Memory Management In Java

What They Never Taught You About Java Memory Management

Manav Juneja

Pariभाषा

☕ Think Java handles memory for you? Think again.

🧹 Garbage Collector isn't your maid – and it's not cleaning up all your mess.

💻 Memory leaks in Java are real, and they're sneakier than your ex stalking your GitHub.

🔍 Whether you're a newbie or a seasoned backend beast – if you've ever written a new keyword without thinking... this post is for you.

Manav Juneja

Heap VS Stack

Stack: "I'm short-term. Quick, simple. I handle local stuff – like variables in methods."

Heap: "I'm the long-term commitment. I handle objects. I get garbage-collected, but emotionally? I'm still a mess."

💡 Think of Stack as your WhatsApp chats, and Heap as your photo gallery – heavy, messy, and you never delete anything.

Manav Juneja

Usecases

📦 **Stack Memory:**

- Stores method calls & local variables
- Fast AF
- Auto-cleaned when method ends

🏠 **Heap Memory:**

- Stores objects & class variables
- Slower, but holds the big stuff
- Needs a Garbage Collector to clean up

Manav Juneja

The Garbage Collector

Java’s personal cleaner bot 🤖

It’s like your mom cleaning your room. You didn’t ask for it, but I’m doing it anyway.”

Finds unused objects
Frees up heap space
Works in the background (like anxiety 😅)

Manav Juneja

Memory Leaks

Yes, even Java has a hoarding problem.

📉 Memory leak happens when:

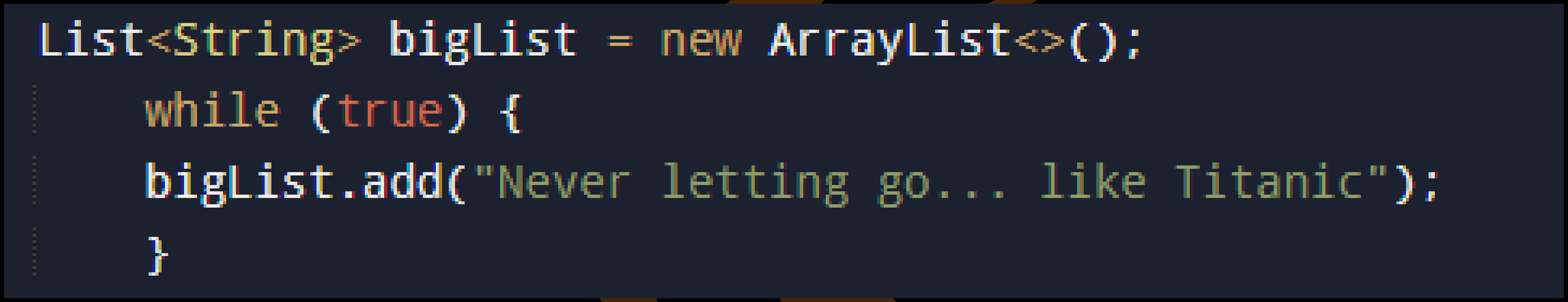
You're done with the object
But references to it are still hanging around
= GC can't clean it 😬

🧠 Imagine your ex still has your Netflix password. You're not together, but they're still draining you. 😭

Manav Juneja

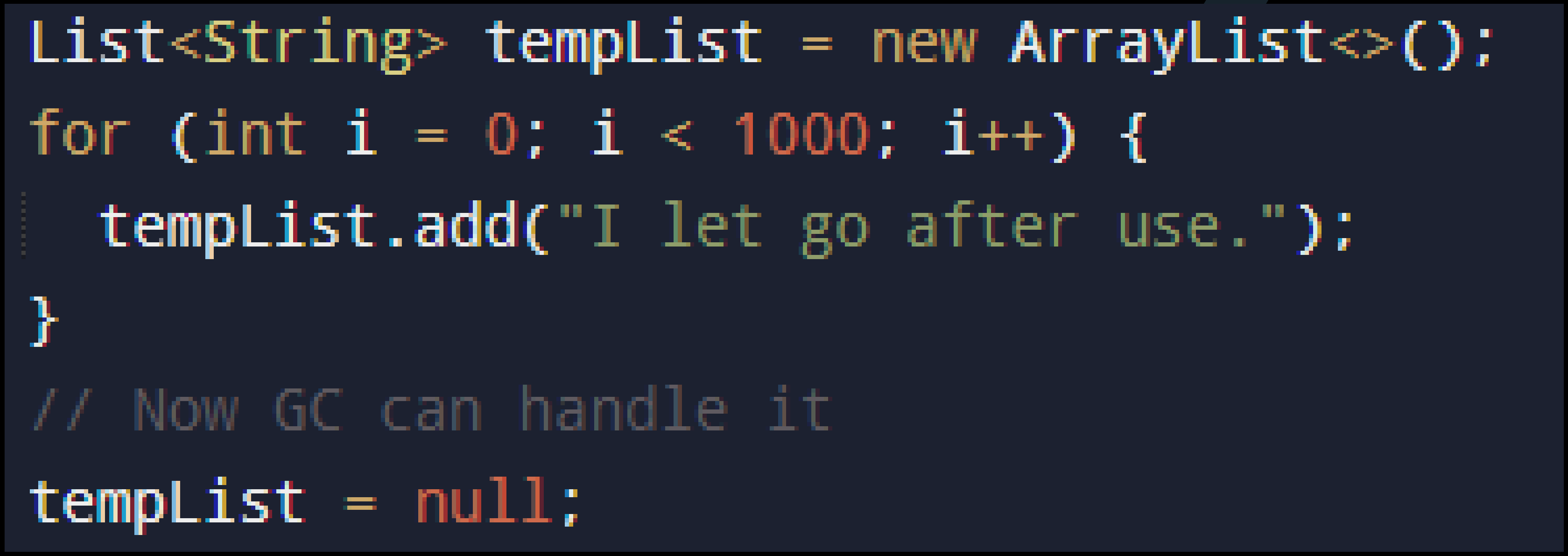
Right & Wrong

❌ **WRONG:**



```java
List<String> bigList = new ArrayList<>();
    while (true) {
    bigList.add("Never letting go... like Titanic");
    }
```

✅ **RIGHT:**



```java
List<String> tempList = new ArrayList<>();
for (int i = 0; i < 1000; i++) {
  tempList.add("I let go after use.");
}
// Now GC can handle it
tempList = null;
```

Manav Juneja

Right & Wrong

❌ **WRONG:**

```java
Map<HeavyObject, String> map = new HashMap<>();
HeavyObject obj = new HeavyObject();
map.put(obj, "still here");
```

✅ **RIGHT:**

```java
Map<HeavyObject, String> map = new WeakHashMap<>();
HeavyObject obj = new HeavyObject();
map.put(obj, "GC can collect this if needed");
```

Manav Juneja

Pro Tips

✅ Remove unnecessary references

✅ Use WeakReference when possible

✅ Don’t hold on to large objects forever

✅ Profile your app (with tools like VisualVM, JConsole)

Manav Juneja

TL:DR

- Stack = short-term, fast, method-level
- Heap = long-term, slow, object-level
- GC = background janitor
- Memory leak = ex with your WiFi password
- Clean up your mess = fewer crashes, more pizza 🍕

Manav Juneja

Follow
For
More

Manav Juneja