Day_9_OOPJ_Sanket_Shalukar

Sunday, September 07, 2025 8:35 AM

Topics are in the Day_9

1. Final variable
2. String
3. Garbage Collection

Final variable.

1. `final` (keyword)

A keyword used to declare constants, prevent method overriding, and prevent class inheritance.

- **Usage:** variable, method, class.
- **final variable** → value cannot be changed (constant).
- **final method** → cannot be overridden.
- **final class** → cannot be inherited.

```
class A{
    final int x = 100;//constant:Compile time constant

    void display(){
        //x=200;//Error: cannot reassign the value
        System.out.println(x);
    }
}

public class FinalDemo1{



    public static void main(String args[]){

        A a1 =new A();//instance
        a1.display();


    }
}
```

- A **final variable** means its value cannot be changed once assigned.
- In this example, the variable is not given a value immediately. Instead, it is assigned a value **inside the constructor**.
- Each time you create a new object, the constructor runs and gives that object its own value of the final variable.
- Once the value is set for that object, it **cannot be reassigned** anywhere else.
- That's why when you create two objects with different values, each object keeps its own fixed value.

`finally` (block)

A block used in exception handling that always executes whether an exception occurs or not.

- Used in **exception handling** (`try-catch-finally`).
- Code in `finally` block **always executes** (whether exception occurs or not).
- Typically used to **release resources** (close files, DB connections).

`finalize()` (method)

A method in the Object class called by the Garbage Collector before destroying an object.

- A **method of Object class**.
- Called by **Garbage Collector (GC)** before destroying an object.
- Rarely used in modern Java (deprecated in Java 9).

Garbage Collection

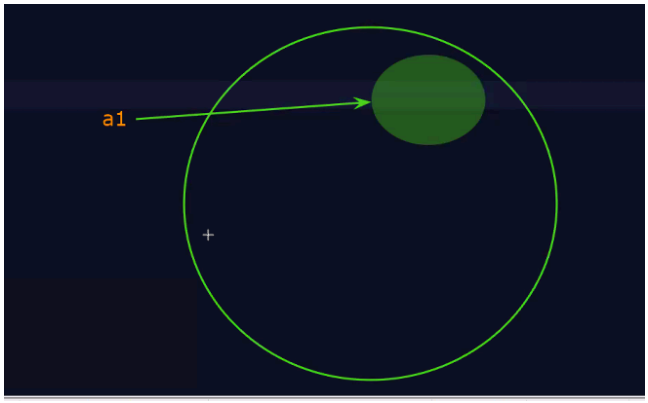
Garbage Collection is the process by which Java automatically identifies and removes objects from memory that are no longer used or reachable by the program, freeing up memory and helping prevent memory leaks.

How Garbage Collection Works

1. **Marking:**
   - GC identifies objects that are no longer reachable from any references in the program.

- Example: If `obj = null;`, then the object previously referenced by `obj` becomes unreachable.

2. **Normal Deletion (Sweeping):**
- The unreachable objects are removed from memory.
3. **Compacting (optional):**
- Some GCs move remaining objects together to reduce fragmentation.



Techniques to Make Objects Eligible for Garbage Collection

1. **Nullifying a Reference Variable**
- Set the reference of an object to `null`, so it becomes unreachable.

```
A a1 = new A();
a1 = null;  // a1 is now eligible for GC
```

2. **Reassigning a Reference Variable**
- Assign a reference to another object; the previous object becomes unreachable.

```
A a1 = new A();
A a2 = new A();
a1 = a2;  // The original object referenced by a1 is now eligible for GC
```

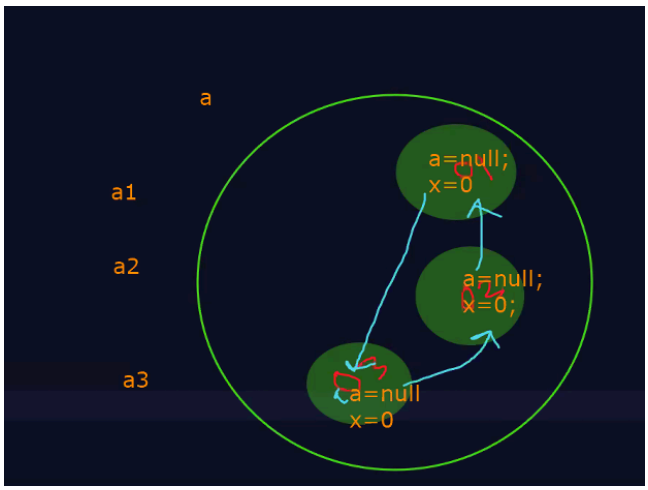
3. **Using Objects in a Limited Scope**
- Objects created inside a method or block become unreachable after the block ends.

```
void method() {
    A a = new A();
} // 'a' goes out of scope here → eligible for GC
```

4. **Dereferencing Objects in Data Structures**
- Remove references from collections (like lists, maps) to make them eligible.

```
List<A> list = new ArrayList<>();
A obj = new A();
list.add(obj);
list.remove(obj); // obj is now eligible for GC
```

5. **Finalization (Optional/Old Technique)**
- Before object destruction, `finalize()` can be called (deprecated in newer Java).

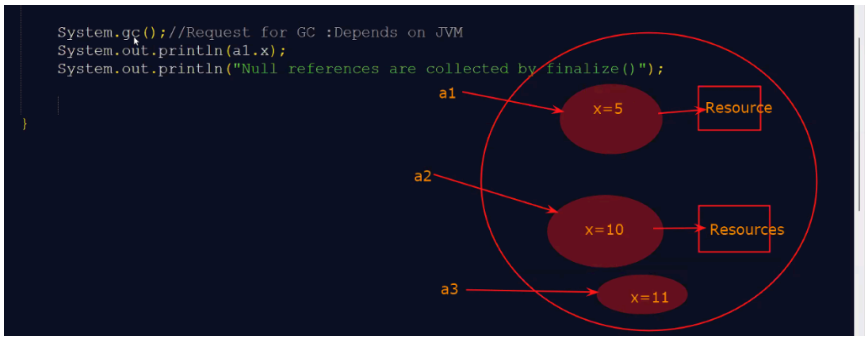


Island of isolation :

In **Java**, an **island of isolation** refers to a **group of objects that are referencing each other but are no longer reachable from any live thread or root objects**.

- Even if objects reference each other, **if no live thread can access them**, they are considered **garbage**.
- GC can detect such groups and reclaim memory.
- These objects form an "island" because they reference each other internally but are **isolated from the rest of the program**.

- a and b reference each other.
- After `a = null; b = null;`, **no live thread can access them**.
- They form an **island of isolation**, eligible for garbage collection.



String :

String in Java

- **Definition:**
  A String is an object that represents a sequence of characters.
- **Class:**
  java.lang.String

Immutability

- Strings are immutable, which means their value cannot be changed once created.

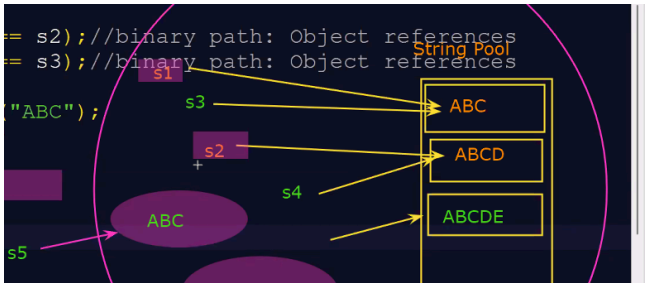
String Pool

A special memory area in Java where string literals are stored.

- **Purpose**: If a string already exists in the pool, Java reuses it instead of creating a new object.

Interfaces Implemented by String

- **Serializable**: Allows strings to be converted into a byte stream for storage or transfer.
- **Comparable**: Allows strings to be compared lexicographically.
- **CharSequence:** Represents a readable sequence of characters.




```
System.out.println(s1 == s2);//binary path: Object ref
System.out.println(s1 == s3);//binary path: Object ref

String s6 = new String("ABC");
String s7 = new String("ABCD");
s5 = new String("ABC");
System.out.println(s1.equals(s3));//Value of reference
```