(2008)

Name :

- Mohamed Farid sabra (61)

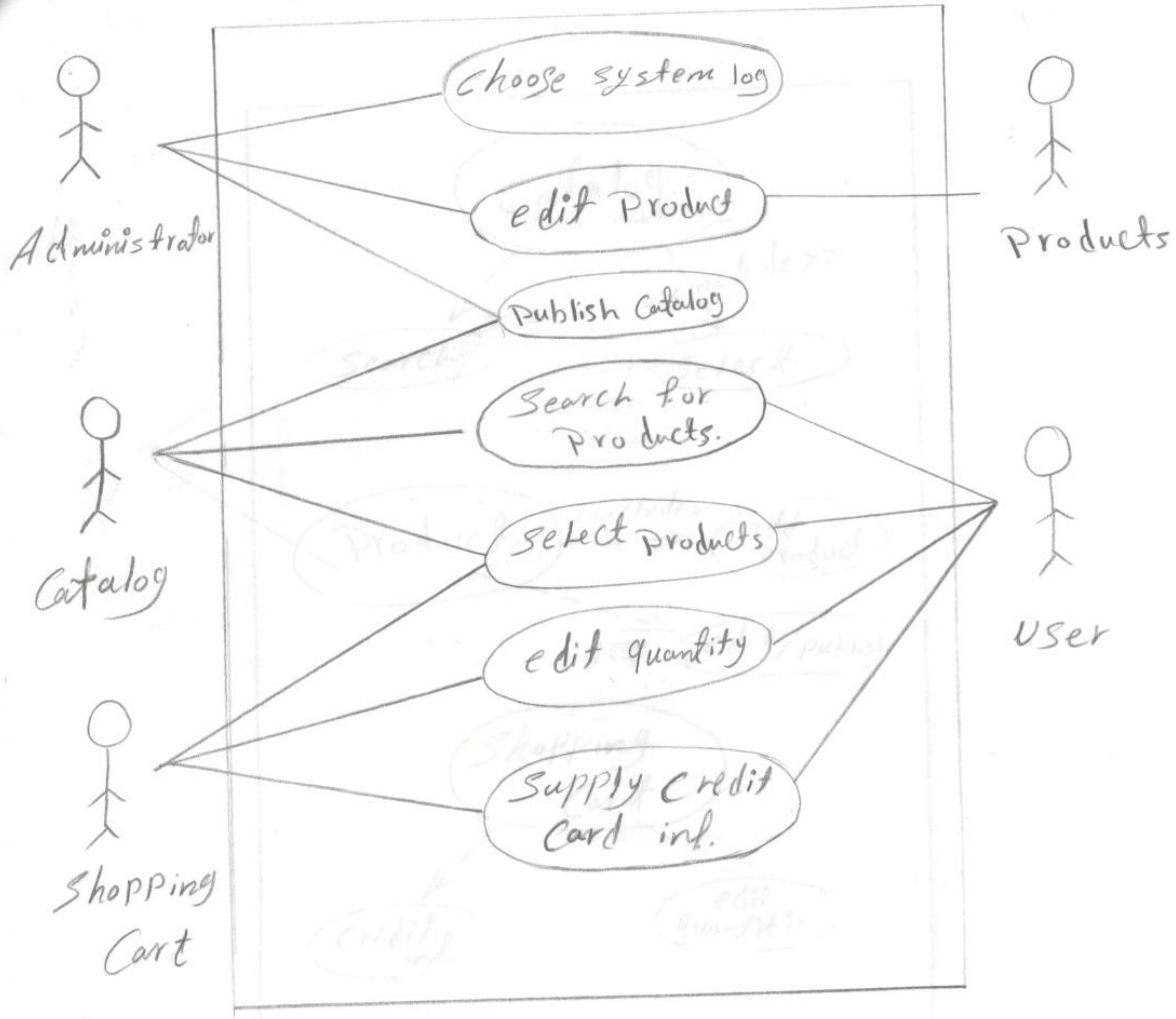
- Mohamed Atef. (59)

- Aya Osam (1)

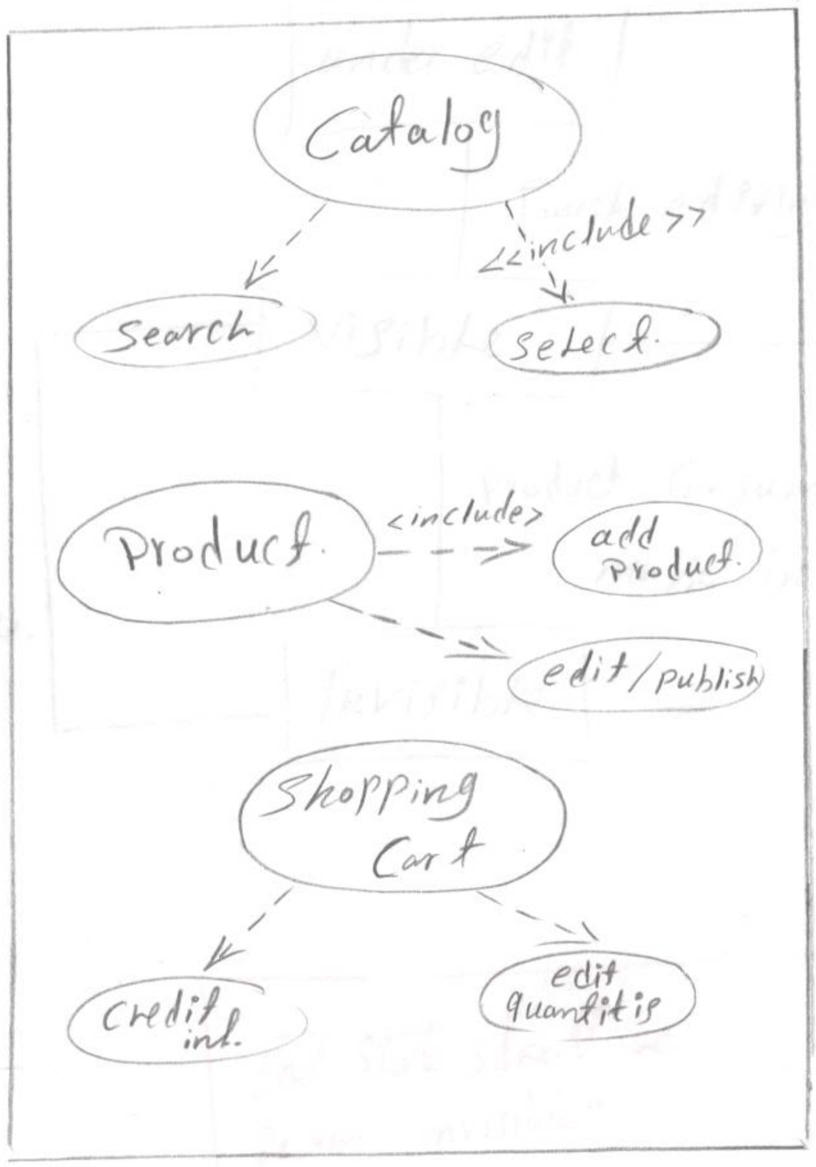
- safaa Hassan (34)

A

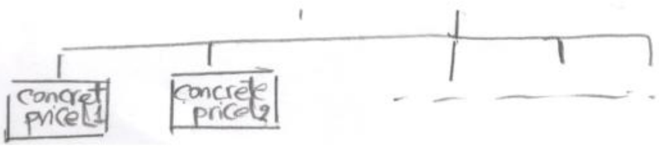
use Case diagram ①



Q1 Use Case ②



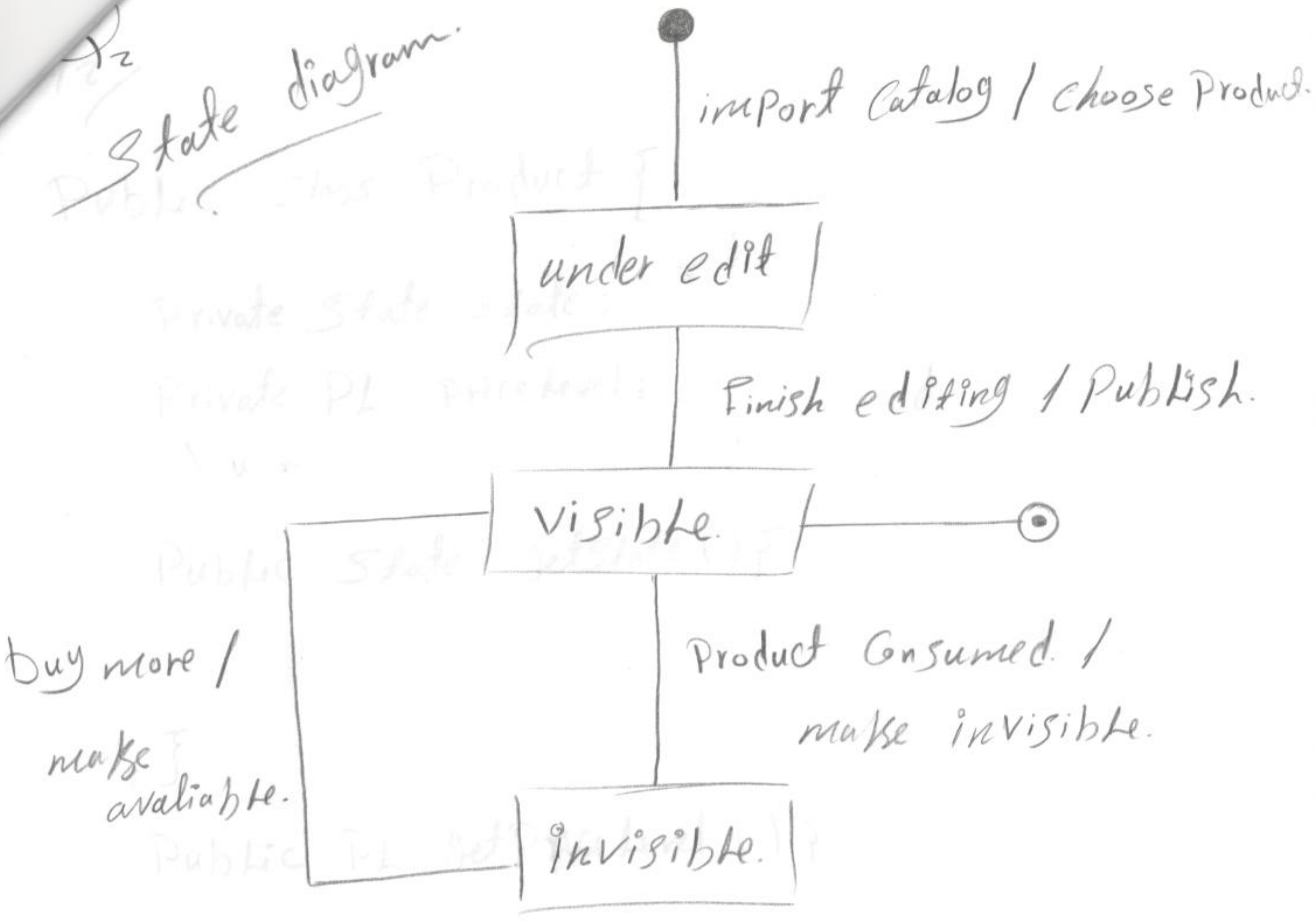
Catalog
<<include>>
search
select
Product
<include>
add Product
edit/publish
Shopping Cart
credit inf.
edit quantitis



concret price 1
concrete price 2

$\frac{1}{2}$

State diagram.

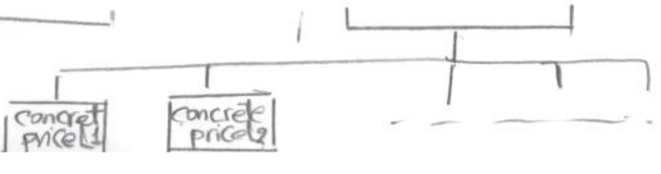

import Catalog / choose Product.
under edit
Finish editing / Publish.
visible.
Product Consumed. / make invisible.
buy more / make avaliable.
invisible.

End State should be from "invisible" as well

Mahmoud Saad

```
Public class Product {

    Private State state;
    Private PL priceLevel;

    Public State getState() {


    }
    Public PL getPriceLevel() {

    }

} // end class.
```

```
Public abstract Class State {

    Public static final int EDIT =0 ;
    Public static final int PUBLISH =1;
    Public static final int OUT_OF_STOK =2;
    Public static final int SUPPLIED =3;

    Private static UnderEdit underEdit;
    Private static Visible visible;
    Private Static UnVisible unVisible;


    Public void start() {
    }

    Public State ProcessEvent (int event) {

    }

    Prodecte State nextState (int event) {

    }

} //end class.
```



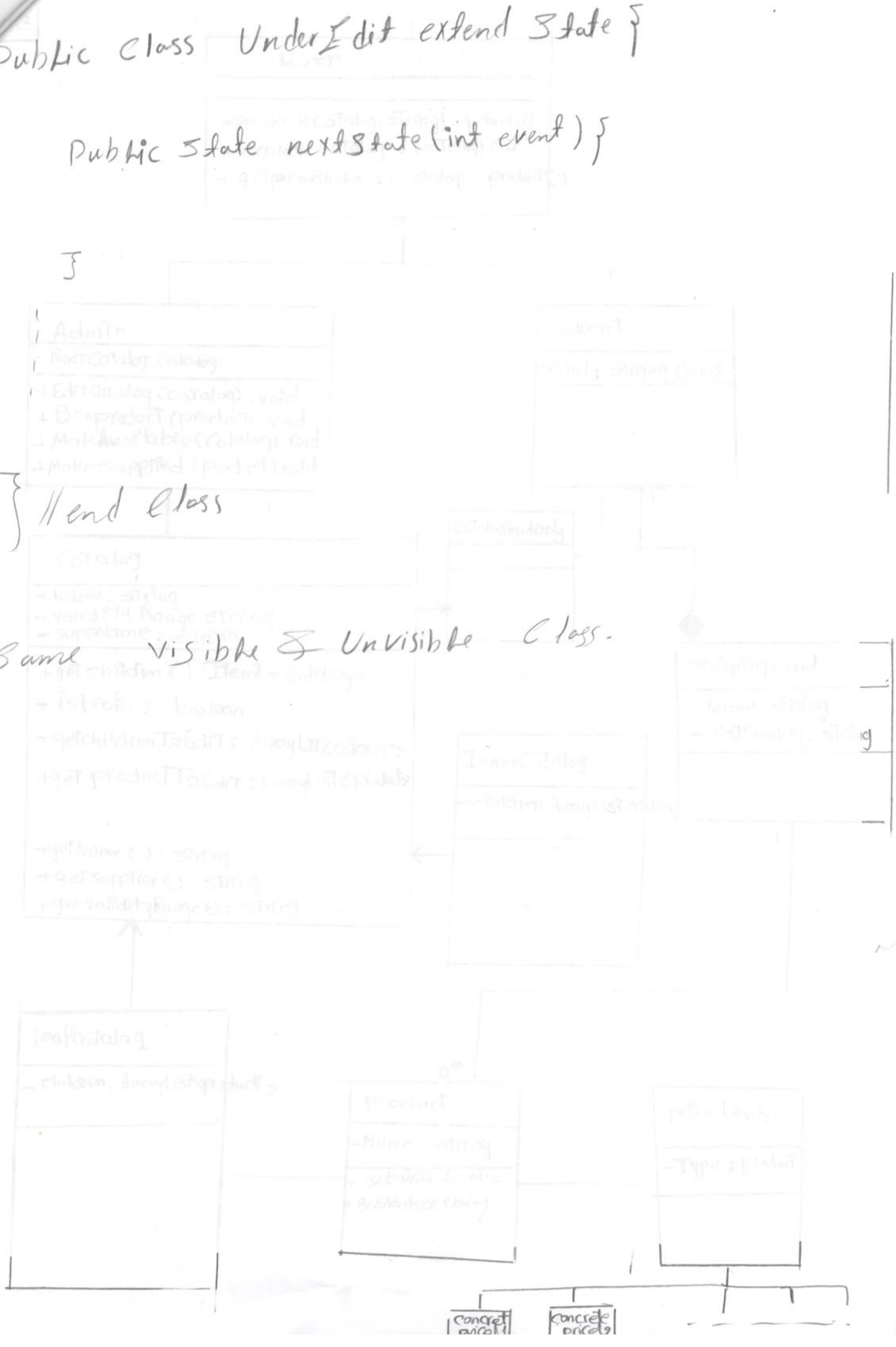
Public Class UnderEdit extend State {

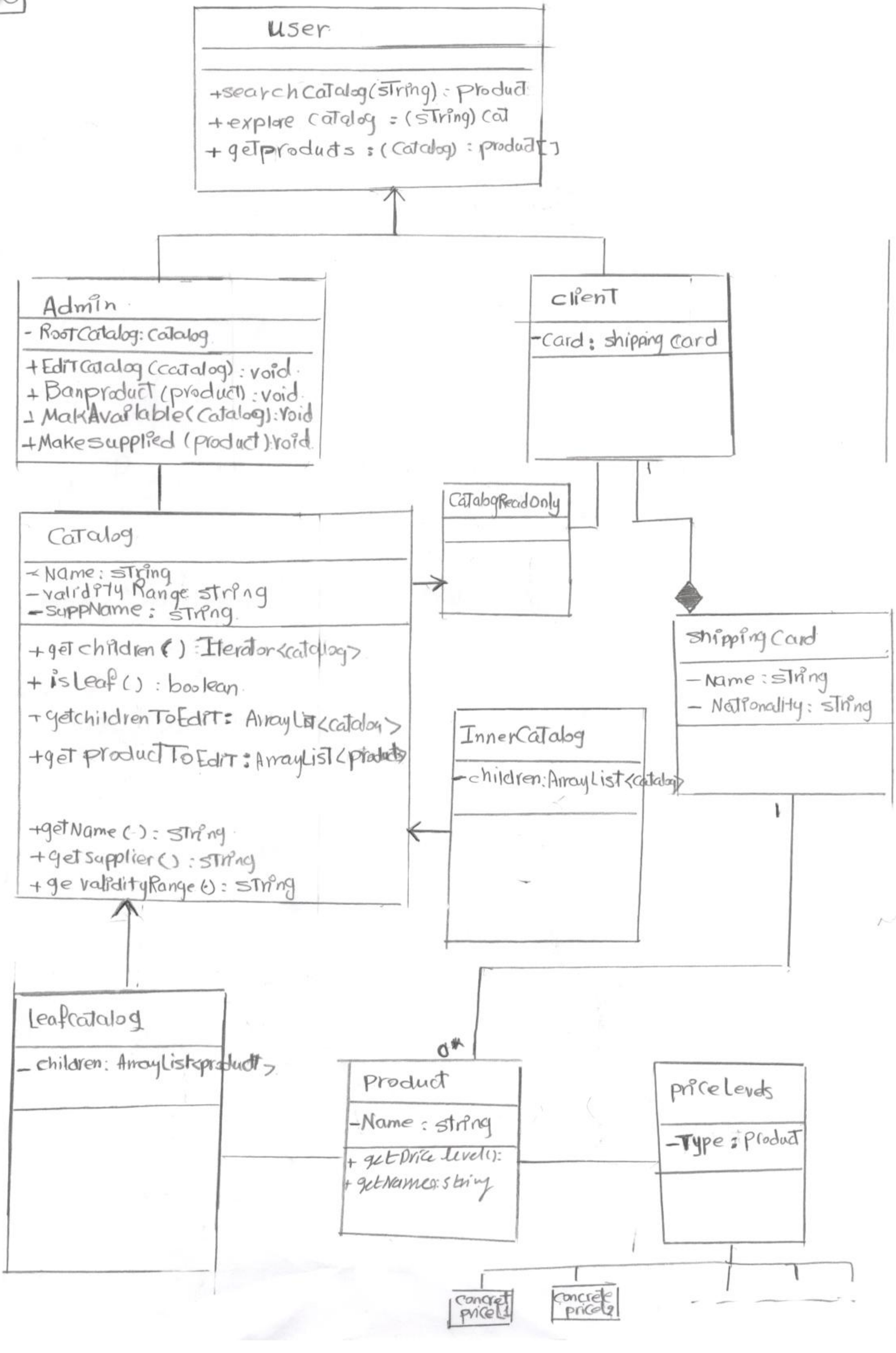
Public State nextState(int event) {

}

} //end Class

the Same visible & Unvisible Class.




User
+searchCatalog(sTring) : Product
+explore catalog = (sTring) Cat
+ getproducts : (Catalog) : product[]
Admin
- RootCatalog: Catalog
+EditCatalog (catalog) : void
+ Banproduct (product) : void
+ MakAvailable(Catalog): Void
+Makesupplied (product): void
clienT
-Card : shipping card
1
CatalogReadOnly
Catalog
- Name : sTring
- validity Range string
- suppName : sTring
+ get children ( ) : Iterator<catalog>
+ isLeaf ( ) : boolean
+ getchildrenToEdit : ArrayList<catalog>
+get productToEdit : ArrayList<products>
+getName ( ) : sTring
+ get supplier ( ) : sTring
+ ge validityRange ( ) : sTring
shipping Card
- Name : sTring
- Nationality : sTring
1
InnerCatalog
-children: ArrayList<catalog>
leafcatalog
_ children: ArrayList<product>
0*
Product
-Name : string
+ getPrice level():
+ getName(): string
priceLevels
-Type : Product
concret price1
concrete price2

```
{ Interface user {
 public Product searchCatalg (Stringname);
 public Product [] get Products ( Catalog catalog);
 public cataloge [] exploreCatalg (String name);
 }

public Class Admin extends user {
 Catalog RootCatalog;
Public
void Edit catalog ( Catalog catalog) {

          }

public
void BanProduct (Product product) {
       }
Public
void MakeAvailable (Product product) {
      }
public
void Make supplied (Product product) {
      }

public product searchCatalog (String, Name) {
            }

      |
      | Interface method,
      |

      | }
          }

Public product[] load(productDisplay screen sr)
      Sr- Add();
   }
```

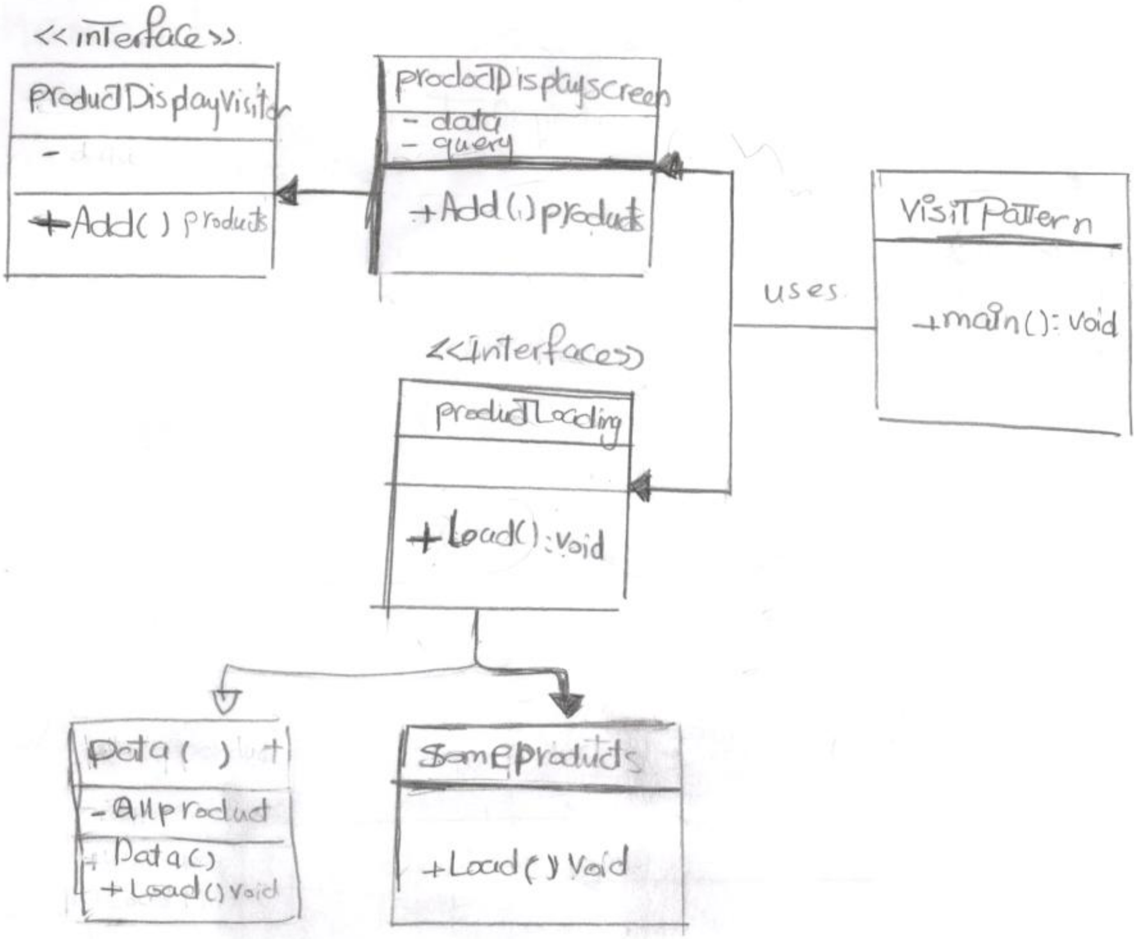
```
ic class client {
  ShippingCard card;

  }

  interface methods

  }
                                  implements CatalogReadOnly
Public Abstract class Catalog {
  protected string Name;
  protected string validity Range;
  protected String Supplier Name;
  public Abstract getChildren ();
  public abstract boolean  is leaf  ();
  public abst Iterator < product > getProducts ToRead();
  public Iterator < Catalogs > getchildrenToRead;
  public

  }

  }


public product[] load(productDisplay screen sr)
    sr.Add();
}
```

```
public Interface catalogReadOnly () {
    public String getName();
    public Iterator <Catalog> getCatalogsToRead)
    Public Iterater <product> getproductsToRead;
    public boolean is leaf ();
}

public class leaf Catalogue extends catalog {
    throw exception when trying to access
    children catalogues, other-wise returns
    the iterator to user @ the ArrayList to
    the admin.

}

public Class Inner Catalog extends catalog {
    throw "     "     "     "
    "  products,  "     "  "
       "  "  "  "  "

}

public product[] load(productDisplay screen sr)
    sr. Add():
```

Chaos

```
c class shipping card {
 private String Name;
 private String Nationality;
 private String shippingNo;
 public String getName(){--};
 public  "   getNat  (){--};
 public  "   getshippingNo(){--};
 public void setName(String Nm){--};
 public void setNat  (String Nat){--}..
 public void setshippingNo (String No){--};

   }

public Interface Pricelevel {
     public setUnitPrice (int n0){--}
     public getPrice (int numUnits {--}:

  }

public product[] load(productDisplay screen sr)
     sr.Add():
```

Q58 A Visitor design pattern

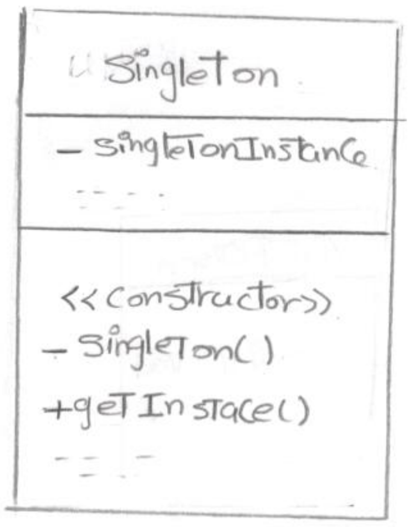


```
class
productDisplay screen() {
    private data
    private query

public Products[] Add() {

    return products[];
  }
```

```
class
SomeProducts( ) {

  public load(productDisplayscreen sr)
   {
   sr. Add();

   }
}
```

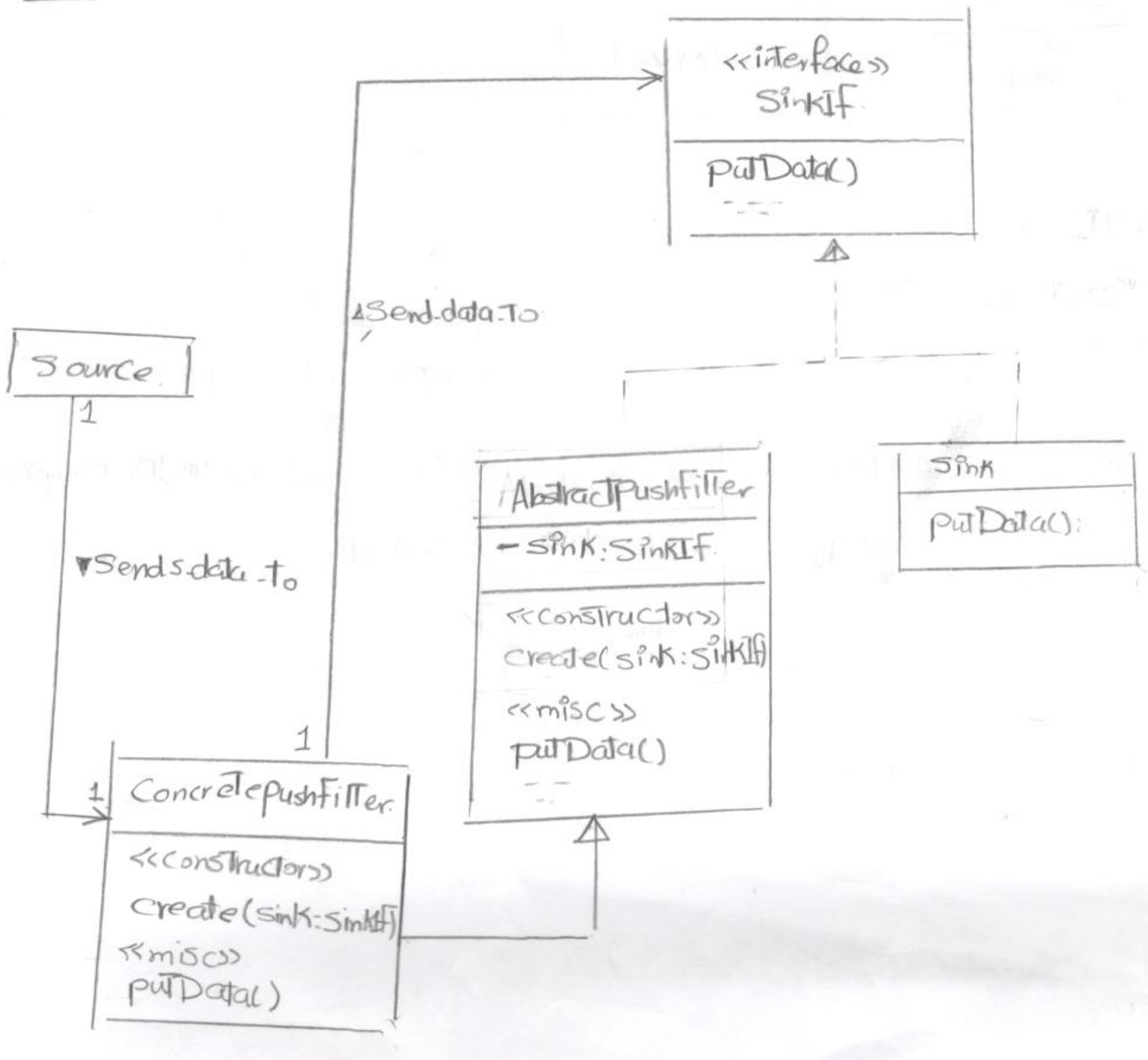
```
class
Data() {
  private all Products[ ];
public Data() {

}
public product[] load(productDisplay screen sr)
   sr. Add();
```

Q6:



Single Ton

Filter


<<interface>>
SinkIF
putData()
- - -
▲Send data To
Source
1
▼Sends data to
1
1
ConcretepushFilter
<<constructor>>
create(sink:SinkIF)
<<misc>>
putData( )
AbstractPushFilter
- sink: SinkIF
<<constructor>>
create(sink:SinkIF)
<<misc>>
putData()
- -
Sink
putData();