AT2 Final Project

30003673

TAFE

PROGRAMMING II

Table of Contents

[Introduction 2](#_Toc11513400)

[A simple explanation of what the program is required to do: 2](#_Toc11513401)

[Development Methodologies 2](#_Toc11513402)

[Analysis 3](#_Toc11513403)

[Project plan: 3](#_Toc11513404)

[Tasks that need to be performed: 3](#_Toc11513405)

[People doing tasks: 3](#_Toc11513406)

[Physical resources required: 3](#_Toc11513407)

[Gantt Chart 4](#_Toc11513408)

[Design 5](#_Toc11513409)

[Product ADD button 5](#_Toc11513410)

[Read binary file 6](#_Toc11513411)

[Write binary file 7](#_Toc11513412)

[Class diagrams 8](#_Toc11513413)

[Test Data and Evidence 9](#_Toc11513414)

[Code 10](#_Toc11513415)

# Introduction

## A simple explanation of what the program is required to do:

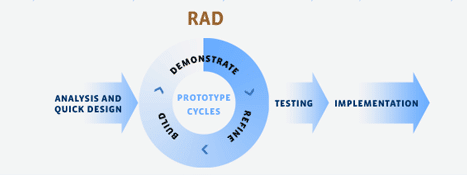
The program is required to load all information from products, customers and transaction from a binary .dat file. The user then inputs the item info and games or item info and platform or item info and accessories then press the add button. The data with be save into List<T> and only the ProductID and Description will be displayed in the list box below the Product Add button. If the user types in to much or not enough data the program with prompt a error message. If the user inputs the wrong type of data (ex. String instead of int) the program with prompt a error message as well. After the user will enter the customer info and press add. This will display the CustID, Family Name and First name into the list box below the Customer Add button. Once these 2 steps are complete the user must select the record from both Product and Customer list boxes and this will populate the CustID, Product ID and Retail Price in the transaction info below. The user then must type in Quantity and Date and press the Transaction Add button. Once this is done the CustID, ProductID, Quantity, Retail Price and Date will be displayed in the list box below the Transaction Add button. If the user wishes to clear all Item info, they must double click the ProductID text box (highlighted in yellow). If the user wishes to clear all Customer Info they must double click the CustID text box (highlighted in pink). When a record in the Transaction list box is clicked the transaction info is filled into the remaining transaction input text boxes, and the appropriate Product and Customer records are selected in the upper list boxes. This will also auto fill the text boxes. Once the user is done they can exit the program which all data will then be written back to the three binary files.

## Development Methodologies

One option would be the RAD Model (Rapid Application Development)

This is a type of step-by-step model with mechanisms of tasks are equal like mini projects. In the RAD model the components and/or system functionalities are developed in a parallel similar as the development of mini projects. The process of the design components and writing the code for function the software involves itself and the planning required for developing the final fully functional system project.

These steps are Phases in RAD, Business modelling, Data modelling, Process modelling, Application generation and testing and turnover.



<https://www.testingexcellence.com/rapid-application-development-rad/>

# Analysis

The data that needs to be inputted into the program is first Products : Item info and/or Games, Item info and/or Platform, and Item info and/or Accessories. Next fill in text boxes under customer and press add. After having a record for both Product and Customer click on one record from each then input Quantity and date in transaction info and press add

The processes that need to be performed while the program is running is first the binary load method which check to see if any data has been saved from before. If nothing has been saved the program with prompt a error message. If it has then the program will load the saved data. If the user forgets to fill in a text box, type in the wrong data type or types to much data then program will prompt a error message. The program will also run 3 display methods for products, customer and transaction to save the data into a List<T> then display it in the appropriate text boxes

When the program close the products, customers and transaction will each be saved to a .dat file (product.dat, customers.dat, transactions.dat)

Project plan:

## Tasks that need to be performed:

* Create a project plan for what needs to be completed - Visual Studio code, MS Documentation with Gantt chart etc.
* Make sure I have proper resources available to complete the task – Pc or laptop, Visual Studios, MS Word, MS Excel
* Application needs to be completed before the end of semester 2 (June 24th)

## People doing tasks:

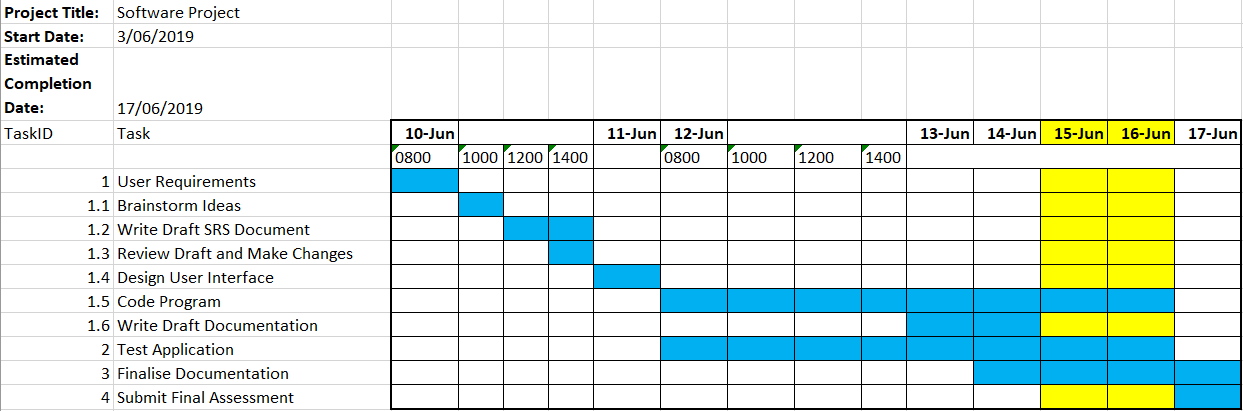
I will be completing the following tasks:

* Using visual studios, create a new project and design a new form with proper placement of text boxes and buttons etc.
* Create classes for the program including – Product (Parent class), Games, Platforms, Accessories, Customers and Transactions
* Code each button to accept data for the text boxes
* Create methods to display text box data into the desired list box
* Provide adequate comments for each button and method
* Design an MS Word document explaining the project and its features
* Explain a development methodology that would work with this project
* An analysis of what the program does
* A Gantt chart of the tasks needed to be completed
* Activity diagrams of the following – Add button, Read binary file, Write binary file and Class diagram
* Test data and evidence using screenshots
* Code for the main form of the program with all Methods and functions used
* Step by step Training Guide, saved as a PDF, of how to navigate through the program
* Upload to Blackboard for marking

## Physical resources required:

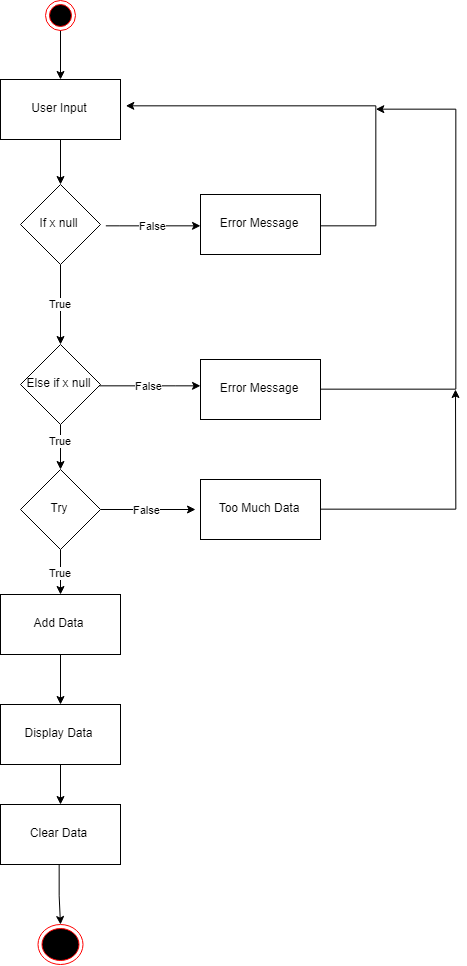
1. User Requirements – Review case study for Games Store (Virada)
   1. Brainstorm idea on how to create application – On campus
   2. Write draft SRS document – PC or laptop
   3. Review draft and make changes if necessary – edit draft thought MS Word on PC or laptop
2. Design User Interface – Design with access using a PC or laptop
3. Code Program – code program using visual studios using a PC or laptop
4. Write Draft Documentation – use MS Word from a PC or laptop
5. Test Application – test application on visual studios from a PC or laptop
6. Finalize Documentation – finalize document using MS word from a PC or laptop
7. Submit final assessment – Email Chris for marking

## Gantt Chart

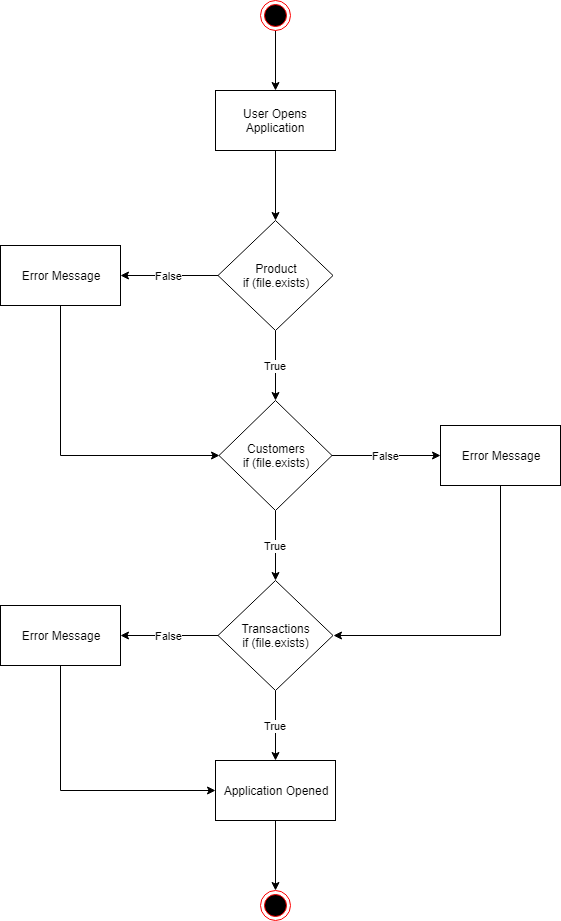


# Design

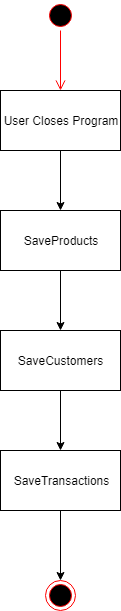
## Product ADD button



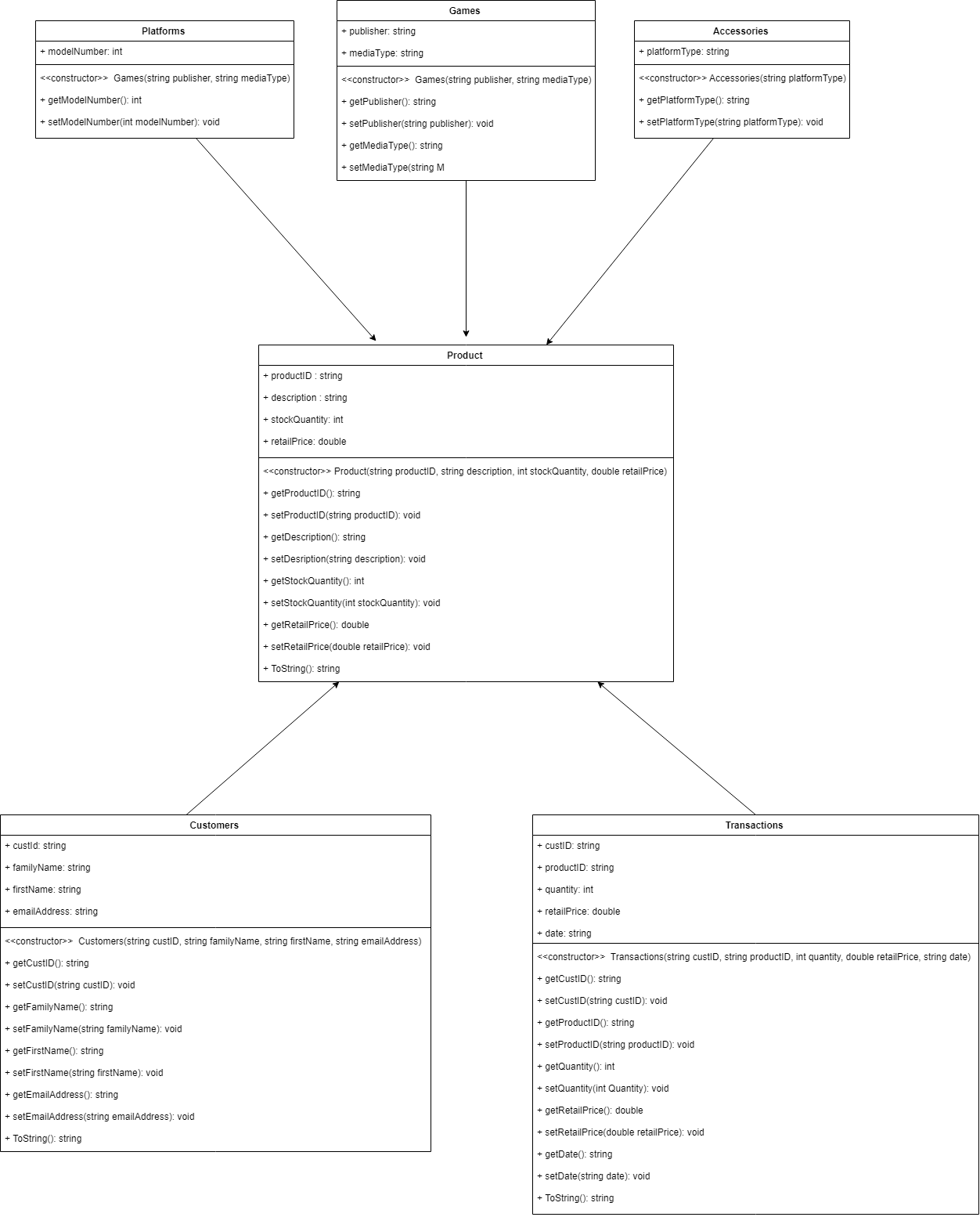
## Read binary file



## Write binary file



## Class diagrams



# Test Data and Evidence

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expected Outcome | Evidence |
| Case 1: | Add Product | Data added to List<Product> then displayed in Product Output |  |
| Case 2: | Add Customers | Data added to List<Customers> then displayed in Customer Output |  |
| Case 3: | Select a record from both Product and Customers | Record from Product and Customers will auto fill appropriate textbox’s in Transaction info |  |
| Case 4: | Input Quantity and Date then Press Add | Data is added to List<Transactions> then displayed in Transaction Output |  |
| Case 5: | Select Transaction Record | This will not only select the Transaction record but also the appropriate record in both Product and Customers |  |
| Case 6: | Delete text box’s | Click the ProductID textbox to clear all Item Info textbox’s. Click the CustID textbox to clear all Customer Info texbox’s |  |
| Case 7: | Save Data | Close program to save all data to .dat files |  |

# Code

//product, customer and transaction files .dat

string productFile = "product.dat";

string customerFile = "customer.dat";

string transactionFile = "transactions.dat";

//List<T> for product, customer and transaction

List<Product> productList = new List<Product>();

List<Customers> customerList = new List<Customers>();

List<Transactions> transactionList = new List<Transactions>();

//Add button which calls upon DisplayProducts method which saves all data to a List<T> structure and displays ProductID and Description in the listbox

private void btnProductsAdd\_Click(object sender, EventArgs e)

{

//If text any of the text boxs are empty display error message

if (String.IsNullOrEmpty(tbItemProductID.Text))

{

MessageBox.Show("Please enter a ProductID", "Error", MessageBoxButtons.OK);

}

else if (string.IsNullOrEmpty(tbItemDescription.Text))

{

MessageBox.Show("Please enter a description", "Error", MessageBoxButtons.OK);

}

else if (string.IsNullOrEmpty(tbItemStockQuantity.Text))

{

MessageBox.Show("Please enter Stock Quantity", "Error", MessageBoxButtons.OK);

}

else if (string.IsNullOrEmpty(tbItemRetailPrice.Text))

{

MessageBox.Show("Please enter Retail Price", "Error", MessageBoxButtons.OK);

}

else

{

try

{

if (!(tbGamesMediaType.Text == "" && tbGamesPublisher.Text == "") && (tbPlatformsModelNumber.Text == "" && tbAccessoriesPlatformType.Text == ""))

{

Games newGames = new Games();

newGames.setProductID(int.Parse(tbItemProductID.Text));

newGames.setDescription(tbItemDescription.Text);

newGames.setStockQuantity(int.Parse(tbItemStockQuantity.Text));

newGames.setRetailPrice(double.Parse(tbItemRetailPrice.Text));

newGames.setPublisher(tbGamesPublisher.Text);

newGames.setMediaType(tbGamesMediaType.Text);

productList.Add(newGames);

}

else if (!(tbPlatformsModelNumber.Text == "") && (tbGamesMediaType.Text == "" && tbGamesPublisher.Text == "" && tbAccessoriesPlatformType.Text == ""))

{

Platforms newPlatforms = new Platforms();

newPlatforms.setModelNumber(int.Parse(tbPlatformsModelNumber.Text));

newPlatforms.setProductID(int.Parse(tbItemProductID.Text));

newPlatforms.setDescription(tbItemDescription.Text);

newPlatforms.setStockQuantity(int.Parse(tbItemStockQuantity.Text));

newPlatforms.setRetailPrice(double.Parse(tbItemRetailPrice.Text));

productList.Add(newPlatforms);

}

else if (!(tbAccessoriesPlatformType.Text == "") && (tbGamesMediaType.Text == "" && tbGamesPublisher.Text == "" && tbPlatformsModelNumber.Text == ""))

{

Accessories newAccessories = new Accessories();

newAccessories.setProductID(int.Parse(tbItemProductID.Text));

newAccessories.setDescription(tbItemDescription.Text);

newAccessories.setStockQuantity(int.Parse(tbItemStockQuantity.Text));

newAccessories.setRetailPrice(double.Parse(tbItemRetailPrice.Text));

newAccessories.setPlatformType(tbAccessoriesPlatformType.Text);

productList.Add(newAccessories);

}

else

{

MessageBox.Show("To Much Information", "Error", MessageBoxButtons.OK);

}

}

//If any of the text boxes are the wrong variable type display error message

catch

{

MessageBox.Show("Incompatible Variable Type", "Try Again", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

DisplayProducts();

Clear();

}

}

//Method to display Products in ListBox

public void DisplayProducts()

{

lstProductsOutput.Items.Clear();

foreach (Product p in productList)

{

lstProductsOutput.Items.Add(p.ToString());

}

}

//Method that clears all text boxes

public void Clear()

{

tbItemProductID.Clear();

tbItemDescription.Clear();

tbItemStockQuantity.Clear();

tbItemRetailPrice.Clear();

tbGamesPublisher.Clear();

tbGamesMediaType.Clear();

tbPlatformsModelNumber.Clear();

tbAccessoriesPlatformType.Clear();

tbCustomerInfoCustID.Clear();

tbCustomerInfoFamilyName.Clear();

tbCustomerInfoFirstName.Clear();

tbCustomerInfoEmail.Clear();

tbTransactionInfoCustID.Clear();

tbTransactionInfoProductID.Clear();

tbTransactionInfoQuantity.Clear();

tbTransactionInfoRetailPrice.Clear();

tbTransactionInfoDate.Clear();

}

//Add button which calls upon DisplayCustomers method which saves all data to a List<T> structure and displays CustID, Family Name and First Name in the listbox

private void btnCustomersAdd\_Click(object sender, EventArgs e)

{

//If text any of the text boxs are empty display error message

if (string.IsNullOrEmpty(tbCustomerInfoCustID.Text))

{

DialogResult result = MessageBox.Show("Generate a default customer?", "Error", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

Customers newCustomers = new Customers();

newCustomers.setCustID("C000");

newCustomers.setFamilyName("unknown");

newCustomers.setFirstName("unknown");

newCustomers.setEmailAddress("unknown");

customerList.Add(newCustomers);

DisplayCustomers();

}

else

{

//Nothing

}

}

else if (string.IsNullOrEmpty(tbCustomerInfoFamilyName.Text))

{

MessageBox.Show("Please enter Family Name", "Error", MessageBoxButtons.YesNo);

}

else if (string.IsNullOrEmpty(tbCustomerInfoFirstName.Text))

{

MessageBox.Show("Please enter First Name", "Error", MessageBoxButtons.YesNo);

}

else if (string.IsNullOrEmpty(tbCustomerInfoEmail.Text))

{

MessageBox.Show("Please enter Email", "Error", MessageBoxButtons.YesNo);

}

else

{

Customers newCustomers = new Customers();

newCustomers.setCustID(tbCustomerInfoCustID.Text);

newCustomers.setFamilyName(tbCustomerInfoFamilyName.Text);

newCustomers.setFirstName(tbCustomerInfoFirstName.Text);

newCustomers.setEmailAddress(tbCustomerInfoEmail.Text);

customerList.Add(newCustomers);

DisplayCustomers();

Clear();

}

}

//Method to display Customers in ListBox

public void DisplayCustomers()

{

lstCustomersOutput.Items.Clear();

for (int i = 0; i < customerList.Count; i++)

{

if (customerList[i] != null)

{

lstCustomersOutput.Items.Add(customerList[i]);

}

}

}

//Save Methods for Product, Customers and Transactions

public void SaveProducts(string filename)

{

try

{

using (Stream fileStream = File.Open(filename, FileMode.Create))

{

BinaryFormatter serializer = new BinaryFormatter();

serializer.Serialize(fileStream, productList);

fileStream.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex, "Exception Throw", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void SaveCustomers(string filename)

{

try

{

using (Stream fileStream = File.Open(filename, FileMode.Create))

{

BinaryFormatter serializer = new BinaryFormatter();

serializer.Serialize(fileStream, customerList);

fileStream.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex, "Exception Throw", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void SaveTransactions(string filename)

{

try

{

using (Stream fileStream = File.Open(filename, FileMode.Create))

{

BinaryFormatter serializer = new BinaryFormatter();

serializer.Serialize(fileStream, transactionList);

fileStream.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex, "Exception Throw", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//Load Methods for Product, Customers and Transactions

public void OpenProducts(string filename)

{

try

{

using (Stream fileStream = File.OpenRead(filename))

{

BinaryFormatter deserializer = new BinaryFormatter();

productList = (List<Product>)deserializer.Deserialize(fileStream);

fileStream.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex, "Exception Throw", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void OpenCustomers(string filename)

{

try

{

using (Stream fileStream = File.OpenRead(filename))

{

BinaryFormatter deserializer = new BinaryFormatter();

customerList = (List<Customers>)deserializer.Deserialize(fileStream);

fileStream.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex, "Exception Throw", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void OpenTransactions(string filename)

{

try

{

using (Stream fileStream = File.OpenRead(filename))

{

BinaryFormatter deserializer = new BinaryFormatter();

transactionList = (List<Transactions>)deserializer.Deserialize(fileStream);

fileStream.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex, "Exception Throw", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//Add button which calls upon DisplayTransactions method which saves all data to a List<T> structure and displays CustID, ProductID, Quantity, Retail Price and Date in the listbox

private void btnTransactionsAdd\_Click(object sender, EventArgs e)

{

if (String.IsNullOrEmpty(tbTransactionInfoQuantity.Text))

{

MessageBox.Show("Please enter quantity", "Error", MessageBoxButtons.OK);

}

else if (string.IsNullOrEmpty(tbTransactionInfoDate.Text))

{

MessageBox.Show("Please enter a date", "Error", MessageBoxButtons.OK);

}

else

{

try

{

Transactions newTransactions = new Transactions();

newTransactions.setCustID(tbTransactionInfoCustID.Text);

newTransactions.setProductID(int.Parse(tbTransactionInfoProductID.Text));

newTransactions.setQuantity(int.Parse(tbTransactionInfoQuantity.Text));

newTransactions.setRetailPrice(double.Parse(tbTransactionInfoRetailPrice.Text));

newTransactions.setDate(tbTransactionInfoDate.Text);

transactionList.Add(newTransactions);

}

catch

{

MessageBox.Show("Incompatible Variable Type", "Try Again", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

DisplayTransactions();

}

}

//Method to display Transactions in ListBox

public void DisplayTransactions()

{

lstTransactionsOutput.Items.Clear();

for (int i = 0; i < transactionList.Count; i++)

{

if (transactionList[i] != null)

{

lstTransactionsOutput.Items.Add(transactionList[i]);

}

}

}

//When a item in the Product ListBox is selected the proper text boxes are filled

private void lstProductsOutput\_SelectedIndexChanged(object sender, EventArgs e)

{

int index = lstProductsOutput.SelectedIndex;

if (index == -1)

{

return;

}

// Select item from Output box

Games game;

Platforms platform;

Accessories accessorie;

if (productList[index] is Games)

{

ClearProducts();

game = (Games)productList[index];

tbItemProductID.Text = game.getProductID().ToString();

tbTransactionInfoProductID.Text = game.getProductID().ToString();

tbItemDescription.Text = game.getDescription();

tbItemStockQuantity.Text = game.getStockQuantity().ToString();

tbItemRetailPrice.Text = game.getRetailPrice().ToString();

tbTransactionInfoRetailPrice.Text = game.getRetailPrice().ToString();

tbGamesPublisher.Text = game.getPublisher();

tbGamesMediaType.Text = game.getMediaType();

}

else if (productList[index] is Platforms)

{

ClearProducts();

platform = (Platforms)productList[index];

tbItemProductID.Text = productList[index].getProductID().ToString();

tbTransactionInfoProductID.Text = productList[index].getProductID().ToString();

tbItemDescription.Text = productList[index].getDescription();

tbItemStockQuantity.Text = productList[index].getStockQuantity().ToString();

tbItemRetailPrice.Text = productList[index].getRetailPrice().ToString();

tbTransactionInfoRetailPrice.Text = platform.getRetailPrice().ToString();

tbPlatformsModelNumber.Text = platform.getModelNumber().ToString();

}

else if (productList[index] is Accessories)

{

ClearProducts();

accessorie = (Accessories)productList[index];

tbItemProductID.Text = productList[index].getProductID().ToString();

tbTransactionInfoProductID.Text = productList[index].getProductID().ToString();

tbItemDescription.Text = productList[index].getDescription();

tbItemStockQuantity.Text = productList[index].getStockQuantity().ToString();

tbItemRetailPrice.Text = productList[index].getRetailPrice().ToString();

tbAccessoriesPlatformType.Text = accessorie.getPlatformType();

tbTransactionInfoRetailPrice.Text = accessorie.getRetailPrice().ToString();

}

}

//When a item in the Customer ListBox is selected the proper text boxes are filled

private void lstCustomersOutput\_SelectedIndexChanged(object sender, EventArgs e)

{

int index = lstCustomersOutput.SelectedIndex;

if (index == -1)

{

MessageBox.Show("Please select item");

return;

}

tbCustomerInfoCustID.Text = customerList[index].getCustID();

tbTransactionInfoCustID.Text = customerList[index].getCustID();

tbCustomerInfoFamilyName.Text = customerList[index].getFamilyName();

tbCustomerInfoFirstName.Text = customerList[index].getFirstName();

tbCustomerInfoEmail.Text = customerList[index].getEmailAddress();

}

//Clear method to just clear text boxes associated with product

public void ClearProducts()

{

tbItemProductID.Clear();

tbItemDescription.Clear();

tbItemStockQuantity.Clear();

tbItemRetailPrice.Clear();

tbGamesPublisher.Clear();

tbGamesMediaType.Clear();

tbPlatformsModelNumber.Clear();

tbAccessoriesPlatformType.Clear();

}

//When ProductID text box is double clicked it will clear all products

private void tbItemProductID\_DoubleClick(object sender, EventArgs e)

{

ClearProducts();

}

//When CustId text box is click it will text boxs associated with Customer

private void tbCustomerInfoCustID\_DoubleClick(object sender, EventArgs e)

{

ClearCustomers();

}

//Method to clear all Customer Text Boxs

private void ClearCustomers()

{

tbCustomerInfoCustID.Clear();

tbCustomerInfoFamilyName.Clear();

tbCustomerInfoFirstName.Clear();

tbCustomerInfoEmail.Clear();

}

//When a item in the Transaction ListBox is selected the proper text boxes are filled and both the appropiate records from the Products and Customer ListBoxs are selected

private void LstTransactionsOutput\_SelectedIndexChanged(object sender, EventArgs e)

{

int index = lstTransactionsOutput.SelectedIndex;

if (index == -1)

{

MessageBox.Show("Please select item");

return;

}

tbTransactionInfoCustID.Text = transactionList[index].getCustID();

tbTransactionInfoProductID.Text = transactionList[index].getProductID().ToString();

tbTransactionInfoQuantity.Text = transactionList[index].getQuantity().ToString();

tbTransactionInfoRetailPrice.Text = transactionList[index].getRetailPrice().ToString();

tbTransactionInfoDate.Text = transactionList[index].getDate();

tbCustomerInfoCustID.Text = customerList[index].getCustID();

tbCustomerInfoFamilyName.Text = customerList[index].getFamilyName();

tbCustomerInfoFirstName.Text = customerList[index].getFirstName();

tbCustomerInfoEmail.Text = customerList[index].getEmailAddress();

Games game;

Platforms platform;

Accessories accessorie;

if (productList[index] is Games)

{

ClearProducts();

game = (Games)productList[index];

tbItemProductID.Text = game.getProductID().ToString();

tbTransactionInfoProductID.Text = game.getProductID().ToString();

tbItemDescription.Text = game.getDescription();

tbItemStockQuantity.Text = game.getStockQuantity().ToString();

tbItemRetailPrice.Text = game.getRetailPrice().ToString();

tbGamesPublisher.Text = game.getPublisher();

tbGamesMediaType.Text = game.getMediaType();

}

else if (productList[index] is Platforms)

{

ClearProducts();

platform = (Platforms)productList[index];

tbItemProductID.Text = productList[index].getProductID().ToString();

tbTransactionInfoProductID.Text = productList[index].getProductID().ToString();

tbItemDescription.Text = productList[index].getDescription();

tbItemStockQuantity.Text = productList[index].getStockQuantity().ToString();

tbItemRetailPrice.Text = productList[index].getRetailPrice().ToString();

tbPlatformsModelNumber.Text = platform.getModelNumber().ToString();

}

else if (productList[index] is Accessories)

{

ClearProducts();

accessorie = (Accessories)productList[index];

tbItemProductID.Text = productList[index].getProductID().ToString();

tbTransactionInfoProductID.Text = productList[index].getProductID().ToString();

tbItemDescription.Text = productList[index].getDescription();

tbItemStockQuantity.Text = productList[index].getStockQuantity().ToString();

tbItemRetailPrice.Text = productList[index].getRetailPrice().ToString();

tbAccessoriesPlatformType.Text = accessorie.getPlatformType();

}

//Selects Product Record

lstProductsOutput.SelectedIndex = index;

lstProductsOutput.Focus();

//Selects Customer Record

lstCustomersOutput.SelectedIndex = index;

lstCustomersOutput.Focus();

}

//Loads .dat files

private void Form1\_Load(object sender, EventArgs e)

{

try

{

//If product file exists then open and display products

if (File.Exists(productFile))

{

OpenProducts(productFile);

DisplayProducts();

}

else

{

MessageBox.Show(productFile + " Does Not Exist", "Error", MessageBoxButtons.OK);

}

//If customer file exists then open and dispay customers

if (File.Exists(customerFile))

{

OpenCustomers(customerFile);

DisplayCustomers();

}

else

{

MessageBox.Show(customerFile + " Does Not Exist", "Error", MessageBoxButtons.OK);

}

//if transaction file exists open and display transactions

if (File.Exists(transactionFile))

{

OpenTransactions(transactionFile);

DisplayTransactions();

}

else

{

MessageBox.Show(transactionFile + " Does Not Exist", "Error", MessageBoxButtons.OK);

}

}

catch (IOException x)

{

MessageBox.Show("Exeption: " + x, "Exception Throw", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//Saves Products, Customer and Transactions to .dat file

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

//Saving product to product file when form closes

SaveProducts(productFile);

//Saving customer to customer file when form closes

SaveCustomers(customerFile);

//Saving transactions to transaction file when form closes

SaveTransactions(transactionFile);

}

}

}