Java Type Casting

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

* **Widening Casting** (automatically) - converting a smaller type to a larger type size  
  byte -> short -> char -> int -> long -> float -> double
* **Narrowing Casting** (manually) - converting a larger type to a smaller size type  
  double -> float -> long -> int -> char -> short -> byte

## Widening Casting

Widening casting is done automatically when passing a smaller size type to a larger size type:

int myInt = 9;

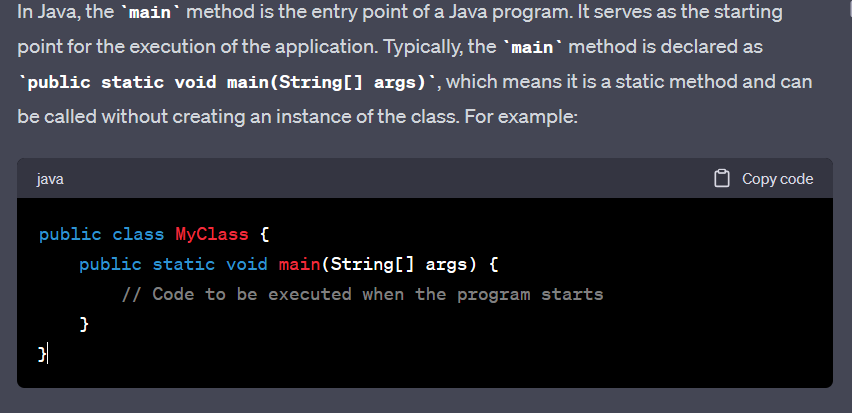
double myDouble = myInt; // Automatic casting: int to double

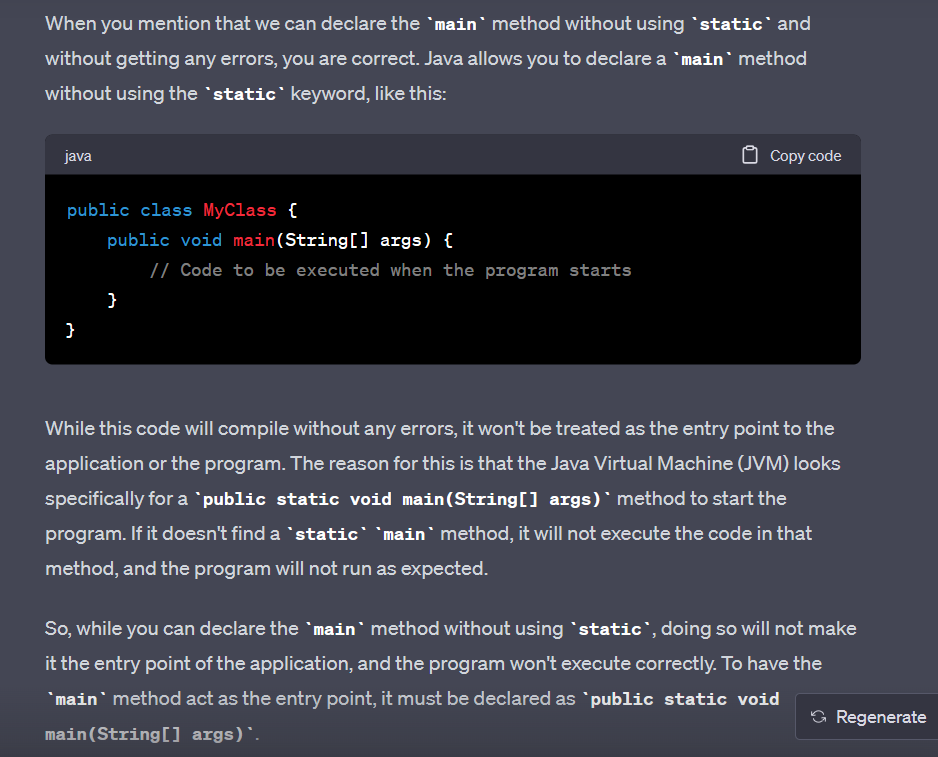
## Narrowing Casting

Narrowing casting must be done manually by placing the type in parentheses in front of the value:

double myDouble = 9.78d;

int myInt = (int) myDouble; // Manual casting: double to int





### **What are Packages in Java?**

- Packages in Java can be defined as the grouping of related types of classes, interfaces, etc providing access to protection and namespace management.

- Packages are used in Java in **order to prevent naming conflicts, control access, and make searching/locating and usage of classes, interfaces, etc easier.**

**There are various advantages of defining packages in Java**.

* Packages avoid name clashes.
* The Package provides easier access control.
* We can also have the hidden classes that are not visible outside and are used by the package.
* It is easier to locate the related classes.

**There are two types of packages in Java**

* User-defined packages
* Build In packages

### **different data types in Java.**

1. **Primitive Data Type :**

Primitive data are single values with no special capabilities. There are 8 primitive data

1. **Non-Primitive Data Type or Object Data type:**

Reference Data types will contain a memory address of the variable’s values because it is not able to directly store the values in the memory.

**Why java not provide pointer ?**

**1)** Java was designed to be a simple and safe language for general-purpose programming. Pointers introduce complexities and potential errors, such as segmentation faults and memory leaks. By avoiding pointers, Java aims to make

**2) Garbage Collection:**

Java uses automatic memory management through a process known as garbage collection. This system eliminates the need for developers to explicitly allocate and deallocate memory using pointers. Garbage collection helps prevent memory leaks and simplifies memory management for developers.

**3) Object-Oriented Paradigm:**

Java promotes an object-oriented programming (OOP) approach, where data and methods are encapsulated within objects. Pointers might hinder the encapsulation and abstraction principles of OOP by allowing direct manipulation of memory outside the context of objects.

*Although Java doesn't have explicit pointers, it has references, which are similar in concept but offer a higher level of abstraction. In Java, objects are accessed through references, and developers don't need to worry about the memory management details associated with traditional pointers in languages like C or C++. Instead, they can focus on writing safer and more maintainable code using Java's object-oriented features.*

# **Wrapper classes in Java**

The **wrapper class in Java** provides the mechanism **to convert primitive into object and object into primitive.**

***USES :***

* **Change the value in Method:** Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
* **Serialization:** We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
* **Synchronization:** Java synchronization works with objects in Multithreading.

## Autoboxing

***The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing,*** for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

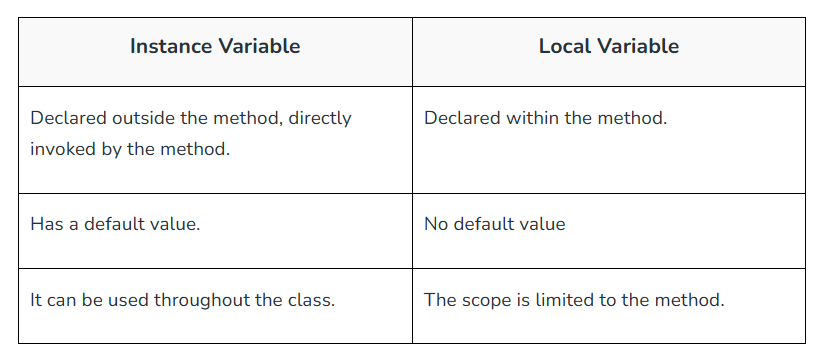
## Unboxing

***The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing.*** It is the reverse process of autoboxing.

### **Why do we need wrapper classes?**

The wrapper class is an object class that encapsulates the primitive data types, and we need them for the following reasons:

1. ***Wrapper classes are final and immutable***
2. ***Provides methods like valueOf(), parseInt(), etc.***
3. ***It provides the feature of autoboxing and unboxing.***



In summary, local variables are temporary variables declared inside methods or blocks of code, while instance variables are associated with instances (objects) of a class and represent the state of the objects. Understanding the difference between local and instance variables is essential for writing well-structured Java programs.

***Local Variables:***

Local variables are declared inside a method, constructor, or block of code and are accessible only within that particular block.

They must be initialized before they are used.

Local variables have no default values and must be explicitly assigned a value before being accessed.

They are temporary and exist only as long as the method or block of code in which they are declared is executing.

Local variables are not shared between different methods or instances of the class and are used for storing temporary data or intermediate results.

***Instance Variables (also called Member Variables or Fields):***

Instance variables are declared within a class but outside any method or block of code. They are associated with instances (objects) of the class.

Each instance of the class will have its copy of instance variables, and their values are independent of each other.

Instance variables are initialized with default values if not explicitly set: 0 for numeric types, false for boolean, and null for object references.

They exist as long as the instance of the class exists, meaning they have a lifespan equal to that of the object.

Instance variables are used to hold state or characteristics of the objects created from the class.

***“ A class variable without a static modifier known as an instance variable is typically shared by all instances of the class.”*** These variables can have distinct values among several objects

**Class Variable:**

Class Variable variable can be declared anywhere at the class level using the keyword static. These variables can only have one value when applied to various objects. These variables can be shared by all class members since they are not connected to any specific object of the class.

**A static variable *:*** in Java is a **class-level** variable that belongs to the class itself, rather than to any specific instance (object) of the class. It is also known as a "class variable" because it is shared among all instances of the class. Each instance of the class accesses and shares the same copy of the static variable, which means any modification to the static variable will be visible to all instances.

The static keyword is used to share the same variable or method of a given class

### **What are the super most classes for all the streams?**

All the stream classes can be divided into two types of classes that are ByteStream classes and CharacterStream Classes. The ByteStream classes are further divided into InputStream classes and OutputStream classes. CharacterStream classes are also divided into Reader classes and Writer classes. The SuperMost classes for all the InputStream classes is java.io.InputStream and for all the output stream classes is java.io.OutPutStream. Similarly, for all the reader classes, the super-most class is java.io.Reader, and for all the writer classes, it is java.io.Writer.

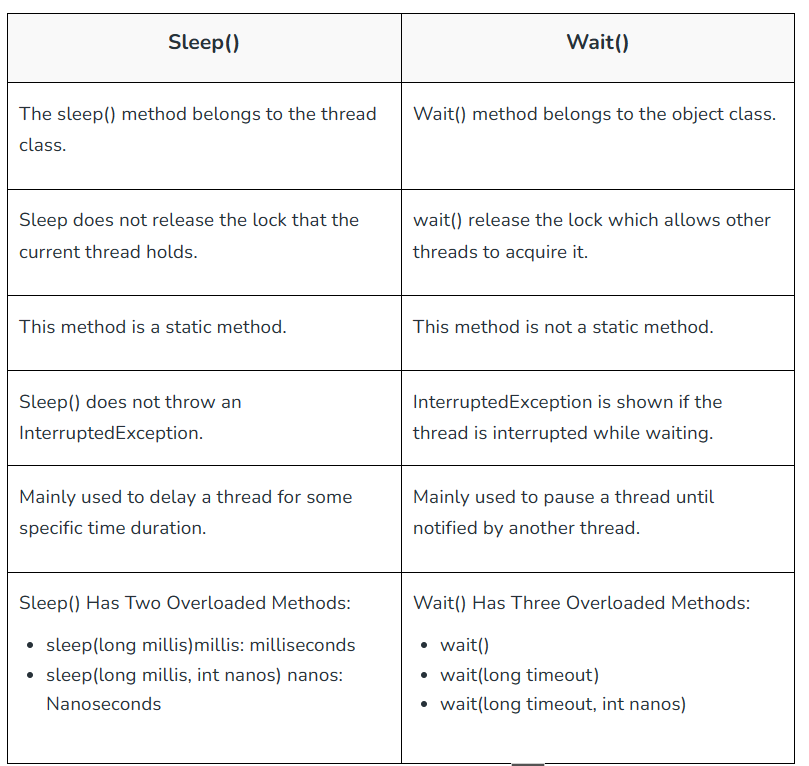
### **Explain the difference between >> and >>> operators.**

Operators like >> and >>> seem to be the same but act a bit differently. >> operator shifts the sign bits and the >>> operator is used in shifting out the zero-filled bits.

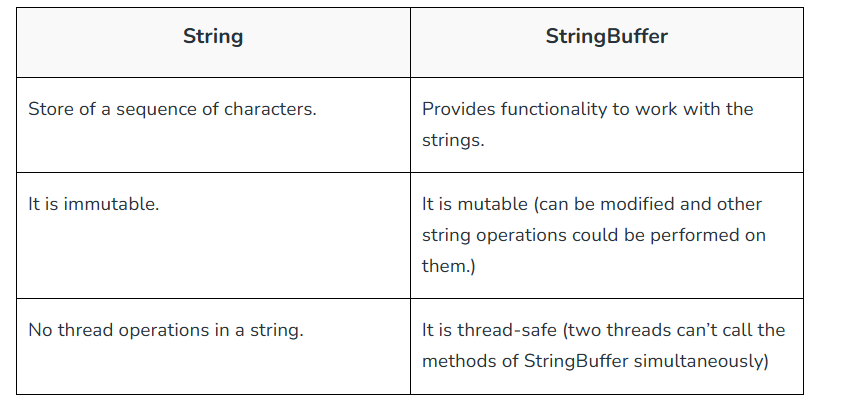
### **What is the transient keyword?**

The transient keyword is **used at the time of serialization if we don’t want to save the value of a particular variable in a file**. **When JVM comes across a transient keyword,** **it ignores the original value of the variable and saves the default value of that variable data type.**

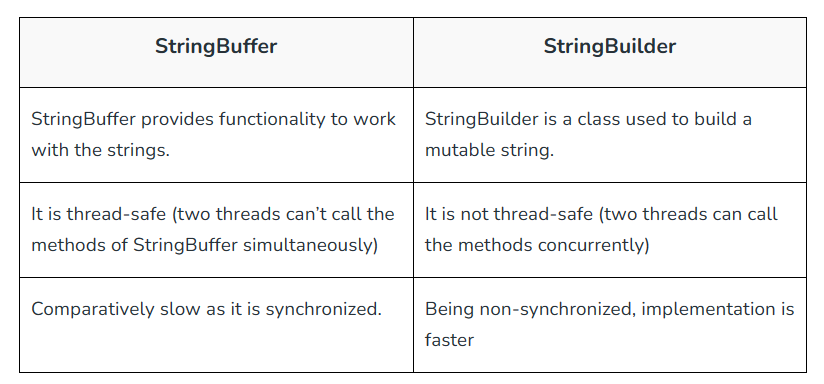
### **difference between the methods sleep() and wait()?**



### **differences between String and StringBuffer?**



### **differences between StringBuffer and StringBuilder?**



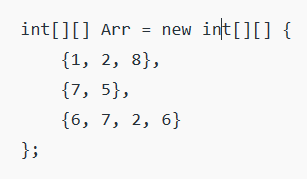
### **How is the creation of a String using new() different from that of a literal?**

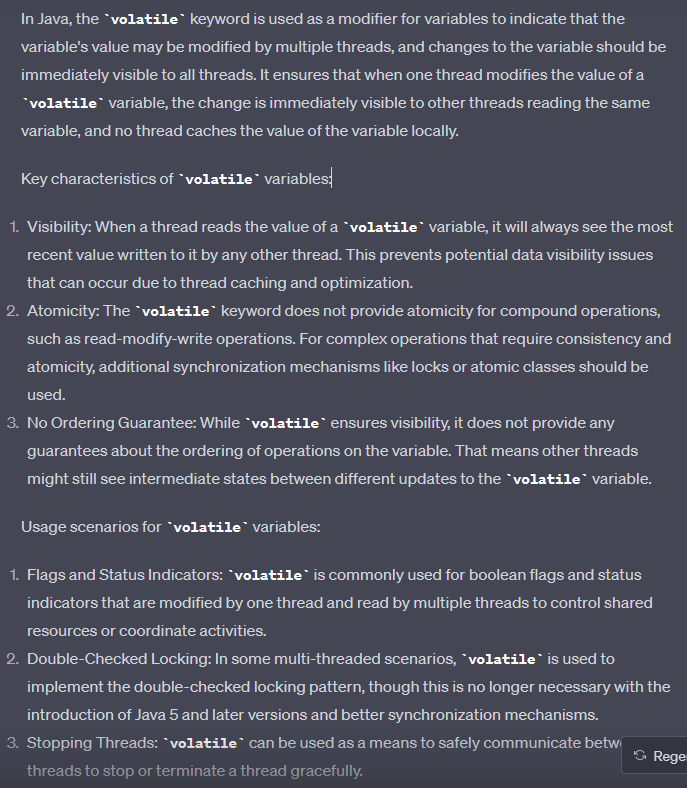
*String using new() is different from the literal as when we declare string it stores the elements inside the stack memory whereas when it is declared using new() it allocates a dynamic memory in the heap memory. The object gets created in the heap memory even if the same content object is present.*

***ARRAY :***

**Arrays in Java are created in heap memory.** When an array is created with the help of a new keyword**, memory is allocated in the heap to store the elements of the array.** In Java, the heap memory is managed by the Java Virtual Machine(JVM) and it is also shared between all threads of the Java Program. **The memory which is no longer in use by the program, JVM uses a garbage collector to reclaim the memory. Arrays in Java are created dynamically which means the size of the array is determined during the runtime of the program**. **The size of the array is specified during the declaration of the array and it cannot be changed once the array is created.**

**A jagged Array** in Java is just **a two-dimensional array in which each row of the array can have a different length**.

**

**

