## ****Global Variable:****

Global variables are variables that are declared outside of any function or block of code and can be accessed from any part of the program. Unlike local variables, which have limited scope, global variables have a broader scope and can be used across multiple functions, modules, or files within a program. Here are some characteristics, features, advantages, disadvantages, and uses of global variables:

## Example of Global Variable:

public class Main {

// Global variable declaration

static int global\_var = 100;

​

public static void exampleFunction() {

// Local variable declaration

int x = 10;

int y = 20;

int z = x + y + global\_var;

System.out.println("The sum is: " + z);

}

​

public static void main(String[] args) {

exampleFunction();

}

}

**Output**

The sum is: 130

**Advantages of global variable:**

1. **Accessibility**: Global variables provide a convenient way to share data across different parts of the program without passing them as function arguments.
2. **Ease of Use**: They simplify the sharing of data between functions and modules, reducing the need for complex parameter passing mechanisms.
3. **Persistence**: Global variables retain their values throughout the entire execution of the program, making them suitable for storing persistent data.
4. **Reduced Code Duplication**: Global variables can help reduce code duplication by centralizing data that is used in multiple parts of the program.

**Disadvantages of global variable:**

1. **Encapsulation Issues**: Global variables can lead to encapsulation issues by allowing any part of the program to modify their values, potentially leading to unintended side effects.
2. **Debugging Complexity**: Since global variables can be accessed and modified from anywhere in the program, tracking down bugs related to their usage can be challenging.
3. **Potential for Race Conditions**: In multithreaded or concurrent programs, global variables can introduce race conditions if accessed and modified concurrently by multiple threads or processes.
4. **Maintainability**: Excessive use of global variables can make code harder to understand and maintain, as their effects may not be localized to specific functions or modules.

Static variables (also known as class variables) are declared with the static keyword and belong to the class itself, rather than to any specific instance of the class. This means there is only one copy of a static variable, which is shared among all instances of the class and can be accessed directly using the class name. This makes them effectively "global" within the scope of the application.

Example:

Java

public class GlobalVariableExample {  
  
 *// A static variable acting as a global variable*  
 public static int globalCounter = 0;  
  
 *// A static final variable acting as a global constant*  
 public static final String APP\_NAME = "My Global App";  
  
 public void incrementCounter() {  
 globalCounter++;  
 System.out.println("Counter in incrementCounter(): " + globalCounter);  
 }  
  
 public static void main(String[] args) {  
 *// Accessing and modifying the static variable directly*  
 System.out.println("Initial globalCounter: " + globalCounter);  
 globalCounter = 5;  
 System.out.println("Modified globalCounter: " + globalCounter);  
  
 *// Accessing the static final variable (constant)*  
 System.out.println("Application Name: " + APP\_NAME);  
  
 *// Creating an instance to call an instance method that uses the static variable*  
 GlobalVariableExample obj1 = new GlobalVariableExample();  
 obj1.incrementCounter(); *// globalCounter becomes 6*  
  
 GlobalVariableExample obj2 = new GlobalVariableExample();  
 obj2.incrementCounter(); *// globalCounter becomes 7 (shared among instances)*  
  
 System.out.println("Final globalCounter in main: " + globalCounter);  
 }  
}

Explanation:

* public static int globalCounter = 0;: This declares a static integer variable globalCounter. It is initialized to 0 and can be accessed and modified from anywhere within the GlobalVariableExample class or even from other classes using GlobalVariableExample.globalCounter.
* public static final String APP\_NAME = "My Global App";: This declares a static final String variable APP\_NAME. The final keyword makes it a constant, meaning its value cannot be changed after initialization. This is a common way to define application-wide constants.
* In the main method, globalCounter is accessed and modified directly using the class name (GlobalVariableExample.globalCounter).
* The incrementCounter() instance method also modifies globalCounter. Since globalCounter is static, changes made by one instance are reflected for all other instances and direct access.