Object Pool

Introduction

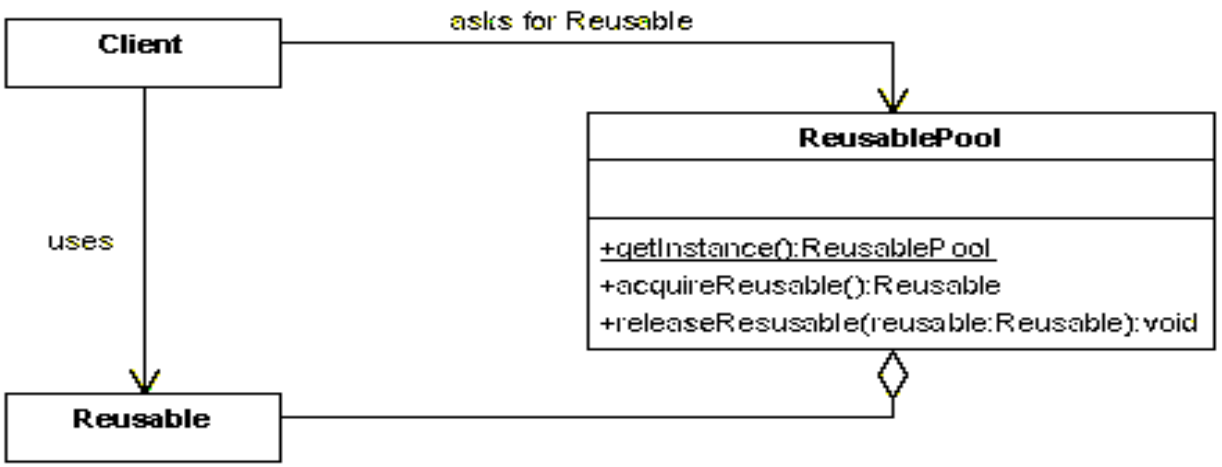
- A pool is a collection of resources (system resources) that are kept ready to use.
- The Object pool design pattern is a creational design pattern which uses a pool of initialized objects as ready to use.
- The resources i.e. objects can be allocated and de-allocated on demand.

Why ?

- Object Pool design pattern will be used when there are several clients who needs the same stateless resource which is expensive to create.

Implementation

- Reusable - Wraps the limited resource, will be shared by several clients for a limited amount of time.
- Client - uses an instance of type Reusable.
- ReusablePool - manage the reusable objects for use by Clients, creating and managing a pool of objects.



Contd.

- When a client asks for a Reusable object, the pool performs the following actions:
  - Search for an available Reusable object and if it was found it will be returned to the client.
  - If no Reusable object was found then it tries to create a new one. If this actions succeeds the new Reusable object will be returned to the client.
  - If the pool was unable to create a new Reusable, the pool will wait until a reusable object will be released

Pros & Cons

- Singleton reusable pool
- *Limited number of resources in the pool*
- *Handling situations when creating a new resource fails*
- *Synchronization*
- *Expired resources(unused but still reserved)*
- Creating the resources might fail and this case should be treated carefully. When there is no available resource(because the number is limited or creating a new one failed) the client should be notified about it.
- Although the object pool is handling the object instantiation it's main purpose is to provide a way for the clients to reuse the objects like they are new objects, without being shared and reused.