Laboration del 2 – Objektorienterad programmering

Beskrivning – del 2

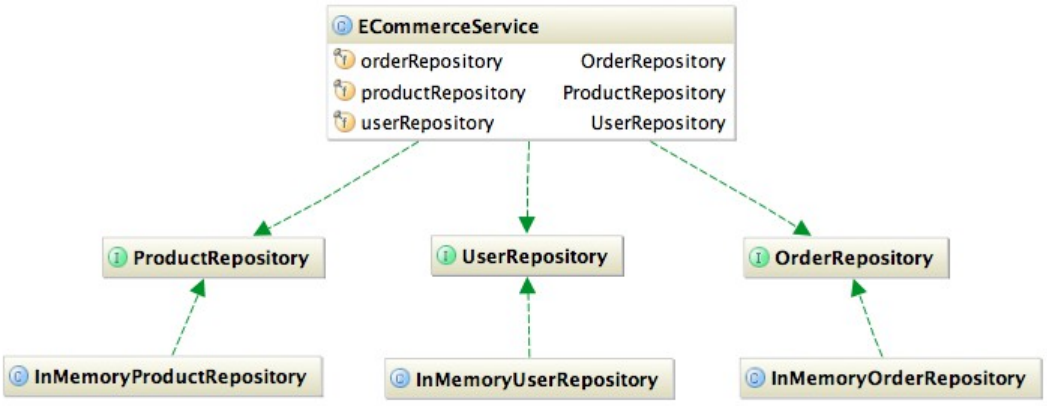
I denna del av laborationen kommer ni att skapa gränssnittet för att bl.a. kunna lagra, hämta och uppdatera exempelvis *User, Product, Order, osv* som ni använder er av i din webbshopp. Lagringen ni använder er av i detta skede kommer inte att erbjuda någon bestående lagringsmöjlighet. En sådan kommer du att skapa längre fram. Nu kommer lagring att ske temporärt i en vanlig Java-klass. För att övergången till en annan typ av lagring skall ske smidigt kommer du att göra ett eller flera interfaces och sedan skapa klasser som implementerar dessa. Detta gör att du med enkelhet kan byta lagringsmetod.

Designförslag

Skapa ett eller flera interface som exponerar de metoder som erbjuds för att kunna utföra önskade operationer med dina *User, Product, Order, osv*. Metoder som kan tänkas erbjudas är:

- Spara, hämta, uppdatera och ta bort en *User*
- Spara, hämta, hämta alla, uppdatera och ta bort en *Product*
- Skapa, hämta, och hämta alla *Order*

Vilka metoder ni erbjuder är beroende av designen på din webbshop. Det viktiga är att ni exponerar metoder som gör det möjligt att kunna hantera de objekt som ni använder er av. Så här skulle det kunna se ut:



I klassdiagramet ovan används en *ECommerceService* som har till uppgift att exponera alla metoder för att kunna hantera en webbshop. Den i sin tur innehåller tre stycken olika s.k. *Repositories* som den delegerar anropen till. Detta gör den efter det att den eventuellt har utfört någon egen logik. Dessa repositories sparas som datamedlemmar av interface-typ i ECommerceService. De konkreta implementationerna tas in genom konstruktorn till ECommerceService när den skapas. Detta gör att det med lätthet går att byta till andra repositories som implementerar något av *Repository*-interfacen. Om *ECommerceService* hade haft ett beroende av en konkret implementation skulle detta inte gå utan att ändra i ECommerceService. I diagrammet ovan implementeras dessa interface av s.k. InMemory-klasser. Dessa klasser används just för att lagra data i minnet. Dessa lagrar objekten i en List, Set, Map eller liknande istället för i en riktig databas. När vi senare vill arbeta mot en databas gör vi klasser som implementerar de olika repository-interfacen men som då arbetar direkt mot en databas. Dessa skulle då förslagsvis kunna heta (om vi arbetar mot en MySQL-databas) *MySQLUserRepository, MySQLOrderRepository, MySQLProductRepository.* Som sagt detta är bara ett förslag och ni är helt fri att använda dig av er egna design.

Till sist

Denna del av projektet kan te sig lite diffus så i korthet skulle man kunna beskriva den som ett lager som arbetar mellan lagringen av din data (*datalager*) och lagret som presenterar din data (*vylager*). Du kommer senare att skapa ett webbgränssnitt som kommer att arbeta mot detta lager. Laboration 2 ska redovisas **när ni är tillbaka i januari**.