**Design Pattern**

Ons design pattern bestaat uit een klasse en 2 Interfaces:

*Een ObserverInterface*

*Een PublisherInterface.*

*De Publisher Klasse*

**Publisherinterface**

De eerste interface is de PublisherInterface hierin zitten 2 abstracte methodes: Register en Unregister. Deze 2 methodes moeten beiden een Object meekrijgen die de Interface ObserverInterface implementeert.

**Observerinterface**

De tweede interface is de ObserverInterface die een abstracte methode Update heeft. Elke klasse die deze interface implementeert, zal deze methode dus ook moeten aanmaken.

**Publisher klasse**

De Publisher klasse implementeert de PublisherInterface en moet dus die 2 methodes (Register en Unregister) ook in zijn klasse aanmaken. De Publisher klasse implementeert niet enkel de PublisherInterface, het erft ook over van de TimerTask klasse. In de Publisher klasse kan men dan over de Run methode van de TimerTask Klasse schrijven.

In de Publisher Klasse wordt er een Arraylist aangemaakt van Objecten die de ObserverInterface implementeren. Met de klasse Register kan je er één toevoegen, met de methode Unregister verwijder je er een. De Publisher zal in de Run methode (overgeschreven van de TimerTask) door deze Arraylist lopen met een For-lus. Per object zal hij dan de Update methode van dit object uitvoeren.

In de HelloXlet klasse kan er dan een Publisher worden aangemaakt en een Timer om aan te geven hoe snel de scene moet updaten.

**Voorbeeld:**

Ik wil een slang hebben in mijn spel Snake. In ons project wordt er een klasse aangemaakt genaamd: SnakeChain. Deze snakeChain zal moeten updaten per timertik, dus we moeten het op een manier kunnen laten updaten in een Run methode van een Timertask.

Ik implementeer de ObserverInterface bij de SnakeChain klasse en geef het dus een Update methode. In deze Update methode schrijf ik wat er moet gebeuren per Timertik met deze slang.

In de HelloXlet klasse maak ik een Publisher aan en een Timer. Ik geef mijn SnakeChain mee aan de Publisher aan de hand van de Register methode van de Publisher.

Nu zal de Publisher in zijn ArralList een Object hebben dat de ObserverInterface Implementeert (de SnakeChain) en dus een Update methode heeft. De Publisher zal deze Update methode uitvoeren per Timertik.