AMPLIACIÓ DE BASES DE DADES I ENGINYERIA DEL PROGRAMARI

Activitat 3: *Exercici de disseny*

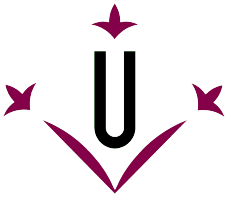
Maria Florencia Martínez Malaret

Jordi Rafael Lazo Florensa

Pere Rollón Baiges

23 Maig de 2021

Grau en Enginyeria Informàtica



Universitat de Lleida
Escola Politècnica Superior

1 Introducció

En aquesta pràctica es crea una infraestructura per facilitar la interconnexió de les instàncies de diferents classes fent ús del patró *Service Locator*.

Aquest patró de disseny s'utilitza en el desenvolupament de programari per encapsular els processos involucrats en l'obtenció d'un servei amb una capa d'abstracció. Aquest patró utilitza un registre central conegut com el *Service Locator*, que a petició retorna la informació necessària per realitzar una determinada tasca.

2 Implementació

2.1 Disseny del patró

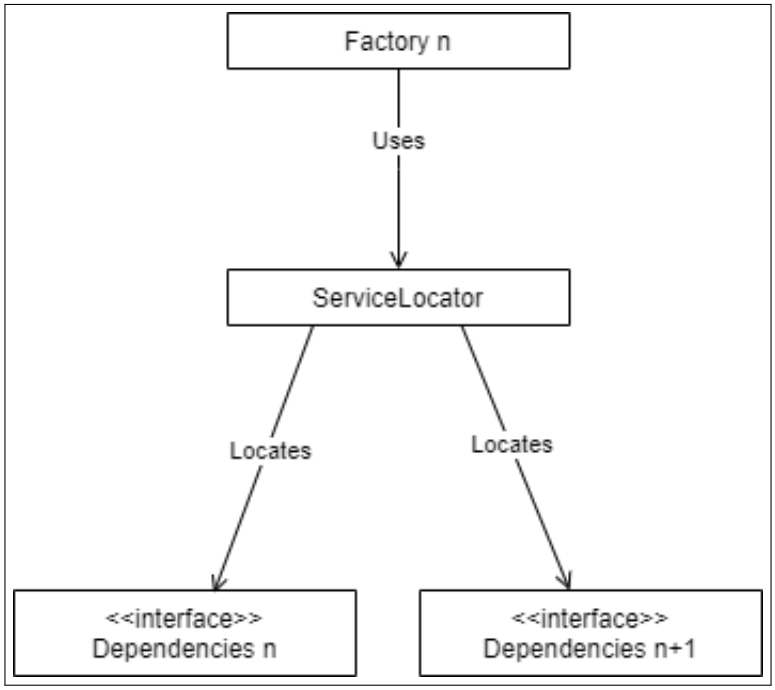


Figure 1: Patró *Service Locator* implementat en la pràctica.

2.2 *CachedServiceLocator*

- *CachedServiceLocator* (primera versió): en aquesta classe s'ha decidit implementar un *HashMap* on la seva clau és un *string* de cadascuna de les interfícies implementades i el seu valor és un *array* on s'emmagatzema la constant que s'associï a la factoria.

  Pel que fa a la funció *setService* abans d'instal·lar una factoria primer es comprova que no hi hagi una emmagatzemada amb el mateix nom dintre de la instància del *HashMap*. En cas que no hi hagi s'emmagatzemarà el nom i la factoria, en cas contrari és llançarà l'excepció *LocatorError*.

  Pel que fa a *setConstants* s'ha implementat una solució idèntica a la funció *setService* amb la diferència que s'emmagatzema una variable de tipus *Object* (una implementació) dintre del *HashMap*.

Finalment en la funció *getObject* abans d'obtenir la constant associada es comprova si aquesta es *null* o no. En el cas que no fos *null* es retorna la seva constant, en cas contrari es crea la constant a partir de la factoria associada i s'emmagatzema dins del *HashMap*.

- *CachedServiceLocator* (segona versió): pel que fa a la segona versió s'ha utilitzat la mateixa estructura que la primera amb la diferència de la introducció de mètodes genèrics i la utilització d'un segon *HashMap* en comptes d'un *array*. Això és pel fet que en els tres mètodes implementats, ja no es passa per paràmetres un *string* sinó classes i aquestes s'han d'emmagatzemar i després de realitzar insatisfactoriament les proves d'execució ens vam adonar compte que no es pot crear un *array* genèric perquè causava un error en temps de compilació. Això és pel fet que *Factory* al ser d'un tipus genèric, per introduir-ho en un *array*, s'ha d'haver declarat amb anterioritat el seu tipus concret. Aquest error va ser resolt gràcies a revisar la documentació oficial de JAVA sobre com funcionen exactament els objectes genèrics. Després d'aquest canvi ens vam donar compte que en la primera versió també es podien utilitzar dos *HashMap* però vam decidir no modificar-ho per entendre les diferencies.

2.3 *SimpleServiceLocator*

- *SimpleServiceLocator* (primera versió): per la implementació d'aquesta classe s'ha optat per un disseny pràcticament igual que al *CachedServiceLocator* amb la petita diferencia que en la funció *getObject* en comptes de retornarà el mateix objecte ara és retornarà un nou objecte que crea la factoria però sense emmagatzemar-lo i si es tornés a cridar-lo és crearia un altre nou objecte i així successivament.

- *SimpleServiceLocator* (segona versió): per aquesta versió s'ha utilitzat un disseny pràcticament idèntic que la primera versió amb la diferència que ara ha sigut necessari utilitzar dos diccionaris per separar les factories de les constants (a causa del mateix motiu explicat en la segona versió de *CachedServiceLocator*) .

3 Tests

Pel que fa als test s'han implementat 4 *mocks* diferents per tal de poder testejar el funcionament de *ServiceLocator*, tant en la primera com en la segona versió. Cadascun d'aquests *mocks* compta amb la seva interfície, implementació i amb les seves factories per testejar els 4 tests. Això ha sigut necessari, ja que les factories de la segona versió afegeixen els mètodes genèrics.

- Tests primera versió: pel que fa al test de la primera versió s'han testejat els mètodes *set* i *get* en un test per comprovar que llancen un error i s'han introduït els objectes correctament i en un altre test s'ha comprovat que es retornin satisfactòriament els objectes definits.

- Tests segona versió: pel que fa al test de *CachedServiceLocator* s'ha comprovat els errors llançats per la classe *LocatorError* siguin correctes així com pels mètodes *setService* i *setConstant*. Respecte al test *SimpleServiceLocator* s'ha realitzat amb les classes *FactoryA1v2* i *FactoryD1v2* i s'han comprovat que no son iguals les factories definides que les noves introduides.