OMD – Kort sammanfattning

# Designprinciper

* **ALP - Lokalitetsprincipen** – Hantera funktionalitet så ”långt ner” som möjligt, Ta bort onödiga getters.
* **SRP - Single Responsibility Principle** – En klass har ett ansvarsområde.
* **OCP - Open/Closed Principle** – Det skall vara möjligt att lägga till ny funktionalitet utan att modifiera existerande kod.
* **Integritetsprincipen** – Gör attribut, metoder och klasser så hemliga de går. Lämna inte ut representationen i onödan. En klass skall inte ha tillgång till klasser som den inte behöver.
* **LSP - Liskov Substitution Principle** – Subklasser ska kunna ersätta sina basklasser.
* **ISP - Interface Segregation Principle** – Klasser bör inte tvingas att bero på metoder som de inte använder.
* **DIP - Dependency Inversion Principle (Abstraktionsprincipen)** – Klasser av hög nivå bör inte vara beroende av klasserna på låg nivå; båda bör bero på abstraktioner. Abstraktioner bör inte vara beroende av detaljer. detaljer bör bero på abstraktioner.

# Stabilitetsprinciper

* **Acyclic dependencies principle (ADP)** – Paket skall ej vara cykliskt beroende.
* **Stable dependencies principle (SDP)** – Beroenden skall gå mot stabila paket.
* **Stable abstractions principle (SAP)** – Stabila paket skall vara abstrakta.

# Designmönster

* **Command [Computer]** – Man delegerar arbetet till klasser med enkelt ansvar.
* **Composite [XL]** – implementerar sin <Basklass>, med en lista med typen <Basklass>.
* **Template Method [Computer]** – Bryt ut det som är gemensamt och placera det i en abstrakt superklass.
* **Strategy [Computer, XL]** – Normalt är strategin ett attribut i klassen, men man kan i stället skicka med den som ett argument till en metod som behöver den. ex skriva ut listor med olika prefix.
* **Decorator** – Lägga på ny funktionalitet som utvidgar grundklassen som inparameter.
* **Singleton** - Att det bara skall kunna skapas en instans av klass.
* **Null Object [XL?]** – Programkoden blir enklare om man istället använder ett riktigt objekt för *null*. Detta är ett exempel på mönstret Null Object.
* **Observer [XL]** – Observer-mönstret används för att separera ”modellen” från användargränssnittet, ”vyn”. Vyn implementerar gränssnittet Observer. Modellen utvidgar klassen Observable.
* **Factory Method [XL]** – vill undvika beroende av konkreta klasser eller måste avgöra vilken sorts objekt som skall skapas vid exekveringen.

# UML och sekvensdiagram

* Objektdiagram
* Sekvensdiagram