**Prototype, Facade (Façade) e Adapter.**

**Introdução**

Os padrões de designers são algoritmos bem parecidos uns com os outros para se desenvolverem. Um designer de padrão fornece estrutura reutilizáveis ou serve para solucionar um problema comum de designer.

**Prototype**

**Descrição**

Especificar tipos de objetos a serem criados usando uma instancia protótipo e novos objetos são criados a partir da cópia desse.

Pela intenção podemos perceber como o padrão vai resolver o problema. Precisamos criar novos objetos a partir de uma instância protótipo, que vai realizar uma cópia de si mesmo e retornar para o novo objeto.

O padrão Prototype e padrão Factory são bem parecidos em suas intenções e funcionalidades. Ambos são padrões criados que irão criar objetos a partir de uma interface sem precisar saber os tipos de classes subjacentes.

A principal diferença entre os dois padrões envolve como os objetos são construídos. O padrão Factory, generaliza, constrói um objeto usando os mesmos parâmetros do construtor. Todo objeto irá ser inicializado com o mesmo estado e informação e ser mais ou menos equivalente um com o outro.

Ex.:

public void factoryFazAlgo(Factory fabrica) {

Point pnt = factory.createPoint();

... Faz algo com o point ...

}

public void prototypeFazAlgo(Point prototype) {

Point pnt = (Point) prototype.clone();

... faz algo com o ponto ...

}

Nota-se que o método no factoryFazAlgo, o ponto que cria é inicializado do mesmo modo para cada e não pode ser customizado. O método prototypeFazAlgo pode criar um ponto de qualquer ponto com qualquer tipo de estado que lhe é atribuído. Podemos dizer que se instarciarmos um "new Point(23,85)" ou "new Point(2929,59483)" os clones terão similares estados, porém podemos customiza-los o estados dos objetos que irão ser criados.