DESIGN PATTERN - SERVICE LOCATOR PATTERN

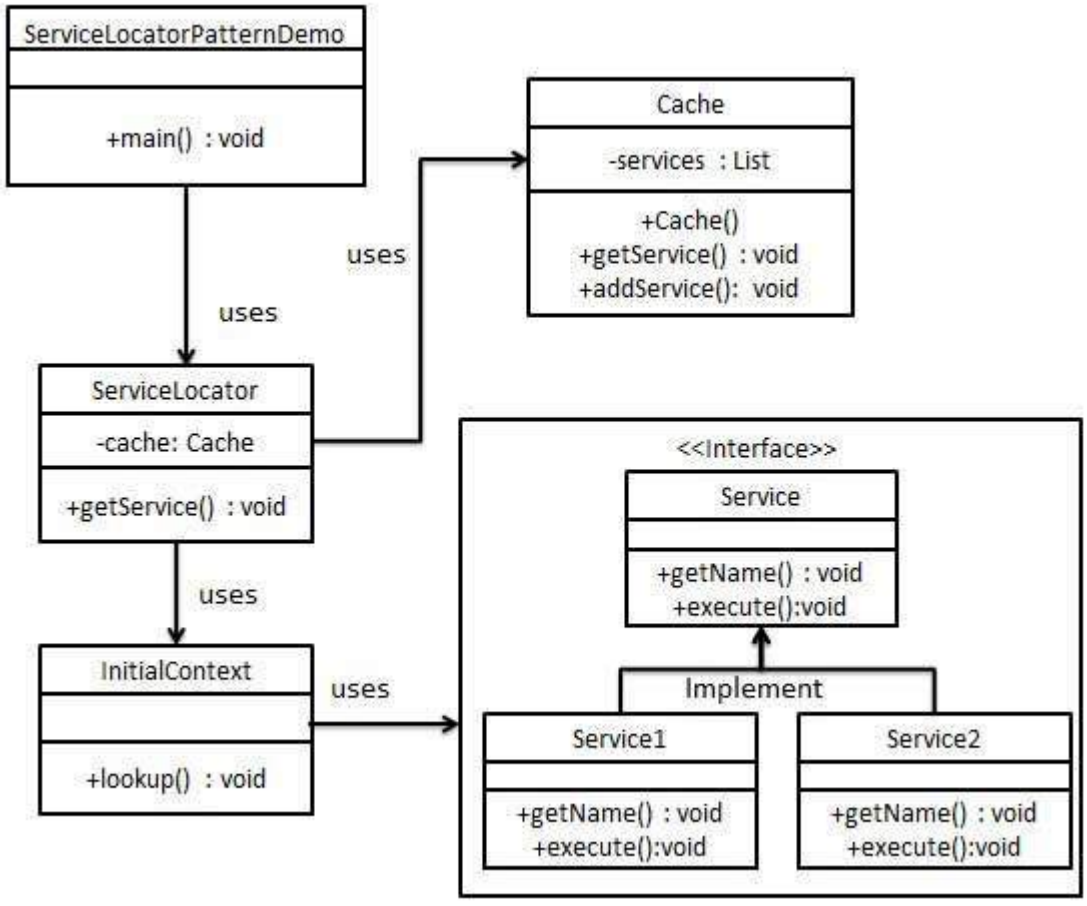
The service locator design pattern is used when we want to locate various services using JNDI lookup. Considering high cost of looking up JNDI for a service, Service Locator pattern makes use of caching technique. For the first time a service is required, Service Locator looks up in JNDI and caches the service object. Further lookup or same service via Service Locator is done in its cache which improves the performance of application to great extent. Following are the entities of this type of design pattern.

- **Service** - Actual Service which will process the request. Reference of such service is to be looked upon in JNDI server.
- **Context / Initial Context** - JNDI Context carries the reference to service used for lookup purpose.
- **Service Locator** - Service Locator is a single point of contact to get services by JNDI lookup caching the services.
- **Cache** - Cache to store references of services to reuse them
- **Client** - Client is the object that invokes the services via ServiceLocator.

Implementation

We are going to create a *ServiceLocator*,*InitialContext*, *Cache*, *Service* as various objects representing our entities.*Service1* and *Service2* represent concrete services.

ServiceLocatorPatternDemo, our demo class, is acting as a client here and will use *ServiceLocator* to demonstrate Service Locator Design Pattern.



Step 1

Create Service interface.

Service.java

```
public interface Service {
   public String getName();
   public void execute();
}
```

Step 2

Create concrete services.

Service1.java

```
public class Service1 implements Service {
   public void execute(){
      System.out.println("Executing Service1");
   }

   @Override
   public String getName() {
      return "Service1";
   }
}
```

Service2.java

```
public class Service2 implements Service {
   public void execute(){
      System.out.println("Executing Service2");
   }

   @Override
   public String getName() {
      return "Service2";
   }
}
```

Step 3

Create InitialContext for JNDI lookup

InitialContext.java

```
public class InitialContext {
   public Object lookup(String jndiName){

      if(jndiName.equalsIgnoreCase("SERVICE1")){
         System.out.println("Looking up and creating a new Service1 object");
         return new Service1();
      }
      else if (jndiName.equalsIgnoreCase("SERVICE2")){
         System.out.println("Looking up and creating a new Service2 object");
         return new Service2();
      }
      return null;
   }
}
```

Step 4

Create Cache

Cache.java

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

```
import java.util.List;

public class Cache {

   private List<Service> services;

   public Cache(){
      services = new ArrayList<Service>();
   }

   public Service getService(String serviceName){

      for (Service service : services) {
         if(service.getName().equalsIgnoreCase(serviceName)){
            System.out.println("Returning cached  " + serviceName + " object");
            return service;
         }
      }
      return null;
   }

   public void addService(Service newService){
      boolean exists = false;

      for (Service service : services) {
         if(service.getName().equalsIgnoreCase(newService.getName())){
            exists = true;
         }
      }
      if(!exists){
         services.add(newService);
      }
   }
}
```

Step 5

Create Service Locator

ServiceLocator.java

```
public class ServiceLocator {
   private static Cache cache;

   static {
      cache = new Cache();
   }

   public static Service getService(String jndiName){

      Service service = cache.getService(jndiName);

      if(service != null){
         return service;
      }

      InitialContext context = new InitialContext();
      Service service1 = (Service)context.lookup(jndiName);
      cache.addService(service1);
      return service1;
   }
}
```

Step 6

Use the *ServiceLocator* to demonstrate Service Locator Design Pattern.

ServiceLocatorPatternDemo.java

```
public class ServiceLocatorPatternDemo {
   public static void main(String[] args) {
      Service service = ServiceLocator.getService("Service1");
      service.execute();
      service = ServiceLocator.getService("Service2");
      service.execute();
      service = ServiceLocator.getService("Service1");
      service.execute();
      service = ServiceLocator.getService("Service2");
      service.execute();
   }
}
```

Step 7

Verify the output.

```
Looking up and creating a new Service1 object
Executing Service1
Looking up and creating a new Service2 object
Executing Service2
Returning cached  Service1 object
Executing Service1
Returning cached  Service2 object
Executing Service2
```