Object Pool

Gio Escobar, Dylan DiPesa-Serven

What is an Object Pool?

The Object Pool Pattern is a creational design pattern that aims to reduce the overhead of initializing and destroying objects by recycling objects that are no longer in use. This pattern maintains a pool of initialized objects that are kept ready to be used, rather than creating new objects. When a client has finished using the object, it is returned to the pool, making it available again for reuse rather than being discarded or wiped completely.

IN USE
IN POOL
Total Allocated Goats

Optimizing Performance with the Object Pool Pattern

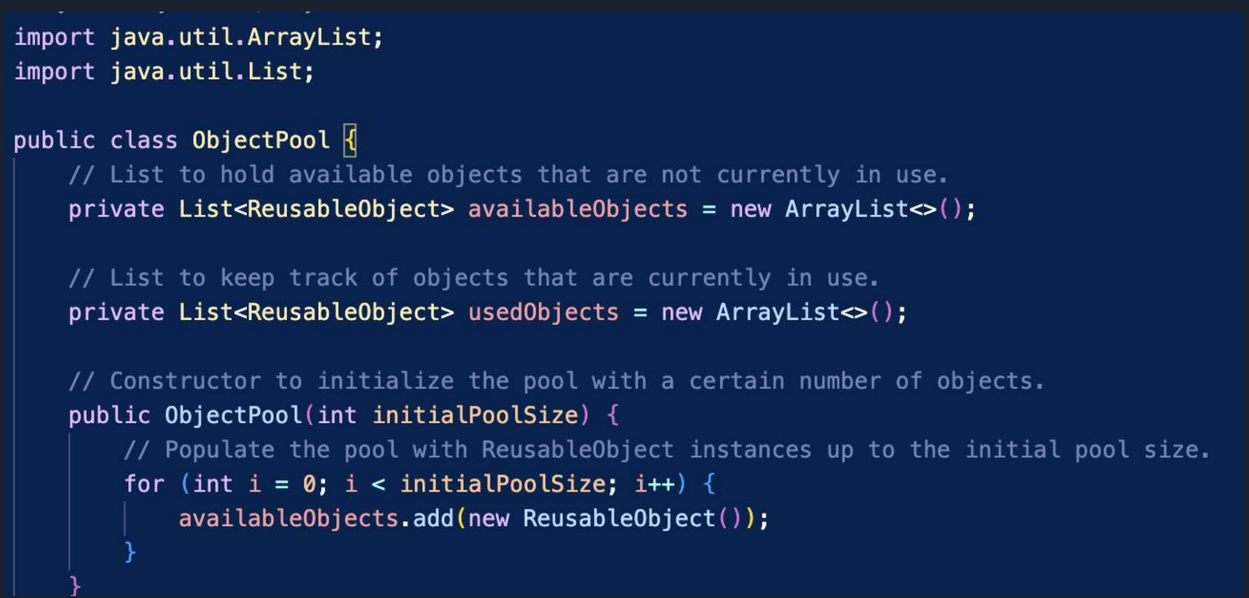
The Object Pool Pattern addresses the performance hit from frequently creating and destroying resource-heavy objects like a database connection. By recycling objects from a pool, it cuts down on the need to generate new instances, reducing memory use and improving application performance like its speed or scalability. This pattern champions resource reuse and efficient lifecycle management. It promotes a sustainable, performance-enhancing design strategy in software development, making it a best practice for managing resource-intensive object creation and destruction.

Design Usage

The object pool design pattern can be used to enhance performance in scenarios such as

- Substantial overhead associated with class instance initialization
- Frequent creation of a class instance
- Temporary requirement for objects
- Continuous cycle of allocating and freeing numerous objects

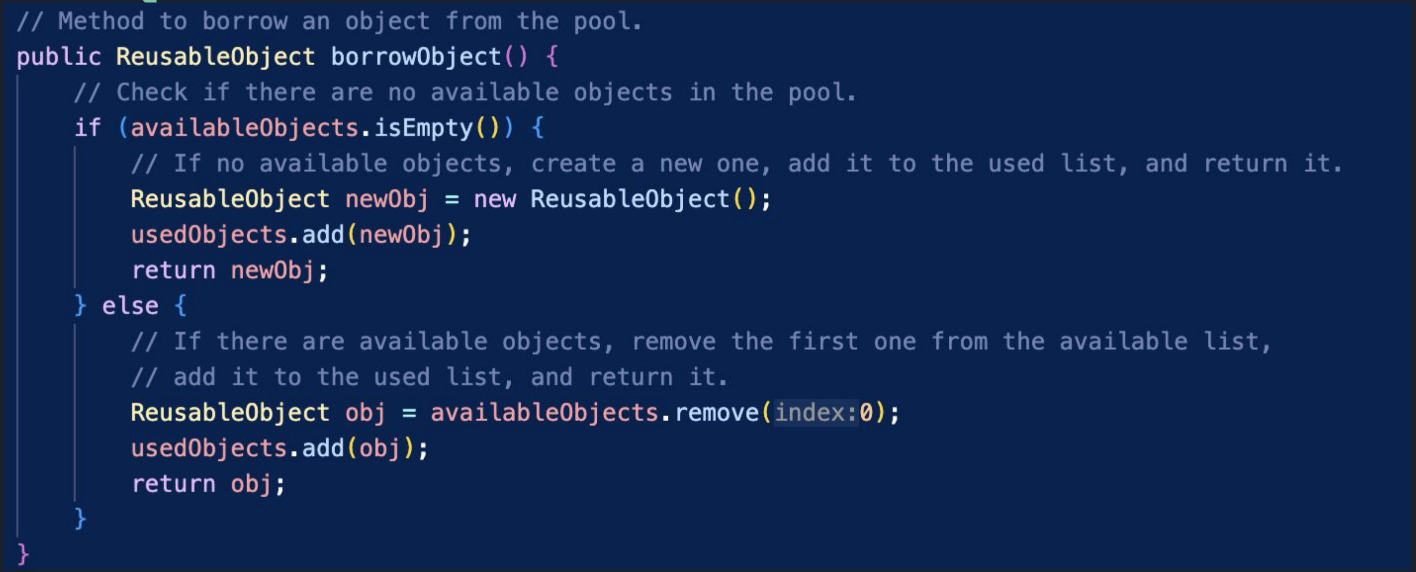
Code Example



```java
import java.util.ArrayList;
import java.util.List;

public class ObjectPool {
    // List to hold available objects that are not currently in use.
    private List<ReusableObject> availableObjects = new ArrayList<>();

    // List to keep track of objects that are currently in use.
    private List<ReusableObject> usedObjects = new ArrayList<>();

    // Constructor to initialize the pool with a certain number of objects.
    public ObjectPool(int initialPoolSize) {
        // Populate the pool with ReusableObject instances up to the initial pool size.
        for (int i = 0; i < initialPoolSize; i++) {
            availableObjects.add(new ReusableObject());
        }
    }
```

Initialize a pool

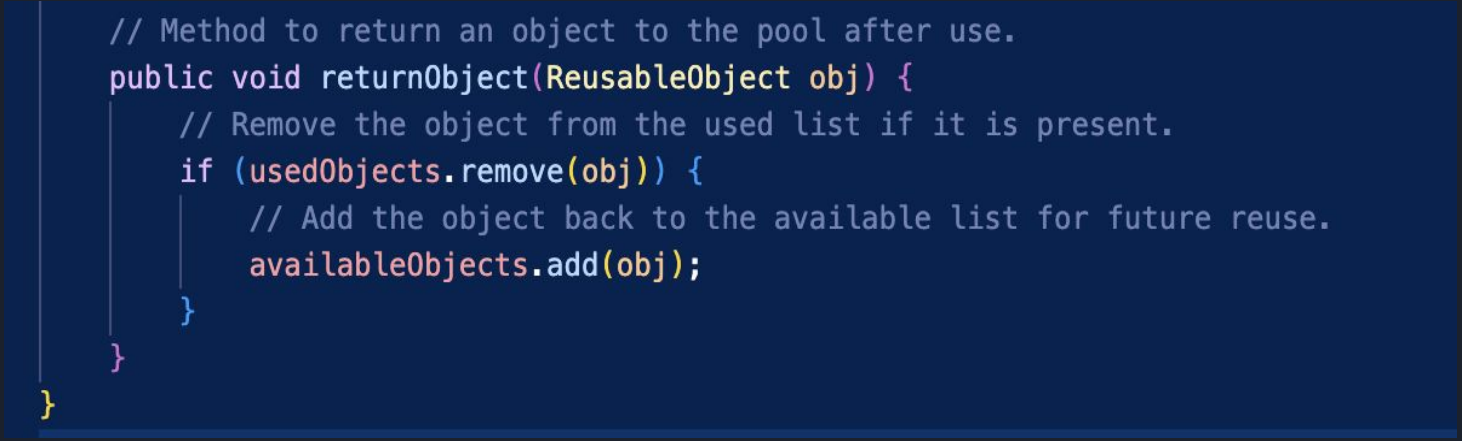
Code example



```java
// Method to borrow an object from the pool.
public ReusableObject borrowObject() {
    // Check if there are no available objects in the pool.
    if (availableObjects.isEmpty()) {
        // If no available objects, create a new one, add it to the used list, and return it.
        ReusableObject newObj = new ReusableObject();
        usedObjects.add(newObj);
        return newObj;
    } else {
        // If there are available objects, remove the first one from the available list,
        // add it to the used list, and return it.
        ReusableObject obj = availableObjects.remove(index:0);
        usedObjects.add(obj);
        return obj;
    }
}
```

Borrow Object

Code example



```
    // Method to return an object to the pool after use.
    public void returnObject(ReusableObject obj) {
        // Remove the object from the used list if it is present.
        if (usedObjects.remove(obj)) {
            // Add the object back to the available list for future reuse.
            availableObjects.add(obj);
        }
    }
}
```

R

Advantages

Benefits of Object Pooling in Resource Management

- Facilitates easier reuse and sharing of objects
- Regulates the simultaneous creation of objects
- Time-Saving in high destruction scenarios
- Decreases reliance on Garbage Collection

Disadvantages

The implementation of Object Pools, while beneficial, can also present challenges when not managed correctly.

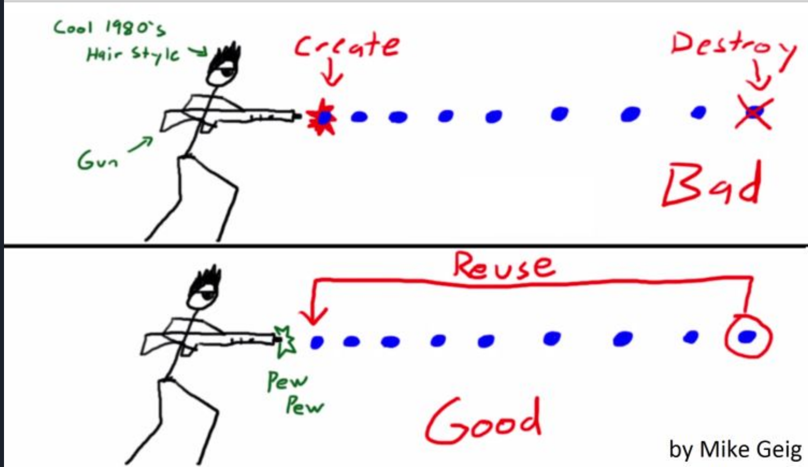
- Objects Must Be Released Post-Use: After an object’s utilization, it is essential to ensure its return to the pool for future reuse.
- Tracking Multi Referenced Objects : Objects that are referenced in multiple places require careful tracking to avoid errors or leaks.
- Pooling Complexity - Managing pools of collections can be tricky since it is not straightforward to pool and utilize in an efficient way.
- Memory Consumption by Idle Objects: Objects that remain idle in the pool continue to occupy memory, which could bring up concern in a resource constrained environment

Real World Uses

- Database Connection
    - Maintaining a pool for user requests

- Video Game design
    - Manage using enemy characters
    - Reusing bullet objects in shooting games
    - Managing special effect particles
        - Explosions
        - Weather Effects
        - Environmental Elements

Visual Example of Object Pooling


Cool 1980's Hair Style
Create
Destroy
Gun
Bad
Reuse
Pew Pew
Good
by Mike Geig