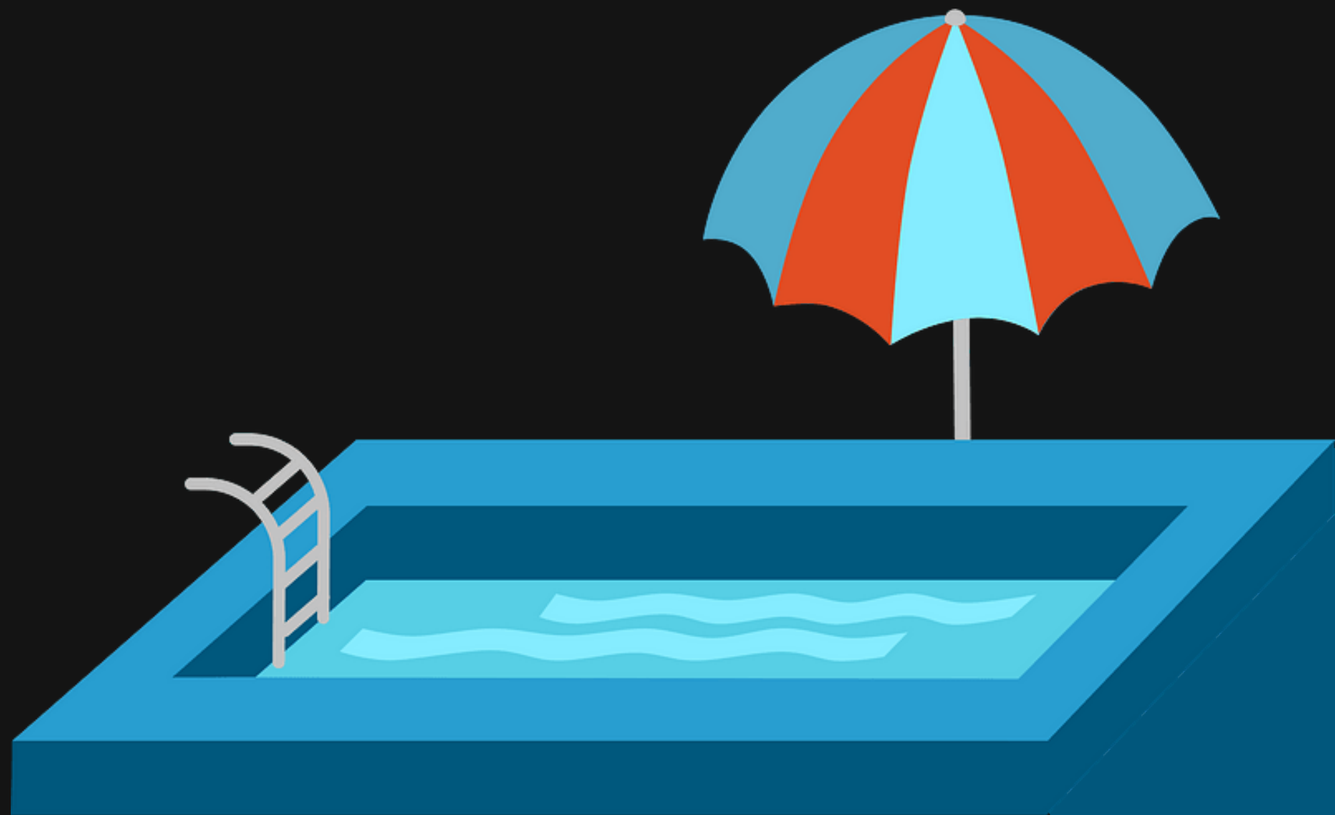
Patrón

Object Pool

de

diseño



- El patrón de diseño "Object Pool" es un patrón de diseño CREACIONAL.
- Se utiliza para mejorar el rendimiento y la eficiencia de las aplicaciones.
- Reutiliza objetos en lugar de crear nuevos cada vez que se necesitan.

El concepto detrás de este patrón es simple: en lugar de crear y destruir objetos constantemente, se mantienen en una "piscina" para su reutilización.

OBJECT POOL

Intención

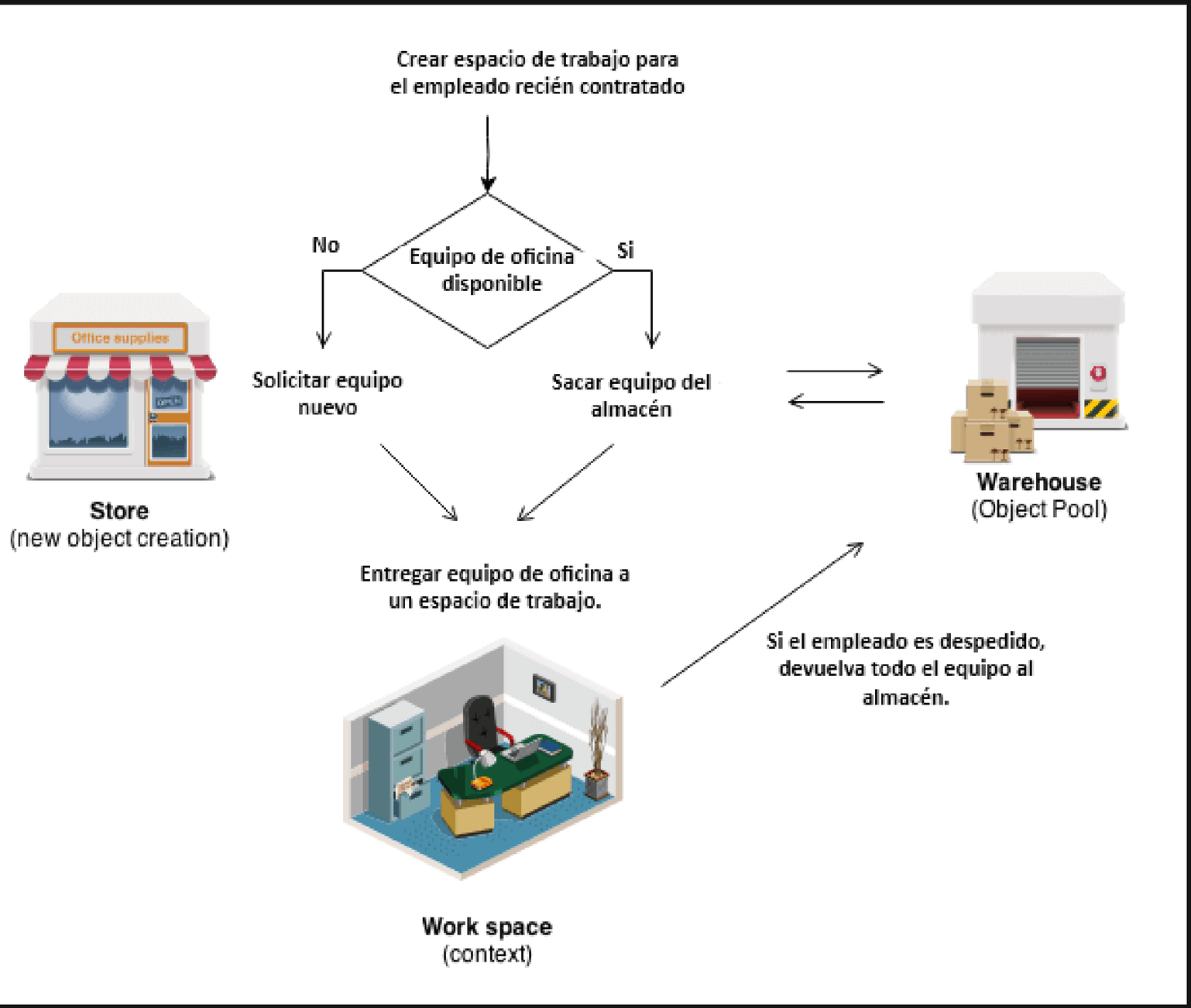
- La agrupación de objetos puede ofrecer un impulso significativo en el rendimiento.
- Es más efectiva en situaciones donde el costo de inicializar una instancia de clase es alto, la tasa de instanciación de una clase es alta y el número de instancias en uso en cualquier momento es bajo.

Problema que resuelve

- Los objects pool (también conocidos como resources pool) se utilizan para gestionar el almacenamiento en caché de objetos. Un cliente con acceso a un object pool puede evitar crear nuevos objetos simplemente pidiendo al pool uno que ya haya sido instanciado. Generalmente, el pool será un pool de crecimiento, es decir, el propio pool creará nuevos objetos si está vacío, o podemos tener un pool que restrinja la cantidad de objetos creados.

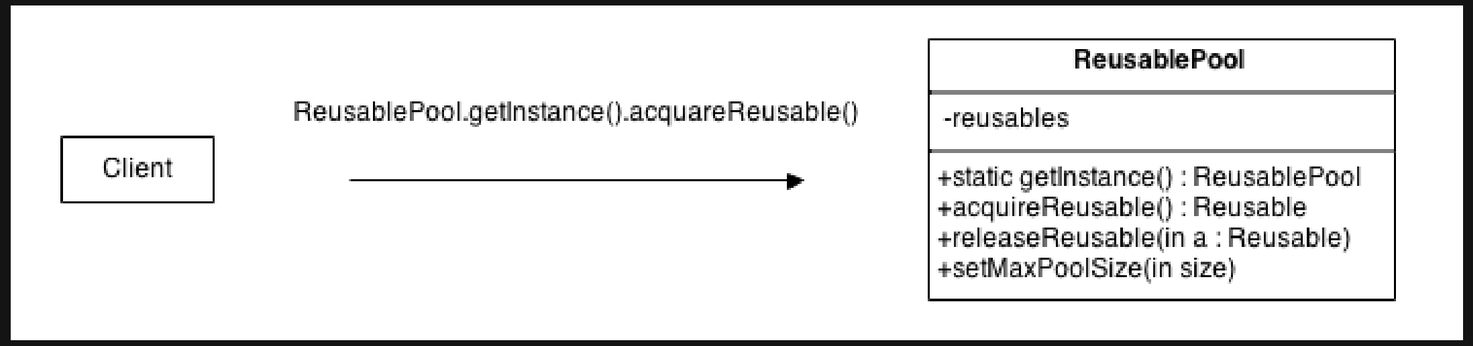
- Es deseable mantener todos los objetos reutilizables que no se están utilizando actualmente en el mismo object pool para que puedan ser gestionados por una política coherente. Para lograr esto, la clase de Grupo Reutilizable está diseñada para ser una clase singleton.

El patrón de object pool es similar a un almacén de oficina. Cuando se contrata a un nuevo empleado, el administrador de la oficina debe preparar un espacio de trabajo para él. Determina si hay equipo de repuesto en el almacén de la oficina. Si lo hay, lo utiliza. Si no, realiza un pedido para comprar nuevo equipo en Amazon. En caso de que un empleado sea despedido, su equipo se traslada al almacén, donde se puede tomar cuando sea necesario para un nuevo lugar de trabajo.



ESTRUCTURA

La idea general del patrón de agrupación de conexiones es que si las instancias de una clase pueden reutilizarse, evitas crear instancias de la clase al reutilizarlas.



POR LO GENERAL, ES DESEABLE MANTENER TODOS LOS OBJETOS REUTILIZABLES QUE NO ESTÁN EN USO EN EL MISMO GRUPO DE OBJETOS PARA QUE PUEDAN SER GESTIONADOS POR UNA POLÍTICA COHERENTE. PARA LOGRAR ESTO, LA CLASE REUSABLEPOOL ESTÁ DISEÑADA COMO UNA CLASE SINGLETON.

REUTILIZABLE

Las instancias de las clases en este papel colaboran con otros objetos durante un tiempo limitado y luego ya no son necesarias para esa colaboración.

CLIENTE

Las instancias de las clases en este papel utilizan objetos Reutilizables.

PISCINA REUTILIZABLE

Las instancias de las clases en este papel gestionan objetos Reutilizables para su uso por objetos Cliente.

Problema

En una aplicación de servidor que maneja conexiones de red, como un servidor de chat en tiempo real, se enfrenta al problema de la creación y destrucción continua de objetos de conexión para atender a los clientes entrantes. La creación y destrucción de objetos de conexión es costosa en términos de recursos y puede afectar negativamente el rendimiento del servidor.

CREACIÓN

Se crea una piscina de objetos de conexión al inicio del servidor con un número predeterminado de conexiones.

VERIFICAR

Al recibir una solicitud de un cliente, en lugar de crear una nueva conexión, el servidor verifica si hay una conexión disponible en la piscina.

TOMAR

Si hay una conexión disponible en la piscina, se toma de la piscina y se utiliza para atender al cliente. Esto evita la creación de una nueva conexión.

DEVOLVER

Una vez que la solicitud del cliente se completa, la conexión se devuelve a la piscina en lugar de ser destruida. La conexión se reinicia y se vuelve a usar para futuras solicitudes de clientes.

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

// Objeto de conexión
class Connection {
    private boolean inUse;

    public Connection() {
        inUse = false;
        // Realizar inicialización costosa, si es necesario
    }

    public void setInUse(boolean inUse) {
        this.inUse = inUse;
    }

    public boolean isInUse() {
        return inUse;
    }
}

// Piscina de conexiones
class ConnectionPool {
    private static final int MAX_CONNECTIONS = 10;
    private List<Connection> connections = new ArrayList<>();

    public ConnectionPool() {
        for (int i = 0; i < MAX_CONNECTIONS; i++) {
            connections.add(new Connection());
        }
    }

    public Connection getConnection() {
        for (Connection connection : connections) {
            if (!connection.isInUse()) {
                connection.setInUse(true);
                return connection;
            }
        }
        return null; // No hay conexiones disponibles
    }
```