**Refactoring**

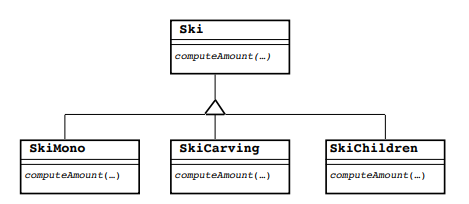
Il codice deve essere strutturato per essere riutilizzato in altri contesti.

Evitare di ripetere pezzi di codice, come switch identici in più parti del codice, usare una soluzione unica.

**Pattern Strategy**

Se si ha una distinzione di tipo, con switch, si potrebbe creare una sotto-classe per ogni sottotipo che ereditano la classe padre, o interfaccia, ed eseguono ciò che devono in base al tipo che rappresentano.

***Strategy usa un’interfaccia o classe astratta comune la quale viene implementata dai figli, così da avere la distinzione di tipo sfruttando la OOP.***

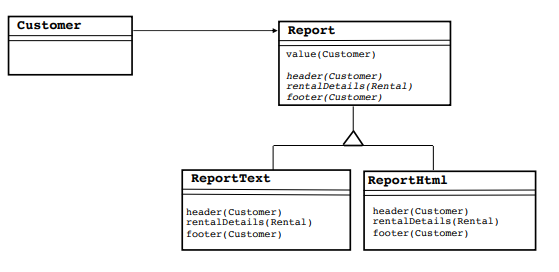


La scelta in fase di creazione può essere fatta si con uno switch, ma si può fare anche con :

* Reflection, tramite il nome della classe vado a creare il relativo tipo
* Dependency Injection

**Pattern Template**

È un pattern che definisce un metodo che rappresenta la parte in comune tra due o più sottotipi



Value fa riferimento a metodi astratti che vanno implementati per ogni sottotipo.

I metodi astratti sono chiamati hooks.

Immagine che contiene testo

Descrizione generata automaticamente