**Methods**

* **Every line of a method should operate at the same level of abstraction**
  + **Think of what each line of code abstracts from a real life operation. Is it a big/ high level abstraction or is it a small/ low level abstractions**

**double itemPrice = item.getQuantity() \* item.getProduct().getPrice();**

**invoice.append("Item: ").append(item.getProduct().getName()).append(", Quantity: ")**

**.append(item.getQuantity()).append(", Price: ").append(itemPrice).append("\n");­­**

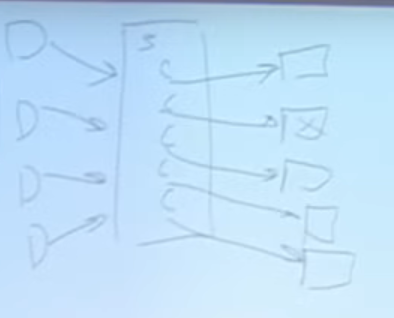
* + **This line deals with high level abstractions: creating an invoice for a transaction**
  + **This line also deals with low level abstractions: calculating an item’s price**
  + **In this case, split this method into 2 different methods, each dealing with its own level of abstraction**
* **A method should do just one thing. A method does one thing if you cannot extract another method from it.**
* **The indent level of a method should not be greater than one or two**
* **Methods should not take more than 3 arguments**
  + **Split the method into more methods**

**OR**

* + **If a method needs many more arguments, then those arguments might be cohesive enough to form an object together.**
* **Try not to pass Booleans as arguments**
  + **Why? If u pass Booleans as arguments, your method will contain if-else statements. Well then why not split that method into 2? One for the true and one for the false branches.**
* **Avoid switch statements**
  + **They need to be updated everytime changes happen.**
  + **They are dependency magnets**

**Squares and rectangles depict modules**

**The switch with all its cases is in the middle**

****

**If each case has a certain functionality, we mash together a lot of modules.**

**Side effects**

* **Side effects are functions which alter the state of the system**
  + **new is a side effect because it allocates a new chunk of memory**
  + **allocc the same**
  + **open is a side effect because it leaves a file open**
  + **close because it closes a file.**
  + **Side effects usually come in pairs like open/close, alloc/free etc …**

**Command and query separation principle**

* **A method which returns void must have a side effect**
* **A method which returns a value must not have a side effect**
* **When using try-catch blocks**
  + **There should be no prefix code(code before the try) and no suffix code(code after the catch)**
  + **There shouldn’t be a lot of code in the try block, only the code which might throw the exception. Big logic should be moved to another method and that method should get called inside the try block. This way, our method only does one thing : Error Processing.**