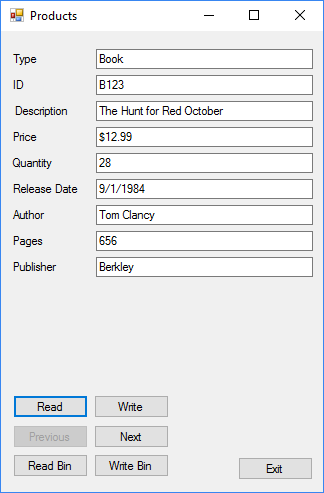
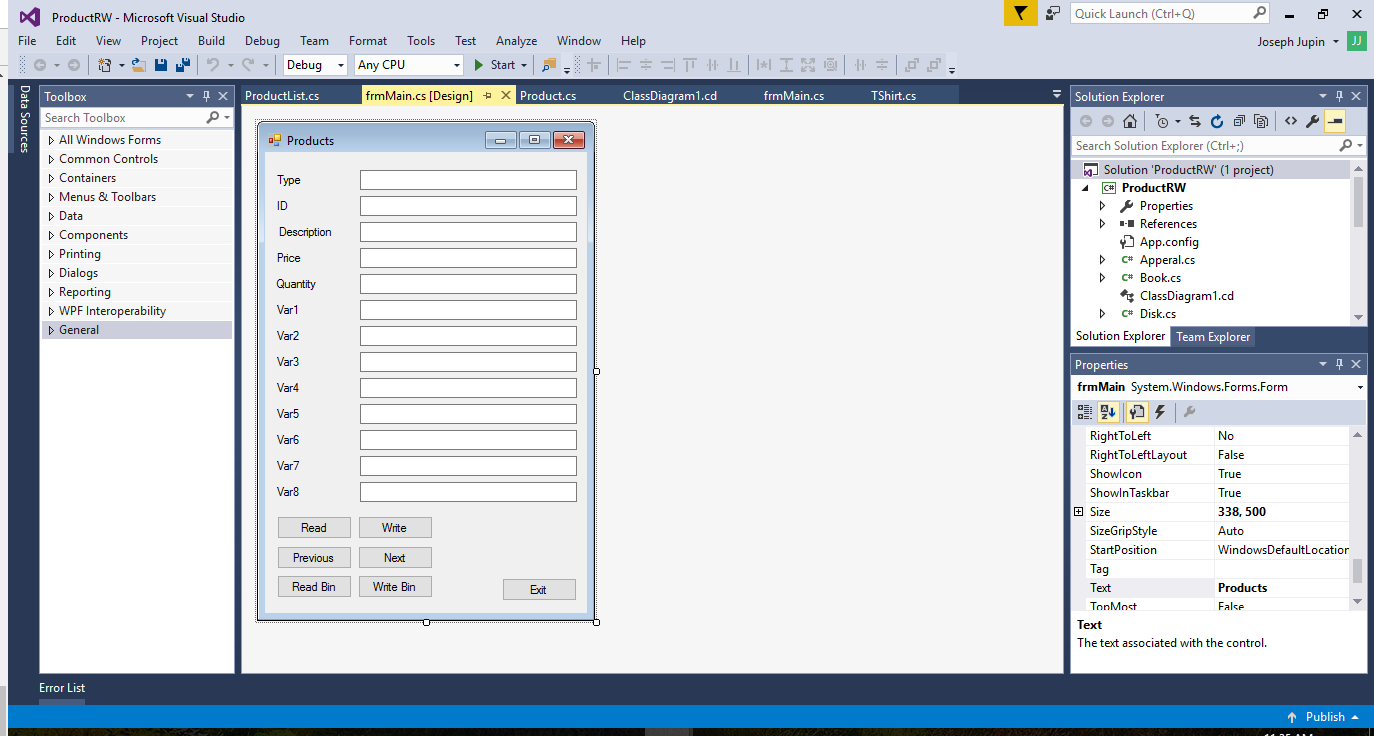
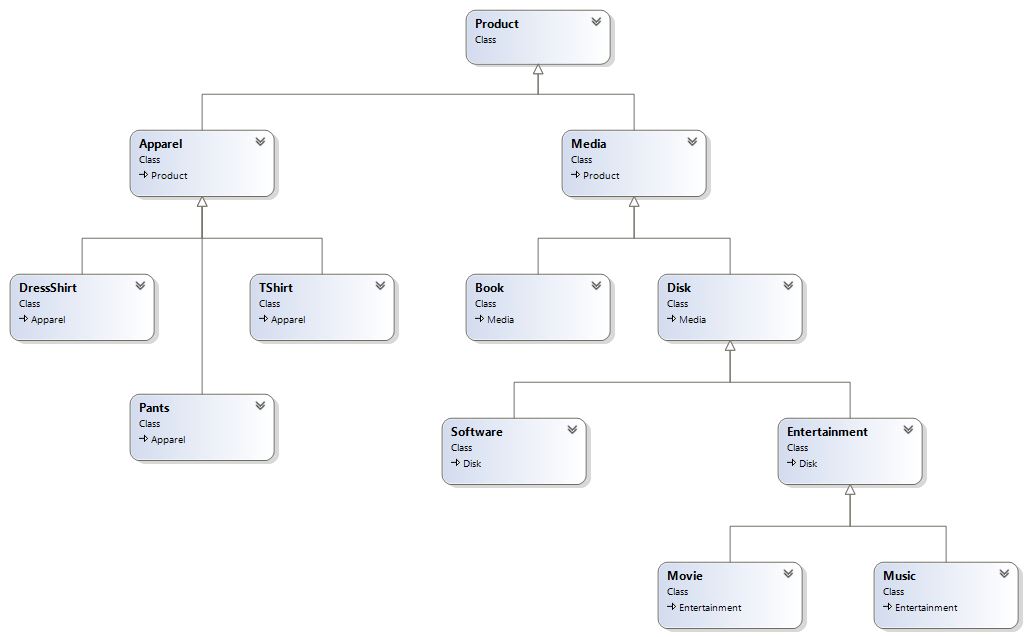
CIS 3309 – Product Object Problem

This project requires product data to be read from a file and inserted into a Product object, which will be stored in a list and displayed in a Form. The first task is to create the project. Create a new project named “ProductRWYourLastNameFirstInitial”. Next, Create the following form (left image):

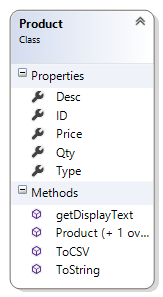


Give all objects on the form meaningful names. For example, the Type Label is named lblType and its corresponding Textbox is named txtType. Do this before creating events for any of the objects on the form. This will ensure that the event handlers have similar name to the objects that define them. The first five Label/Textbox pairs are the Product data that are included in all Product objects. The remaining pairs are variable and will have properties that change as the Product changes. Consider the example in the second form above. This example shows the configuration of the form after a book is displayed. Notice that 4 of the 8 variable pairs are hidden and that the Text property of the Labels has changed to reflect the data contained in the Textboxes. Also, note that the Product properties are fixed.

The next step is to create the Product object hierarchy. The Graph below shows this hierarchy. The actual real world products are in the leaf nodes of the graph. They are: DressShirt, TShirt, Pants, Software, Book, Music and Movie Products. Create the hierarchy with the necessary inheritance references.



After creating the hierarchy, create the Product class. Remember that the class is the definition and that the object is an instance of the class in memory. The UML below shows the properties (auto-implemented) and methods in the Product class. Desc is a string describing the Product. ID is a string with the Product’s unique ID. Price is a double representing the cost of the Product. Qty is an integer representing the quantity in stock. Type is a string with the type of object as described for the leaf nodes. The getDisplayText accepts a separator and returns a string with the Products data as is used in the text. There are two constructors: a default constructor and one that accepts all of the Product data and stores them in the Products fields. The ToCSV method sends a comma as a separator to getDisplayText and returns a CSV string with all the Product data. The ToString passes a newline to getDisplayText and returns a string with all the Product data, line-by-line. See sample output below.



**Sample of ToCSV output**

Software,S123,Starcraft,16.99,12,CD,1,10/24/1999,Game

**Sample of ToString output**

Software

S123

Starcraft

16.99

12

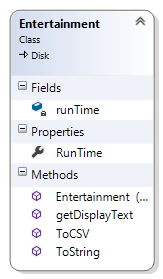
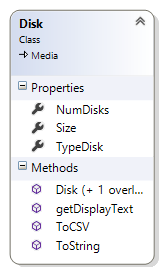
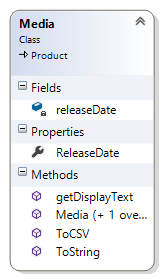
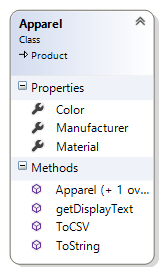
CD

1

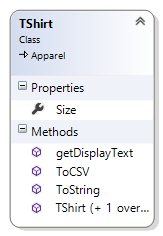
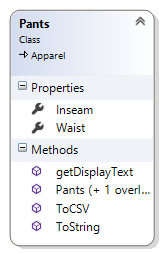
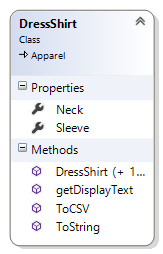
10/24/1999

Game

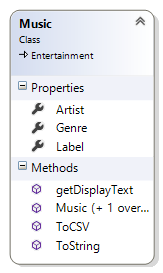
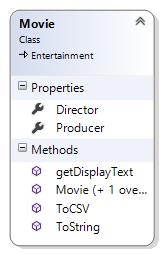
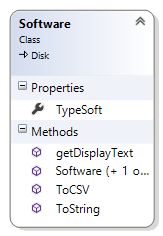
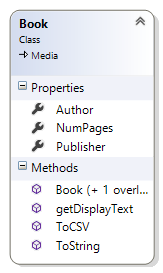
The Apparel, Media, Disk and Entertainment inner nodes of the hierarchy are shown below. The data are: Color is a string with the color of the Product. Manufacturer is a string with the manufacturer’s name. Material is the Product material (cotton, etc.). The relraseDate field contains a DateTime object. The property should hide this from the user by treating it as a string outside the class. NumDisks is an integer representing the number data disks in the package. Size is an integer representing the amount of data on the disk. TypeDisk is a string with value CD, DVD or BluRay. The runTime field is a TimeSpan object containing the runtime of the Entertainment item. The TimeSpan object should be hidden outside the class by accepting a string and returning a string from outside the class. The methods are similar to those described in the Product class.



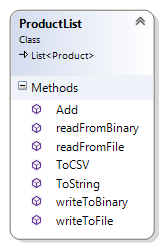
The DressShirt, Pants and TShirt Apparel classes are shown below. Neck and Sleeve are integers representing the neck size and sleeve length of the shirt. Inseam and Waist are integers representing the waist size and inseam length. Size is a string with the shirt size (S, M, l, XL, etc.). The methods are similar to those described in the Product class.



The Book, Software, Movie, and Music Products are shown below. Author and Publisher are strings and NumPages is an integer. TypeSoft is a string representing the type of Software (Game, Office, etc.). Director and Producer are strings. Artist, Genre and Label are strings. The methods are similar to those described in the Product class.



Next, write the ProductList class. The UML below shows the Product List class (remember the use of the this keyword). Notice that it inherits the List class. This saves a lot of coding. The Add method adds a Product to the List. After creating the class, write the readFromFile method to read Products from a text file into the ProductList. Open the Product.csv file. The first field is the type of product. Use this to decide which type of object to instantiate and insert. Hint: Use an if statement to call the appropriate constructor and then insert it into the list. Write the ToString and ToCSV methods to write all Products to their respective strings. Hint: Use each Product’s same name methods to do this with less code. Code the writeToFile method to write the data to a text file with the same format as the Product.csv file (Use each Products ToCSV method). I suggest naming the output file Output.csv. This will protect the Products.csv file. Implement the writeToBinary method to write the Products as string objects (Use each products ToCSV method). Last, code the readFromBinary to read the binary string to the ProductList.



Code the methods to draw the objects and data on the Form. The code below shows how to draw the forms and populate them with data. (This is free stuff to use in your project! You can copy-and-paste!)

public void drawBook()

{

Book b = (Book) p;

drawSet(true, true, true, true, false, false, false, false);

drawMedia();

lblVar2.Text = "Author";

txtVar2.Text = b.Author;

lblVar3.Text = "Pages";

txtVar3.Text = b.NumPages.ToString();

lblVar4.Text = "Publisher";

txtVar4.Text = b.Publisher;

}

The drawBook method casts the current Product as a Book and copies the data to the form. The drawSet method shows and hides the Labels and Textboxes based on the Boolean data passed. The call in drawBook shows the first 4 and hides the last 4. The call to drawMedia method is shown below.

public void drawMedia()

{

Media m = (Media)p;

lblVar1.Text = "Release Date";

txtVar1.Text = m.ReleaseDate;

}

This method casts the Product as a Media object and prints its data to the form. The drawProduct is below.

public void drawProduct()

{

if(idx >=0 && idx < pl.Count)

{

p = pl.ElementAt(idx);

txtType.Text = p.Type;

txtID.Text = p.ID;

txtDesc.Text = p.Desc;

txtPrice.Text = p.Price.ToString("C");

txtQuantity.Text = p.Qty.ToString();

if (p.Type == "Book")

drawBook();

else if (p.Type == "Software")

drawSoftware();

else if (p.Type == "Music")

drawMusic();

else if (p.Type == "Movie")

drawMovie();

else if (p.Type == "Pants")

drawPants();

else if (p.Type == "TShirt")

drawTShirt();

else if (p.Type == "DressShirt")

drawDressShirt();

else

MessageBox.Show("Not found.");

}

}

This method is the last method called for all Product objects. The if statement ensures that the Product to be drawn is within array bounds. The idx variable is a global integer that contains the current Product to be drawn. The Product p and the ProductList pl are also have global scope.

Code the buttons. Use the included exe to see how these should work.

When completed, please test your project, close Visual Studio, compress your project and upload it to Blackboard.

Good luck!

# About the sample application.

Double-click on the exe file. Click the read button. Traverse to the last Product by clicking on the Next Button. Click the Write Button and check that the file was written properly. Click the Write Binary Button and check that the file was created. Click the Read Binary Button. This clears the ProductList and reads the data from the binary file into the List. Click Next to traverse to the last Product. Click Exit to close. You project should work similar to this application.