Final 2009

Names: Ebrahim Taha Ebrahim #3

Ahmed Ebrahim Hafez #4

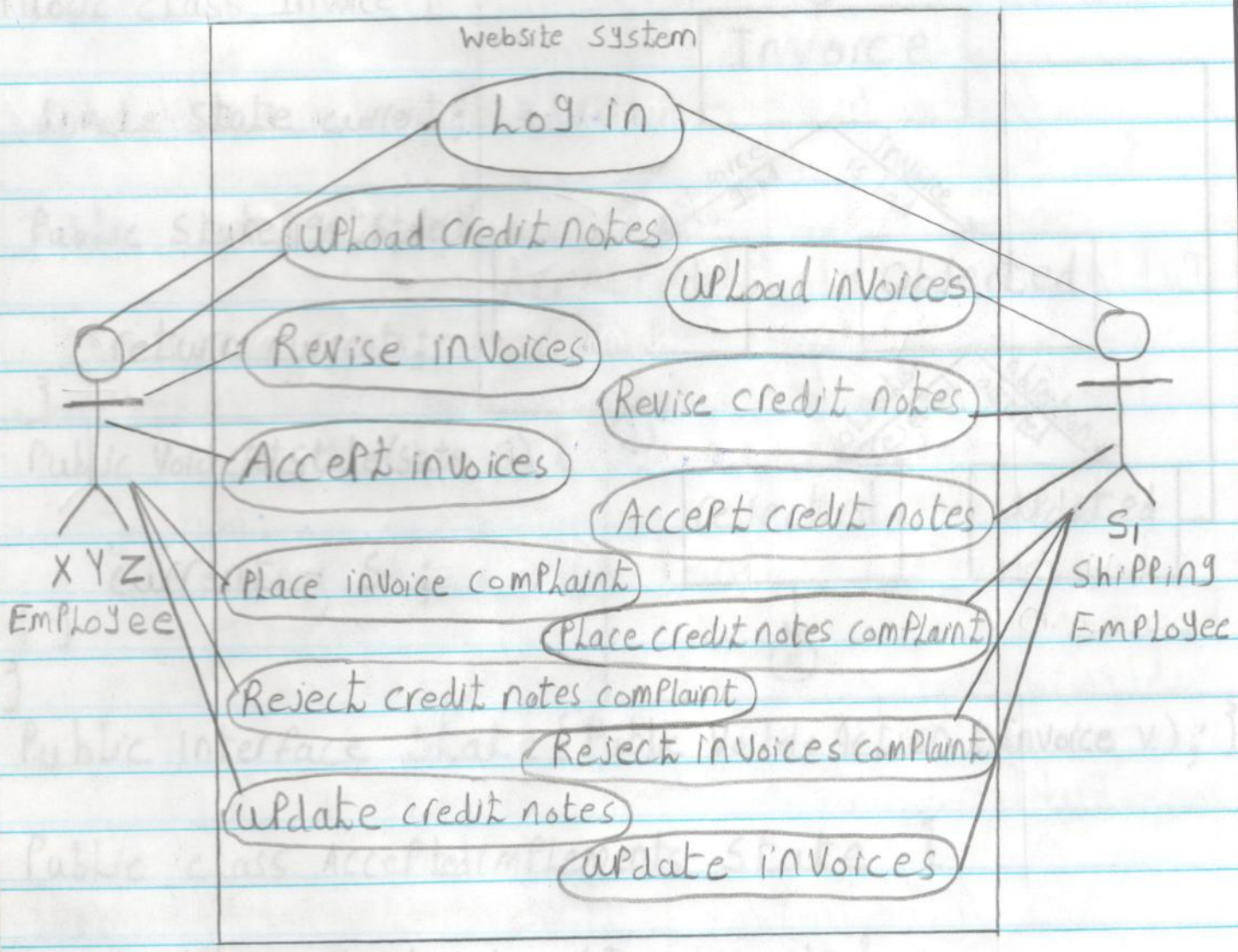
Amr Gamal Mohamed #44

Mohamed El Sayed Mohamed #54

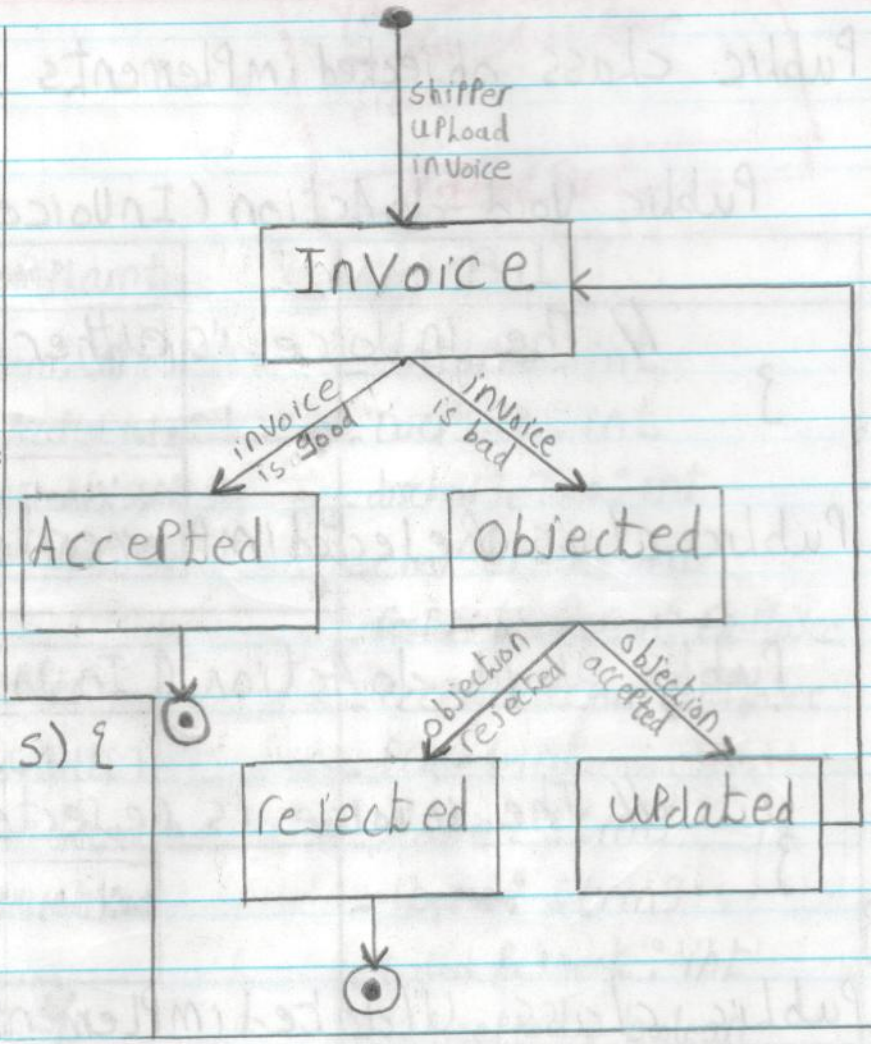
Mohamed Adel Abdel Fatah #58

A

Question 1:


Website system
Log in
Upload credit notes
Upload invoices
Revise invoices
Revise credit notes
Accept invoices
Accept credit notes
Place invoice complaint
Place credit notes complaint
Reject credit notes complaint
Reject invoices complaint
Update credit notes
Update invoices
XYZ Employee
S1 Shipping Employee

Question 2 :




```
Public class Invoice {

  Private State current;

  Public State getState(){

     return current;
  }
  Public Void setState(State S) {

     current = S ;
   }
}
Public interface State { Public Void doAction (Invoice v); }

Public class Accepted implements State {

    Public Void doAction (Invoice V) {

          // set State Accept and not changed
    }

}
```

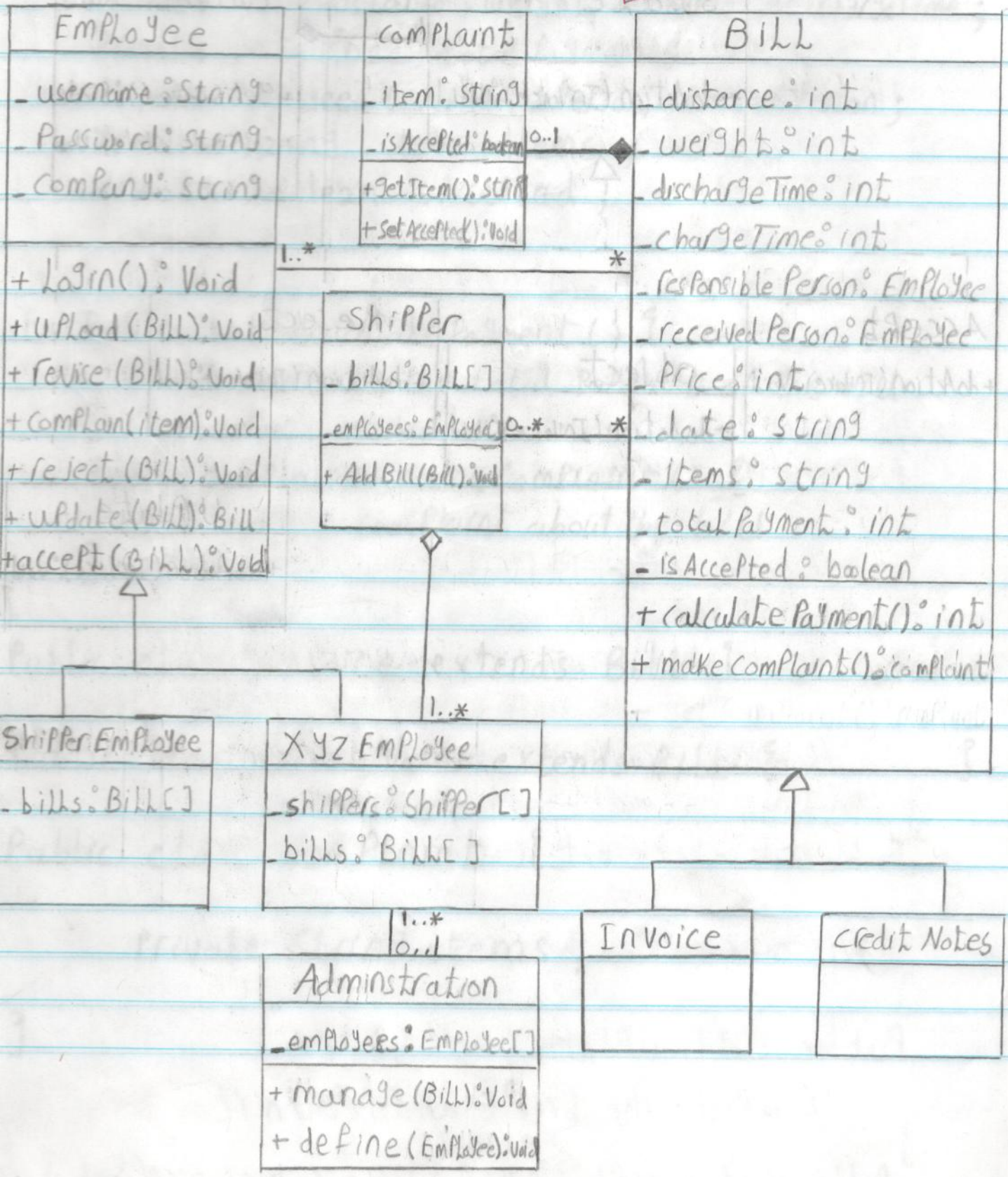
```
Public class objected implements state {

    Public Void doAction (InVoice V) {

        // The inVoice is either rejected or updated.
    }
}
Public class Rejected implements State {

    Public Void doAction ( InVoice V) {

        // The inVoice is rejected
    }
}
Public class Updated implements State {

    Public Void doAction (InVoice V) {

        // The inVoice is updated and either be accepted
           or objected

    }
}
```

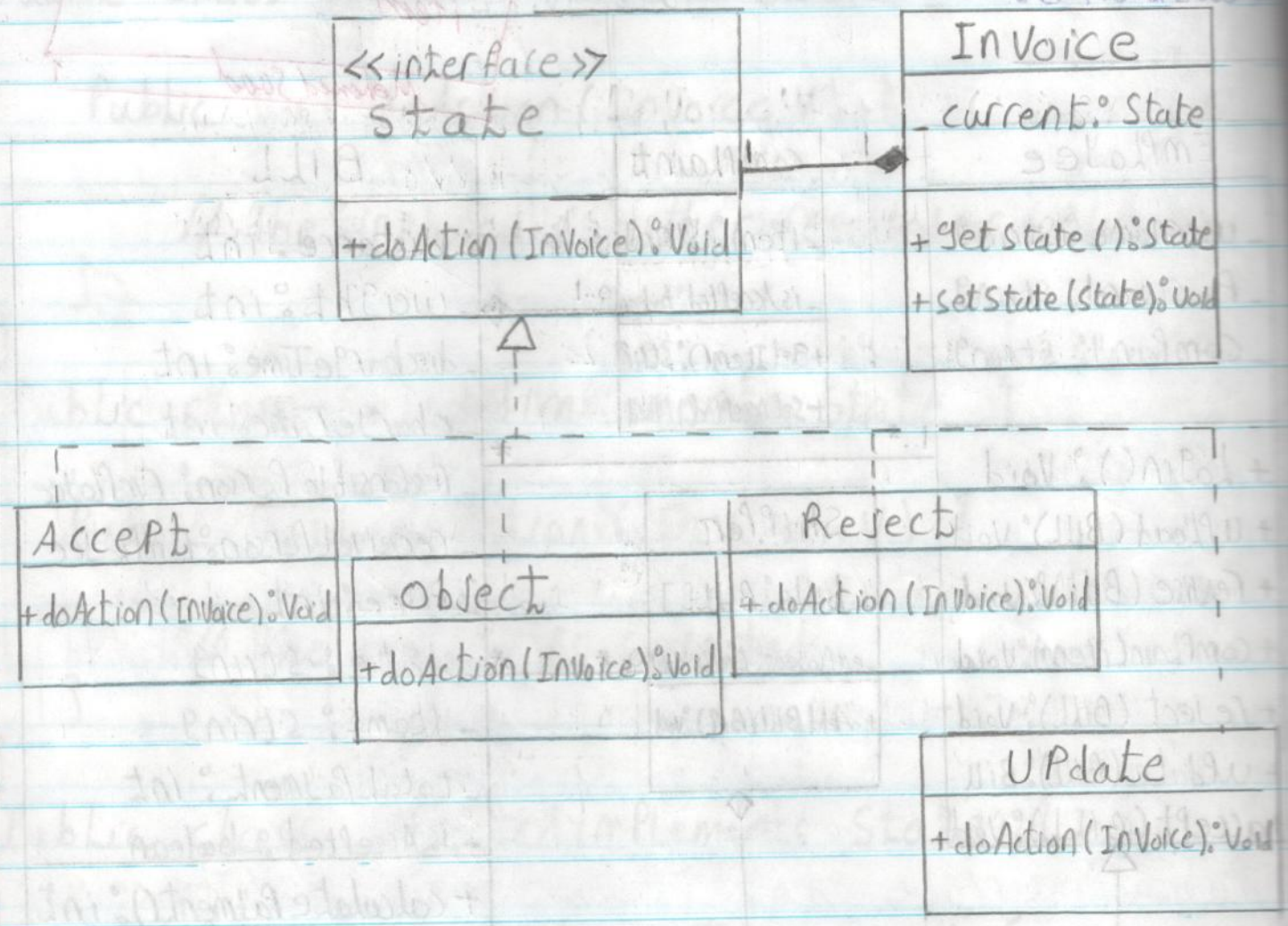
(5)

Bill must have multiple items not a single one You need to add another class for item

Mohamed Saad

Question 3:




<<interface>>
State
+doAction(InVoice):Void
InVoice
-current:State
+getState():State
+setState(State):Void
Accept
+doAction(InVoice):Void
Object
+doAction(InVoice):Void
Reject
+doAction(InVoice):Void
UPdate
+doAction(InVoice):Void

Question 4:

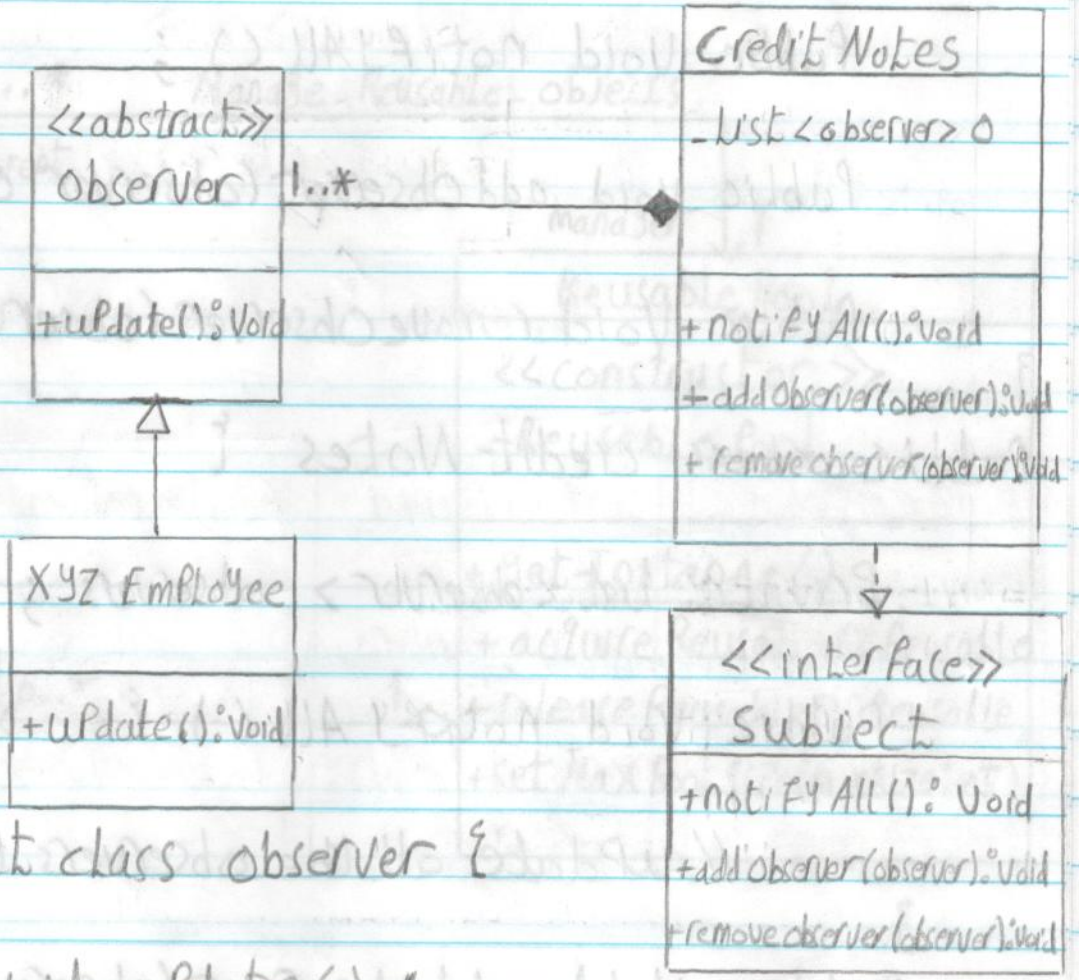
```
Public class BILL {

    Private int distance, weight, dischargeTime, chargeTime;
    Private int Price, totalPayment;
    Private Employee responsiblePerson, recievedPerson;
    Private String date, items;
    Private boolean isAccepted;

    Public int calculatePayment() {
        // calculate the total Payment of the bill
    }
    Public complaint makeComplaint() {
        // make a complaint about the bill
    }
}
Public class Invoice extends BILL { }

Public class CreditNotes extends BILL { }

Public class ComPLaint {

    Private String items;

}
```

```
Public class Employee {

    Private string username, Password, Company ;

    Public Void Login () {

        // log the user into the site.
    }
    Public Void upload ( BILL b) {

        // upload Bill b to the site
    }
    Public Void revise ( BILL b ) {

        // revise the BILL b
    }
    Public Void complain ( string items ) {

        // make complaint with the items.
    }
    Public Void reject ( BILL b) {

        // mark the bill b as rejected.
    }
    Public BILL update ( BILL b) {
        // update the bill b and return it
    }
    Public Void accept ( BILL b) { // mark b as accepted }
}
```

```
Public class ShipperEmployee extends Employee {

    Private BILL[] bills;
}
Public class XyZEmployee extends Employee {

    Private Shipper[] shippers;
    Private Bill[] bills;
}
Public class Adminstration {

    Private Employee[] employees;

    Public Void manage(BILL b) {

        // gives Bill b to an Employee.
    }
    Public Void define(Employee e) {

        // add Employee e to the list of Employees.
    }
}
```

```
Public class Shipper {

    Private BILL [] bills ;
    Private Employee [] employees ;

    Public addBILL (BILL b) {

        // add BILL b to the list of BILLs.
    }
}
Public class Complaint {

    Private String items ;
    Private boolean isAccepted ;

    Public string getItem( ) {

        // return items of the complaint
    }
    Public Void setAccepted ( ) {

        // mark the Complaint as Accepted
    }
}
```

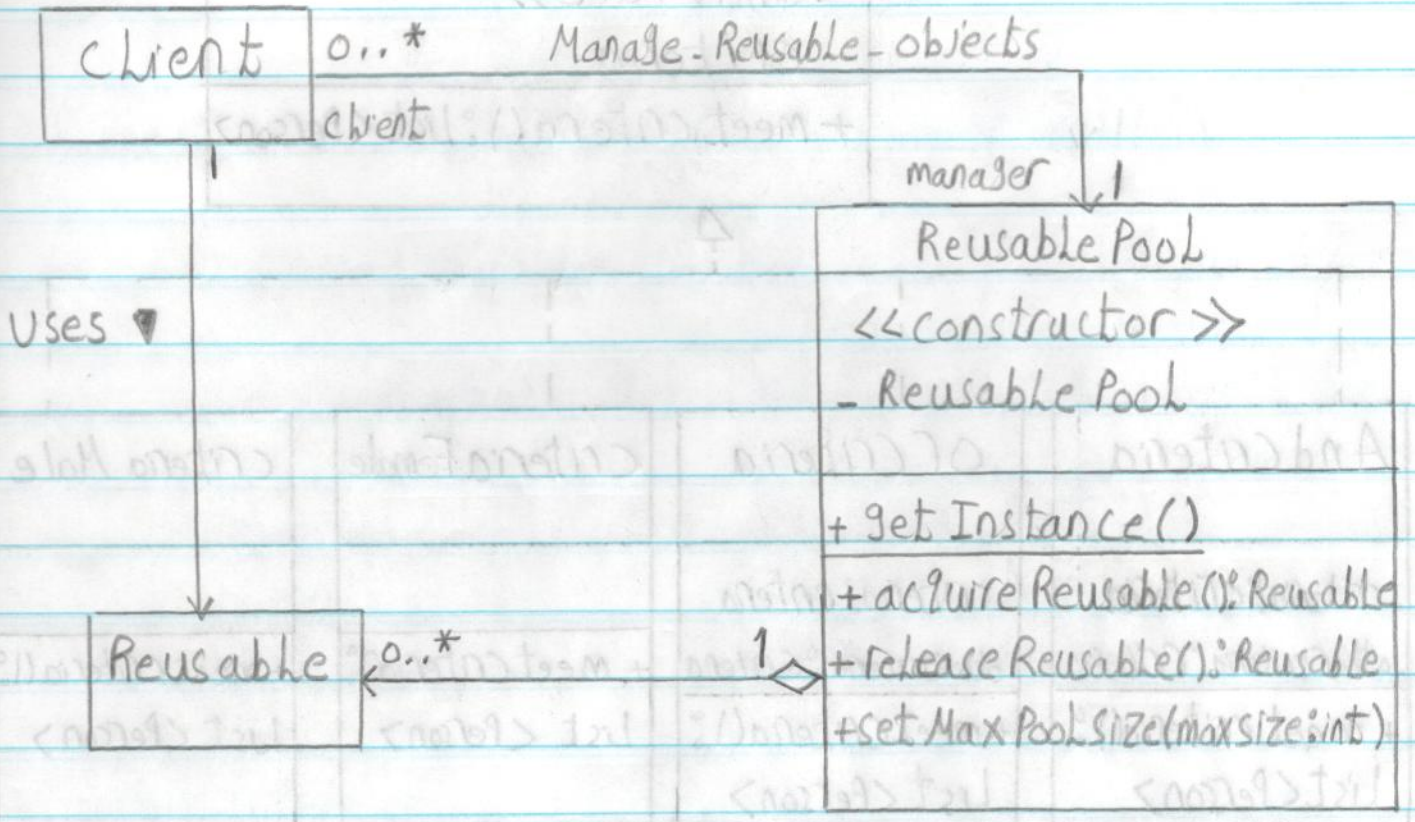
Question 5: observer design Pattern.




```
Public abstract class observer {

    Public void update();
}
Public XYZ Employee extends observer {

    Public void update() {

    // make action according to notify
    }
}
```

```
Public interface Subject {

    Public Void notifyAll();

    Public Void addObserver(observer o);

    Public Void removeObserver(observer o);
}
Public class creditNotes {

    Private list<observer> observers;

    Public Void notifyAll() {
        // update all the observers that action happened
    }
    Public Void addObserver(observer o) {
        // add observer o to the list of observers.
    }
    Public Void removeObserver(observer o) {
        // remove observer o from the list of observers
    }
```

Question 6: a- object Pool design Pattern.



b_ Pull filter design Pattern