

Il pattern factory ci permette di avere l'istanza giusta tra un insieme di classi provenienti dalla stessa classe padre a partire da una condizione di selezione.

Più in dettaglio possiamo dire che date una serie di classi F_1, F_2, \dots, F_n derivanti dalla stessa classe o interfaccia P , il pattern factory prevede una ulteriore classe con un eventuale metodo statico che in base ad un parametro di ingresso istanzia una delle n classi figlie. La classe istanziata sarà dunque scelta a runtime sfruttando il polimorfismo.

Vediamo un esempio pratico:

supponiamo di avere una classe FiatPunto:

```
public class FiatPunto{  
  
    public abstract void stampaAccessori(){  
  
        .....  
  
        // altri metodi  
  
}
```

Da questa classe deriviamone altre tre: FiatPuntoAllestimentoBase, FiatPuntoAllestimentoMedio e FiatPuntoAllestimentoTop

```
public class FiatPuntoAllestimentoBase extends FiatPunto {  
  
    public void stampaAccessori(){  
  
        System.out.println("Sterzo, ruote, sedili");  
  
    }  
  
    .....  
  
    // altri metodi  
  
}
```

Creiamo ora la classe che fa da factory

```
public class FactoryFiat{

    public static FiatPunto get(int prezzo){

        if(prezzo < 10.000)

            return new FiatPuntoAllestimentoBase();

        else if(prezzo >= 10.000 && < 12.0000)

            return new FiatPuntoAllestimentoMedio();

        else

            return new FiatPuntoAllestimentoTop ();

    }

}
```

Il parametro di selezione nel nostro esempio è il prezzo. Quindi utilizzando le classi precedenti:

```
public class ComprAuto{

    int prezzo = 0;

    public ComprAuto (int prezzo){

        this.prezzo = prezzo;

    }

    // altri metodi
```

```
public void acquista(){  
    .....  
    FiatPunto auto = FactoryFiat.get(prezzo);  
    auto.stampaAccessori();  
    .....  
}
```

```
public static void main(String[] arg){  
    ComproAuto auto = new ComproAuto(11.000);  
  
    // verranno stampanti gli accessori dell'allestimento medio  
    auto.acquista ();  
}  
  
}
```

Durante l'invocazione del metodo `stampaAccessori`, metodo comune a tutte le classi figlie con implementazione diversa, verranno stampati gli accessori relativi alla classe restituita dalla factory anche se l'istanza ottenuta è quella della classe padre.

Marcello Pirazzoli