the actual object reference.

### **. What Is the Difference Between Static and Dynamic Loading?**

Static class loading involves the creation of objects and instances using new keywords, and dynamic class loading is done when the name of the class is not known at compile time.

### **What Is Multi-Threading?**

Multi-threading is a programming concept used to run multiple tasks in a concurrent manner within a single program.

### **What Are the Functions of JVM and JRE?**

JVM provides a runtime environment for Java Byte Codes to be executed. JRE includes sets of files required by JVM during runtime.

### **What Is the Difference Between equals() and = = ?**

Equals() method is used for checking the equality of two objects defined by business logic.

== or the equality operator is used to compare primitives and objects

#### **What are the main 3 Object Oriented Programing (OOP) concepts?**

#### **1. Encapsulation**

* Encapsulation is a mechanism by which developer could hide implementation behind an [interface](https://crunchify.com/java-how-to-sort-a-map-on-the-values-the-map-interface-java-collections/).
* Encapsulated code has two features:
  + Instance variables are kept protected (usually with the private modifier).
  + Getter and setter methods provide access to instance variables.
* Kindly take a look at tutorial: [Complete End to End working Encapsulation example](https://crunchify.com/complete-end-to-end-java-tutorial-with-singleton-object-employee-crunchify-object-pojo-detailed-testcase/)
  + The public setName() and getName() methods are the access points of the instance variables.

#### **2. Inheritance**

* Inheritance allows a class to be a subclass of a superclass, and thereby inherit public and protected variables and methods of the superclass.
* Inheritance is a key concept that underlies polymorphism, overriding, overloading and casting.

#### **3. Polymorphism**

* Polymorphism means “many forms.”
* A [reference variable](https://crunchify.com/fundamentals-of-java-static-methods-and-variables/) is always of a single, unchangeable type, but it can refer to a subtype object.
* A single object can be referred to by reference variables of many different types —as long as they are the same type or a supertype of the object.
* Polymorphic method invocations apply only to [overridden](https://crunchify.com/java-method-overriding-examples-and-concepts-overriding-rules/) instance methods.

#### **Are you aware of Daemon Thread in Java?**

Daemon threads in [Java](https://crunchify.com/category/java-tutorials/) are like a service providers for other threads or objects running in the same process as the daemon [thread](https://crunchify.com/category/java-tutorials/). Daemon threads are used for background supporting tasks and are only needed while normal threads are executing. If normal threads are not running and remaining threads are daemon threads then the interpreter exits.

Kindly take a look at complete example: [https://crunchify.com/what-is-daemon-thread-in-](https://crunchify.com/what-is-daemon-thread-in-java-example-attached/)

#### **What is Singleton Pattern and Do you know how to make it Thread-Safe and Fast?**

The singleton pattern is a design pattern that restricts the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system.

Kindly take a look at detailed example: <https://crunchify.com/thread-safe-and-a-fast-singleton-implementation-in-java/>

<https://crunchify.com/top-10-java-interview-questions-answers-must-read-before-appearing-for-any-java-interview/>

Collection interview questions

<https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html#axzz6mbhksJQc>

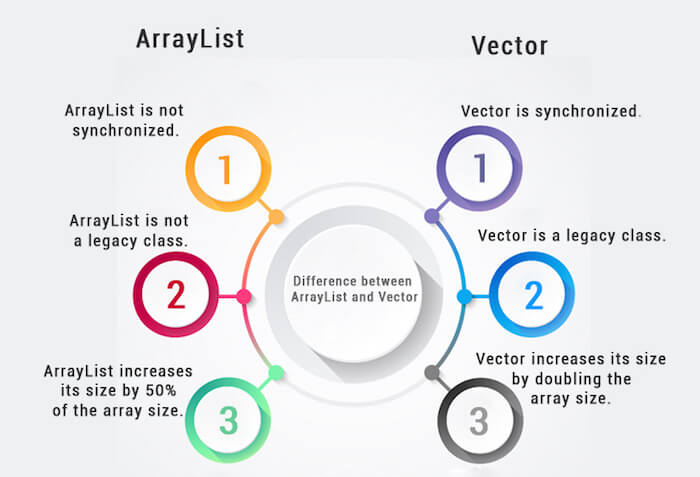
HashMap vs HashTable

|  |  |  |
| --- | --- | --- |
|  | HashMap | HashTable |
| Thread Safety | Not Thread Safe | Thread Safe |
| Speed | Fast | Slow |
|  |  |  |

### **What is the difference between Hashtable and ConcurrentHashMap in Java?**

ConcurrentHashMap are thread-safe here but later provides more scalability than former.

|  |  |  |
| --- | --- | --- |
|  | **ArrayList** | **Vector** |
|  | ArrayList is **not synchronized**. | Vector is **synchronized**. |
|  | ArrayList **increments 50%** of current array size if the number of elements exceeds from its capacity | Vector **increments 100%** means doubles the array size if the total number of elements exceeds than its capacity. |
|  | ArrayList is **not a legacy** class. It is introduced in JDK 1.2. | Vector is a **legacy** class. |
|  | ArrayList is **fast** because it is non-synchronized. | |  | | --- | | Vector is **slow** because it is synchronized, i.e., in a multithreading environment, it holds the other threads in runnable or non-runnable state until current thread releases the lock of the object. | |
|  | ArrayList uses the **Iterator** interface to traverse the elements. | A Vector can use the **Iterator** interface or **Enumeration** interface to traverse the elements. |



### **What Is the Difference Between Static and Dynamic Loading?**

**New Keyword: -** Static class loading involves the creation of objects and instances using new keywords,

**Name Of Class Not know At Compile Time: -** dynamic class loading is done when the name of the class is not known at compile time.

### **What Is Multi-Threading?**

**Multiple Task Concurrent Manner: -** Multi-threading is a programming concept used to run multiple tasks in a concurrent manner within a single program.

### **When and by Whom was Java Developed?**

Java was developed by James Gosling in Sun Microsystem in 1995.

### **What Do JDK, JRE, and JVM Stand for?**

• JVM stands for Java Virtual Machine

• JRE stands for Java Runtime Environment

• JDK stands for Java Development Kit

