**Visitor Pattern**

Her hansi bir proyekte yeni bir sey elave ederken butun classlari deyismemek ucun visitor patternden istifade edirik.

**Solid Principles - Obyekt yonumlu proqramlasdirma**

* **Single responsibility** principle - Class has one job to do. Each change in requirements can be done by changing just one class
* **Open/closed** principle - Class is happy (open) to be used by others. Class is not happy (closed) to be changed by others. (kodumuz deyisilmeye yox genislenmeye meyilli olmalidir)
* **Liskov substitution** principle - Class can be replaced by any of its children. Children classes inherit parent's behaviours. (real ve oyuncaq ordek) -
* **Interface segregation** principle - When classes promise each other something, they should separate these promises (interfaces) into many small promises, so it's easier to understand. (interfacelerde tekrarlarin qarsisinin alinmasi numunesi, interfacelerin bolunmesi)
* **Dependency** **inversion** principle - When classes talk to each other in a very specific way, they both depend on each other to never change. Instead classes should use promises (interfaces, parents), so classes can change as long as they keep the promise. (databasa numunesi - interface), abtract

Clean Code examples

1. Bir metodda 100den artiq setir kod varsa, o metod diger kicik metodlara ayrilmilidir.
2. Bir classda 1000den arrtiq setir kod varsa, o class diger kicik classlara bolunmelidir.