**Project 1**

**1. Write two possible use cases for each user story: one is the common case and one is the exception.**

“Cases depend on the type”

“Attached the source code what the common type is and what the exceptions are”

* Product
* Common case:

public int mProductID;  
public String mName, mVendor, mDescription;  
public double mPrice, mQuantity;

* exception:

public void actionPerformed(ActionEvent actionEvent) {  
 ProductModel product = new ProductModel();  
  
 String id = txtProductID.getText();  
  
 if (id.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "ProductID cannot be null!");  
 return;  
 }  
  
 try {  
 product.mProductID = Integer.*parseInt*(id);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "ProductID is invalid!");  
 return;  
 }  
  
 String name = txtName.getText();  
 if (name.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "Product name cannot be empty!");  
 return;  
 }  
  
 product.mName = name;  
  
 String price = txtPrice.getText();  
 try {  
 product.mPrice = Double.*parseDouble*(price);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "Price is invalid!");  
 return;  
 }  
  
 String quant = txtQuantity.getText();  
 try {  
 product.mQuantity = Double.*parseDouble*(quant);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "Quantity is invalid!");  
 return;  
 }  
  
 switch (StoreManager.*getInstance*().getDataAdapter().saveProduct(product)) {  
 case SQLiteDataAdapter.*PRODUCT\_DUPLICATE\_ERROR*:  
 JOptionPane.*showMessageDialog*(null, "Product NOT added successfully! Duplicate product ID!");  
 default:  
 JOptionPane.*showMessageDialog*(null, "Product added successfully!" + product);  
 }  
}

* Customer
* Common case:

public int mCustomerID;  
public String mName, mPhone, mAddress;

* Exception:

public void actionPerformed(ActionEvent actionEvent) {  
 CustomerModel customer = new CustomerModel();  
  
 String id = txtCustomerID.getText();  
  
 //Check if the id is not empty  
 if (id.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "CustomerID cannot be null!");  
 return;  
 }  
  
 //Check if the customer id is a valid input  
 try {  
 customer.mCustomerID = Integer.*parseInt*(id);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "CustomerID is invalid!");  
 return;  
 }  
  
 String name = txtName.getText();  
 if (name.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "Customer name cannot be empty!");  
 return;  
 }  
  
 customer.mName = name;  
  
 String phone = txtPhone.getText();  
 if (phone.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "Customer phone cannot be empty!");  
 return;  
 }  
  
 customer.mPhone = phone;  
  
 String address = txtAddress.getText();  
 if (address.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "Customer phone cannot be empty!");  
 return;  
 }  
  
 customer.mAddress = address;  
  
 switch (StoreManager.*getInstance*().getDataAdapter().saveCustomer(customer)) {  
 case SQLiteDataAdapter.*CUSTOMER\_DUPLICATE\_ERROR*:  
 JOptionPane.*showMessageDialog*(null, "Customer NOT added successfully! Duplicate customer ID!");  
 default:  
 JOptionPane.*showMessageDialog*(null, "Customer added successfully!" + customer);  
 }  
  
}

* Purchase
* Common case:

public int mPurchaseID, mProductID, mCustomerID;  
public double mPrice, mQuantity, mCost, mTax, mTotal;  
public String mDate;

* Exception:

public void actionPerformed(ActionEvent actionEvent) {  
  
 String pId = txtPurchaseID.getText();  
 if (pId.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "PurchaseID cannot be null!");  
 return;  
 }  
  
 try {  
 purchase.mPurchaseID = Integer.*parseInt*(pId);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "PurchaseID is invalid!");  
 return;  
 }  
  
 String cId = txtCustomerID.getText();  
 if (cId.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "CustomerID cannot be null!");  
 return;  
 }  
  
 try {  
 purchase.mCustomerID = Integer.*parseInt*(cId);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "CustomerID is invalid!");  
 return;  
 }  
  
 String productId = txtProductID.getText();  
 if (productId.length() == 0) {  
 JOptionPane.*showMessageDialog*(null, "ProductID cannot be null!");  
 return;  
 }  
  
 try {  
 purchase.mProductID = Integer.*parseInt*(cId);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "ProductID is invalid!");  
 return;  
 }  
  
 String quant = txtQuantity.getText();  
 try {  
 purchase.mQuantity = Double.*parseDouble*(quant);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "Quantity is invalid!");  
 return;  
 }  
  
 //receipt  
 product = StoreManager.*getInstance*().getDataAdapter().loadProduct(purchase.mProductID);  
 customer = StoreManager.*getInstance*().getDataAdapter().loadCustomer(purchase.mCustomerID);  
  
 receipt.appendHeader("\nCustomer Receipt\n");  
 receipt.appendCustomer(customer);  
 receipt.appendProduct(product);  
 receipt.appendPurchase(purchase);  
 receipt.appendHeader("\nThank you for your purchase!");  
  
 switch (StoreManager.*getInstance*().getDataAdapter().savePurchase(purchase)) {  
 case SQLiteDataAdapter.*PURCHASE\_DUPLICATE\_ERROR*:  
 JOptionPane.*showMessageDialog*(null, "Purchase NOT added successfully! Duplicate product ID!");  
 default:  
 JOptionPane.*showMessageDialog*(null, "Purchase added successfully!\n"  
 + purchase + "\n"  
 + receipt.toString());  
 }  
}

**2. Design the screens (UI windows and widgets) the system should display in each use case.**

Name: “Add a new product into the system”

Steps:

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose command “Add Product”  Main screen:  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.20.54 AM.png | 2. Display “Add Product” screen  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.30.54 AM.png |
| 3. Input data then click “Add” button  “Add Product” screen with data:  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.35.55 AM.png | 4. Hide “Add Product” screen and display “Add Product Successfully!” screen:  “Add Product Successfully” screen:  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.38.25 AM.png |
| 5. Click on “OK” button:  Screen%20Shot%202019-09-10%20at%203.29.36%20PM.png | 6. Display “Main” screen  Main screen:  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.20.54 AM.png |

Name: “Add a new customer into the system”

Steps:

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose command “Add Customer”  Main screen:  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.20.54 AM.png | 2. Display “Add Customer” screen  ../../Activities/Activity9/Screen%20Shot%202019-09-17%20at%202.05.34%20PM.png |
| 3. Input data then click “Add” button  “Add Customer” screen with data:  ../../Activities/Activity9/Screen%20Shot%202019-09-17%20at%202.05.34%20PM.png | 4. Hide “Add Customer” screen and display “Add Product Successfully!” screen:  “Add Customer Successfully” screen:  Customer Added Successfully!  OK |
| 5. Click on “OK” button:  Customer Added Successfully!    OK | 6. Display “Main” screen  Main screen:  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.20.54 AM.png |

Name: “Add a new purchase into the system”

Steps:

|  |  |
| --- | --- |
| **Actor** | **System** |
| 1. Choose command “Add Purchase”  Main screen:  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.20.54 AM.png | 2. Display “Add Purchase” screen  ../../Activities/Activity9/Screen%20Shot%202019-09-17%20at%202.05.38%20PM.png |
| 3. Input data then click “Add” button  “Add Purchse” screen with data:  ../../Activities/Activity9/Screen%20Shot%202019-09-17%20at%202.05.38%20PM.png | 4. Hide “Add Purchase” screen and display “Add Purchase Successfully!” screen:  “Add Purchase Successfully” screen:    Purchase Added Successfully!  OK |
| 5. Click on “OK” button:  Purchase Added Successfully!  OK | 6. Display “Main” screen  Main screen:  /Users/jiyeonlee/Desktop/Screen Shot 2019-09-10 at 11.20.54 AM.png |

**3. Design the database physically and prepare data for the tables, with at least 5 products, 5 customers, and 10 purchases.**

**<Relations>**

Sell products to customers:

* Customer presents a list of items (to buy)
* Cashier check price of each item and calculate the total cost
* Customer pays
* Cashier gets the payments and provides the sale receipt

Manage inventory

* If a product is in inventory, check it is available
* If not, uncheck it is available

Buy products from suppliers:

* The owner or store manager check the current paper inventory
* The owner or store manager order materials needed
* Suppliers receive the order and ship for the products
* The owner or store manager inspect the product and pay the bill

**<Attributes and keys>**

- Customers(CustomerID\*, Name, PhoneNumber, Address);

- Products (ProductID\* ,Name, Price, Quantity, TaxRate);

- Purchases (PurchaseID\*, CustomerID, ProductID, DateTime, Quantity, Price, Tax,

TotalCost);

\* represents primary keys.

CREATE TABLE Customers(

CustomerID INTEGER not null PRIMARY key,

Name TEXT,

PhoneNumber TEXT,

Address TEXT);

CREATE TABLE Products (

ProductID INTEGER not null PRIMARY key,

Name TEXT,

Price REAL,

Quantity REAL,

TaxRate REAL);

CREATE TABLE Purchases(

PurchaseID INTEGER not null PRIMARY key,

CustomerID INTEGER,

ProductID INTEGER,

DateTime TEXT,

Quantity REAL,

Price REAL,

Tax REAL,

TotalCost REAL);

INSERT INTO Customers

VALUES

(1, 'Tung', ' ', 'Auburn, AL'),

(2, 'Nguyen', ' ', 'Auburn, AL'),

(3, 'Adam Smith', '800-100-1000', 'USA'),

(4, 'Eve', ' ', 'Eden'),

(5, 'Jen', '(213) 456-7890', 'California');

INSERT INTO Products

VALUES

(1, 'Apple', 0.99, 100.0, 0.09),

(2, 'Orange', 1.99, 200.0, 0.09),

(3, 'iPhone X', 1199.99, 100.0, 0.09),

(4, 'Banana', 0.5, 100.0, 0.09),

(5, 'Lemon', 0.99, 150.0, 0.09);

INSERT INTO Purchases

VALUES

(1, 1, 1, '09/09/19 10:48', 1, 0.99, 0.09, 1.08),

(2, 2, 2, '09/02/19 04:56', 3, 5.97, 0.54, 6.51),

(3, 3, 3, '09/03/19 08:33', 2, 2399.98, 216.00, 2615.98),

(4, 4, 4, '09/05/19 12:12', 7, 3.50, 0.315, 3.815),

(5, 5, 5, '09/10/19 02:38', 5, 24.0, 2.16, 26.16),

(6, 1, 1, '09/10/19 03:24', 10, 9.90, 0.89, 10.79),

(7, 2, 2, '09/10/19 04:58', 15, 9.95, 0.90, 10.85),

(8, 3, 3, '09/11/19 22:33', 1, 1199.99, 108.00, 1307.99),

(9, 4, 4, '09/12/19 23:40', 20, 10.00, 0.90, 10.90),

(10, 5, 5, '09/13/19 14:43', 6, 5.94, 0.53, 6.50);