

Assignment 1

COMP 1004 – Rapid Application Development  
Assignment 1  
Weight: **15%** of your Final Mark

IDE/Language: Visual Studio 2013 .NET Framework 4.5, C#

Conditions

THIS IS A GROUP ASSIGNMENT, ONLY 2 MEMBERS PER GROUP. No individual assignments will be accepted. If you cannot find a partner, please email me at the earliest and I will try to pair you up.

Your submission must be “ENTIRELY” your work. Make sure you understand the academic policies and procedures as detailed on the college website at:

<http://www.georgianc.on.ca/admissions/policies-procedures>.

Pay special attention to warnings about cheating, plagiarism, and acceptable use of college computer resources.

If your submitted work is found out to have been re-produced from any human or non-human resource, for example, the web, I will strictly be following the procedure as in the above document.

When you are done, please zip up your ENTIRE project folder and submit the zipped file via the **Assignment 1** link provided on Blackboard. Name the zip file **your\_names.Assignment1.zip.**

Overview  
  
You have been approached by Robellus, a leading nationwide cellular provider, to produce an application for their employees to calculate the total cost of purchasing an iPhone for their customers, depending on the colour, capacity and model selected.  
  
In addition, they would like to upsell their accessories to the customer and offer a trade in credit to reduce the purchase cost of the phone.

Specifications - Costs  
  
Depending on the model selected, either an iPhone 6 or an iPhone 6s, and the capacity chosen, this will determine the base price of the phone:

**iPhone 6**

|  |  |
| --- | --- |
| **Capacity** | **Cost** |
| 16GB | $749 |
| 64GB | $859 |
| 128GB | $969 |

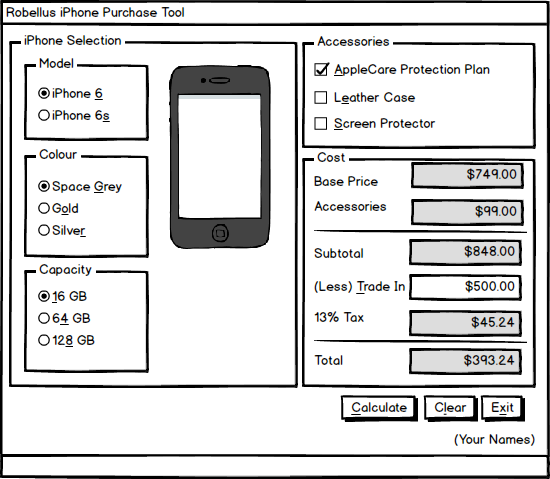
**iPhone 6s**

|  |  |
| --- | --- |
| **Capacity** | **Cost** |
| 16GB | $969 |
| 64GB | $1099 |
| 128GB | $1229 |

**Accessories**

|  |  |
| --- | --- |
| **Accessory** | **Cost** |
| AppleCare Protection Plan | $99 |
| Leather Case | $49 |
| Screen Protector | $29 |

Specifications – Form Layout  
  
Robellus had provided a sample layout as shown below. Add an appropriate heading.



Specifications – Form Behaviour  
  
Here are the behaviours and business rules that Robellus expects the form to operate:

1. When the form loads, the *starting default state* of the application is to have no radio buttons selected, no checkboxes checked, no iPhone picture shown, and no amounts displayed.
2. The form should have the minimize, maximize and the close controls. The maximize control should be disabled.
3. The form should display in the center of the screen when the application is run and should not be resizable.
4. Include a **text box** for the input of the **(Less) Trade In** allowance by the user. It should have a useful tool tip.
5. The **text** **controls** for the **Base Price,** **Accessories**, **Subtotal**, **13% Tax**, and **Total** should be disabled and not allow user input. Tab should not stop on these output boxes.
6. **Check boxes** will indicate if the buyer wants additional accessories: AppleCare Protection Plan, Leather Case, Screen Protector
7. **Group boxes** for the iPhone **Model**, **Colour**, **Capacity**, the parent **iPhone** **Selection**, **Accessories** and **Cost**
   1. The user can only choose between the two models in the **iPhone Model** group box
   2. The user can only choose between one of the three colours in the **Colour** group box
   3. The user can only choose between one of the 3 capacities in the **Capacity** group box
   4. The user can choose any number of selections or none in the **Accessories** group box
8. When one of the **colours** and one of the **models** are selected, an image of the iPhone **representing that combination** will show up beside both of the group boxes as indicated in the **Form Layout** mock-up.
9. For the **(Less) Trade-In** allowance value, show a default value of zero ($0.00) in the associated input text box when the form loads and after every clear.

1. The **Calculate** button, with a hot key of **(C)** when clicked or pressing **Enter**,validates the selection of the **iPhone** options, then if valid, displays and determines the values in the fields
2. First validate that a **model**, **colour**, and **capacity** have been selected by the user. If the user has not selected any or all of these, then show separate appropriate error messages. One error message must appear at a time if there are more than one errors in a logical sequence – error for model, colour and capacity in that order.
3. **Validate** the input value provided in the trade in input text box. The **(Less) Trade-In** (should be >=0, numeric and within ‘type’ limits)

Use a try-catch block and embedded ‘if’ to catch invalid data for this input. You should catch the exceptions (format, overflow, generic) in separate catch blocks. I want to see one error message at a time if there are multiple errors. Showing ‘multiple’ error messages to the user together is not a good idea.

Then (*if there are no errors from above*):

1. The text box labeled **Base Price** shows the cost of the iPhone selected
2. The text box labeled **Accessories** shows the sum of the prices of the selections made by the user in the **Accessories** group box. If no **Accessories** are selected, show a zero value.
3. Add the price of selected **Accessories** and the iPhone model and colour combination to the base price and display the result in the **Subtotal** control.
4. Calculate the **13% Tax** on the (subtotal- trade in) and display the result in the **13% Tax** control. (*Trade-in discounts are applied to the selling price before taxes are calculated*).
5. Display the final total= Subtotal – Less Trade In + Calculated Tax in the **Total** control.
6. All amounts displayed in the controls are to be **right-justified**, to **two-decimal places** with a **dollar currency sign**.
7. The **Clear** button, with a hot key of **(l)** when clicked or pressing **Esc**ape,resets the form to its *starting default state*.
8. The **Exit** button, with a hot key of **(x)**, when clicked will close the application.
9. All hot keys must be operational.
10. The tab order of the application is from top-down on the left first and then top-down on the right. So iPhone selection group box first, top to bottom and then Accessories group box, followed by Cost group box and then the buttons.

Evaluation

|  |  |
| --- | --- |
| **Requirement** | **Maximum Marks** |
| Code is commented well with names, date and purpose at head; every method/event has an explanatory comment about its functionality; comment wherever necessary in the main logic | 4 |
| Project should have a good meaningful name. Rename form and all programmable controls to well-chosen names. Variables names, controls and event/method names follow camel case naming convention and form name uses Pascal case convention. All identifier names must have class name/variable type attached | 4 |
| Named constants have been used in code; no magic numbers in the code | 2 |
| Declare variables where required (local or module). Use the correct type of variables | 2 |
| Interface as provided | 6 |
| Display form in center screen; cannot be resized or maximized; can be minimized and has the usual X for closing the form/application | 2 |
| Tabbing as required, only on input controls | 2 |
| Keyboard access key for trade in input is working; has a useful tool tip | 2 |
| Accept (Enter) and Cancel (Esc) buttons of the form are set to the Calculate and Clear buttons respectively | 1 |
| (Less) Trade In should show and use a default value of zero | 2 |
| Clear button resets form to *starting default state* | 2 |
| Exit button closes the application | 1 |
| (Less) Trade In must be numeric and >= 0 and within type limits. Separate and clear error messages are displayed one at a time for format, overflow and generic exceptions | 4 |
| Efficient, non-redundant code for the functionality of the Calculate button, including handling of missing iPhone model, colour and capacity and the correct picture display. Trade-in value should be validated (marks separate). Output correct and as required | 10 |
| **Total** | **44** |