Java Basics

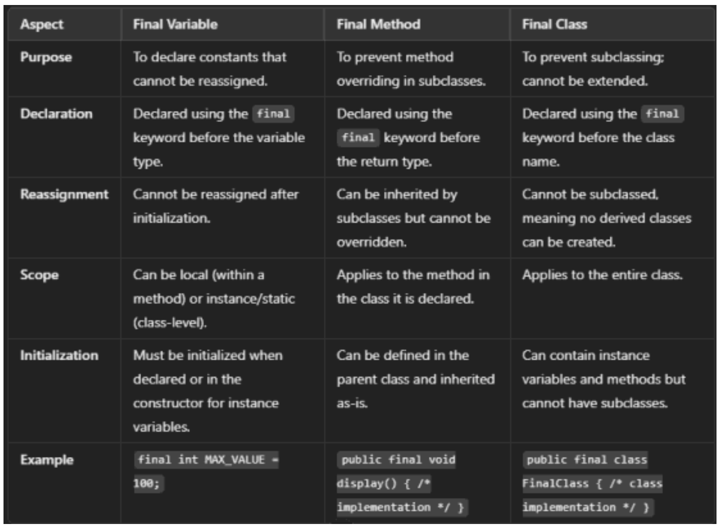
27 September 2024 20:07

Constants:

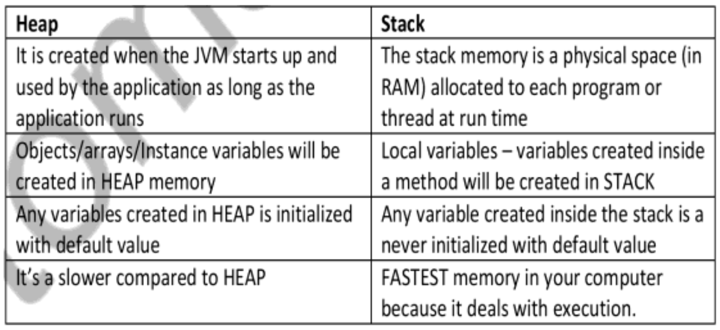
- *In Java, constants are fixed values that cannot be modified once they are assigned. They are often used to represent values that remain unchanged throughout the program, They enhance code readability, maintainability, and clarity by providing meaningful names for static values. Constants are typically defined using the **final** keyword. Constants are typically named using uppercase letters with words separated by underscores (_). This makes them easily distinguishable from regular variables*

- ***Global Constants:*** *If we define any static variable with final keyword*
- ***Object Constants:*** *If we define any non static variable with final keyword*
- ***Local Constants:*** *If we define any local variable with final keyword*

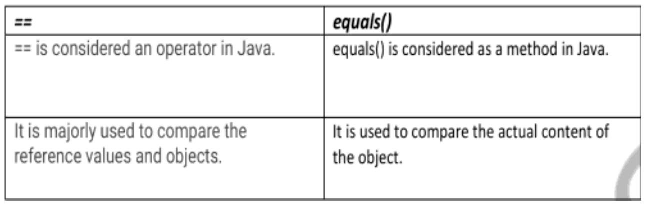
Final Keyword:



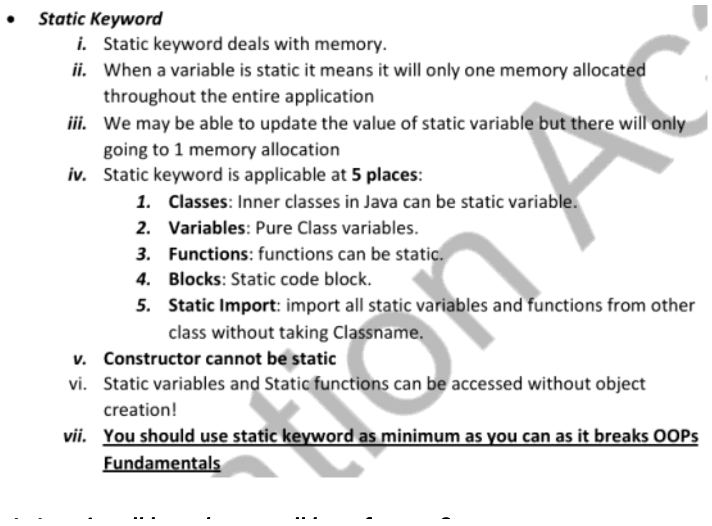
Aspect	Final Variable	Final Method	Final Class
Purpose	To declare constants that cannot be reassigned.	To prevent method overriding in subclasses.	To prevent subclassing; cannot be extended.
Declaration	Declared using the `final` keyword before the variable type.	Declared using the `final` keyword before the return type.	Declared using the `final` keyword before the class name.
Reassignment	Cannot be reassigned after initialization.	Can be inherited by subclasses but cannot be overridden.	Cannot be subclassed, meaning no derived classes can be created.
Scope	Can be local (within a method) or instance/static (class-level).	Applies to the method in the class it is declared.	Applies to the entire class.
Initialization	Must be initialized when declared or in the constructor for instance variables.	Can be defined in the parent class and inherited as-is.	Can contain instance variables and methods but cannot have subclasses.
Example	`final int MAX_VALUE = 100;`	`public final void display() { /* implementation */ }`	`public final class FinalClass { /* class implementation */ }`



Heap	Stack
It is created when the JVM starts up and used by the application as long as the application runs	The stack memory is a physical space (in RAM) allocated to each program or thread at run time
Objects/arrays/Instance variables will be created in HEAP memory	Local variables – variables created inside a method will be created in STACK
Any variables created in HEAP is initialized with default value	Any variable created inside the stack is a never initialized with default value
It's a slower compared to HEAP	FASTEST memory in your computer because it deals with execution.



==	*equals()*
== is considered an operator in Java.	equals() is considered as a method in Java.
It is majorly used to compare the reference values and objects.	It is used to compare the actual content of the object.



- ***Static Keyword***
  - ***i.*** Static keyword deals with memory.
  - ***ii.*** When a variable is static it means it will only one memory allocated throughout the entire application
  - ***iii.*** We may be able to update the value of static variable but there will only going to 1 memory allocation
  - ***iv.*** Static keyword is applicable at **5 places**:
    1. **Classes**: Inner classes in Java can be static variable.
    2. **Variables**: Pure Class variables.
    3. **Functions**: functions can be static.
    4. **Blocks**: Static code block.
    5. **Static Import**: import all static variables and functions from other class without taking Classname.
  - ***v.*** **Constructor cannot be static**
  - vi. Static variables and Static functions can be accessed without object creation!
  - ***vii.*** **<u>You should use static keyword as minimum as you can as it breaks OOPs Fundamentals</u>**

What Java Package?

- *A **Java package** is a grouping of related classes and interfaces that provides namespace management, access protection, and code organization.*
- *Packages help avoid naming conflicts and facilitate code reuse and maintenance.*
- *Java has built-in packages as well as the ability for developers to create user-defined packages*

In Java, objects can be created in various ways, including:

- *Using the new keyword*
- *Class.newInstance()*
  Bank b1 = Bank.class.newInstance();
- *Object.clone()*

What is out in System.out.println?

- *In Java System.out.println(): The statement can be broken into 3 parts which can be understood separately as:*
  - ***System**: It is a final class defined in the java.lang package*
  - ***.out:** This is an instance of PrintStream type, which is a public and static member field of the System class*
  - *.**println():** As all instances of PrintStream class have a public method println(), hence we can invoke the same on out as well*

Why Java is not 100% Object-oriented?

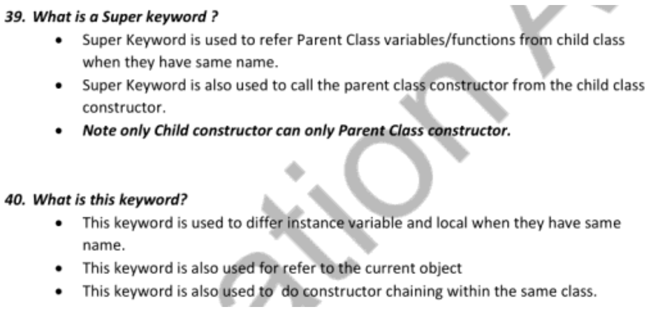
- *Java still supports Primitive datatypes even though Wrapper classes. Which breaks the rule of reference variable and encapsulation*
- *Java also uses static keywords which enables to access the variables and functions without creating the project. Which breaks the rule that object needs to be created for* accessing *variables and function.*
- *Support for Functional Programming*

What are object classes and name some object class methods?

- *Object class is parent class or super class for the all Classes in Java.*
- *Some popular functions that are present in Object class:*
  - ***toString():** returns the string representation of this object.*
  - ***hashCode():**returns the hashcode number for this object*
  - ***equals(Object obj)**: compares the given object to this object.*
  - ***finalize():** is invoked by the garbage collector before object is being garbage collected.*

Is there Constructor class in Java?

- *Yes, This class is present in package java.lang.reflect*
- *It is used to manage the constructor metadata like the name of the constructors, parameter types of constructors, and access modifiers of the constructors.*

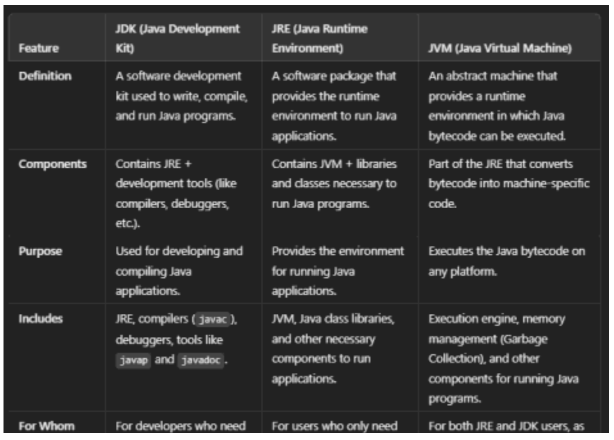


39. What is a Super keyword ?

- Super Keyword is used to refer Parent Class variables/functions from child class when they have same name.
- Super Keyword is also used to call the parent class constructor from the child class constructor.
- ***Note only Child constructor can only Parent Class constructor.***

40. What is this keyword?

- This keyword is used to differ instance variable and local when they have same name.
- This keyword is also used for refer to the current object
- This keyword is also used to do constructor chaining within the same class.

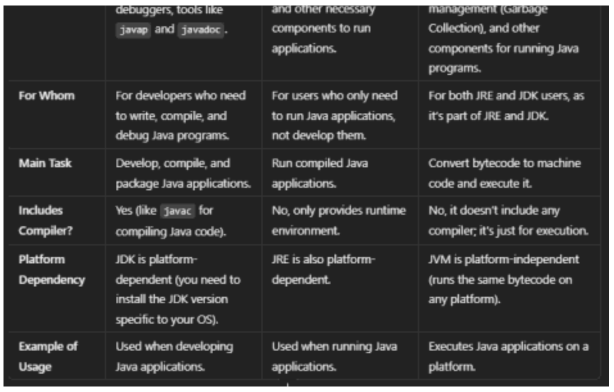


Feature	JDK (Java Development Kit)	JRE (Java Runtime Environment)	JVM (Java Virtual Machine)
Definition	A software development kit used to write, compile, and run Java programs.	A software package that provides the runtime environment to run Java applications.	An abstract machine that provides a runtime environment in which Java bytecode can be executed.
Components	Contains JRE + development tools (like compilers, debuggers, etc.).	Contains JVM + libraries and classes necessary to run Java programs.	Part of the JRE that converts bytecode into machine-specific code.
Purpose	Used for developing and compiling Java applications.	Provides the environment for running Java applications.	Executes the Java bytecode on any platform.
Includes	JRE, compilers (`javac`), debuggers, tools like `javap` and `javadoc`.	JVM, Java class libraries, and other necessary components to run applications.	Execution engine, memory management (Garbage Collection), and other components for running Java programs.
For Whom	For developers who need	For users who only need	For both JRE and JDK users, as

vii. **You should use static keyword as minimum as you can as it breaks OOPs Fundamentals**

Is Java is call by value or call by reference?

- *Java is call by value, but it behaves differently depending on whether you're dealing with primitives or objects:*
    - ***For primitives:** Java passes the **value** of the variable, and changes made to the parameter inside the method do not affect the original variable.*
    - ***For objects:** Java passes the **reference by value**, so changes to the object's state inside the method affect the original object, but reassigning the reference does not.*
- *If we call a method passing a value, it is known as call by value. The changes being done in the called method, is not affected in the calling method*



	debuggers, tools like `javap` and `javadoc`.	and other necessary components to run applications.	management (Garbage Collection), and other components for running Java programs.
For Whom	For developers who need to write, compile, and debug Java programs.	For users who only need to run Java applications, not develop them.	For both JRE and JDK users, as it's part of JRE and JDK.
Main Task	Develop, compile, and package Java applications.	Run compiled Java applications.	Convert bytecode to machine code and execute it.
Includes Compiler?	Yes (like `javac` for compiling Java code).	No, only provides runtime environment.	No, it doesn't include any compiler; it's just for execution.
Platform Dependency	JDK is platform-dependent (you need to install the JDK version specific to your OS).	JRE is also platform-dependent.	JVM is platform-independent (runs the same bytecode on any platform).
Example of Usage	Used when developing Java applications.	Used when running Java applications.	Executes Java applications on a platform.

Java Garbage Collection

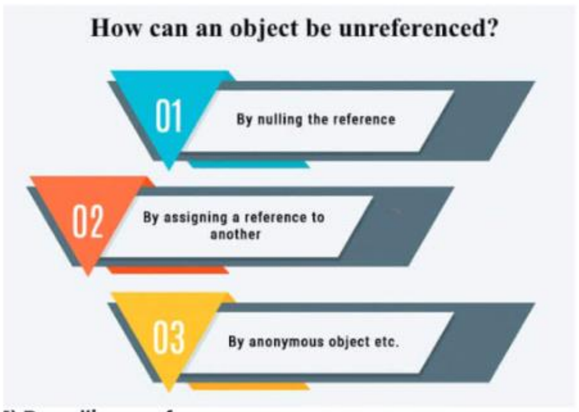
In java, garbage means unreferenced objects.

Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

To do so, we were using free() function in C language and delete() in C++. But, in java it is performed automatically. So, java provides better memory management.

Advantage of Garbage Collection

- It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
- It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.


How can an object be unreferenced?
01
By nulling the reference
02
By assigning a reference to another
03
By anonymous object etc.

1) By nulling a reference:

```
Employee e=new Employee();
e=null;
```

2) By assigning a reference to another:

```
Employee e1=new Employee();
Employee e2=new Employee();
e1=e2;//now the first object referred by e1 is available for garbage collection
```

3) By anonymous object:

```
new Employee();
```