**Agile Assignment-2**

**By**

**ABHISHEK SHARMA**

**R171218005**

**500067644**

void withdraw(int amount) throws BalanceException {

if (amount > \_balance) {

throw new BalanceException();

}

balance -= amount;

}

Returning blunder codes is an old remnant from procedural programming. In present day programming, mistake taking care of is performed by uncommon classes, which are named special cases. In the event that an issue happens, you "toss" a blunder, which is then "gotten" by one of the exemption overseers. Uncommon blunder dealing with code, which is overlooked in typical conditions, is actuated to react.

**Benefits of refactoring-**

* Frees code from countless conditionals for checking different mistake codes. Special case controllers are a considerably briefer approach to separate typical execution ways from unusual ones.
* Exception classes can execute their own strategies, in this way containing some portion of the mistake taking care of usefulness, (for example, for sending blunder messages).
* Unlike exceptions, error codes can’t be used in a constructor, since a constructor must return only a new object.

2)

int basePrice = quantity \* itemPrice;

double finalPrice = discountedPrice(basePrice);

An extensive rundown of boundaries is difficult to comprehend. Furthermore, calls to such techniques regularly take after a progression of falls, with winding and thrilling worth figurings that are difficult to explore yet must be passed to the strategy. So if a boundary worth can be determined with the assistance of a strategy, do this inside the technique itself and dispose of the boundary.

**Benefits of refactoring-**

We dispose of unneeded boundaries and disentangle technique calls. Such boundaries are regularly made not for the undertaking as it's presently, but rather with an eye for future necessities that may never come.