



**C# PROGRAMMING**

(330)

**REGIONAL 2022**

**Production:**

 **Program 1: Currency Calculator \_\_\_\_\_\_\_\_\_\_ (370 points)**

 **TOTAL POINTS \_\_\_\_\_\_\_\_\_\_ (370 points)**

**Test Time: 90 minutes**

**Currency Calculator**

In this exercise you will create a C# Windows Form Application that can manipulate money in two different ways. On the left-hand side of the window the application will convert one type of currency into several other currency types (for example, US dollars to British pounds). On the right-hand side of the window the application will break the user given amount, rounded down to the nearest dollar, into the fewest number of US bills possible. On the bottom of the window a set of radio buttons let the user choose which currency to convert from and a “Convert” button that performs the above calculations.

In the example below the user enters 123.45 into the Dollars text box, selects the “Dollars” radio button, and clicks the “Convert” button. The user’s given dollar amount is then converted into Pounds, Euros, and Yen and those values are displayed in their respective text boxes. Additionally, the fewest number of US bills necessary to represent $123.45 rounded down to the nearest dollar (one $100 bill, one $20 bill, and three $1 bills) is calculated and displayed in the text boxes on the right.



**Requirements:**

1. You must create a C# Windows Form Application named CS \_330\_ContestantNumber, where ContestantNumber is your BPA assigned contestant number (including dashes). For example, CS\_123\_45\_6789\_0000.
2. Your contestant number must appear as a comment at the top of the main source code file.
3. The main form must not have minimize or maximize buttons in the Form Handle.
4. The main form must have text “Currency Calculator” for its title. The form must start in the center of the screen and must have a Fixed Dialog setting so the size cannot be adjusted.
5. The main form must contain the following items:
	1. Four labels, one for each currency type
	2. Four textboxes, one for each currency type
	3. Six labels, one for each US Dollar bill type
	4. Six textboxes, one for each US Dollar bill type
	5. Four radio buttons, one for each currency type

|  |  |  |
| --- | --- | --- |
| UI Object | Name | Text |
| Label | lblDollars | Dollars |
| Label | lblPounds | Pounds |
| Label | lblEuros | Euros |
| Label | lblYen | Yen |
| Textbox | txtDollars |  |
| Textbox | txtPounds |  |
| Textbox | txtEuros |  |
| Textbox | txtYen |  |
| Label | lblHundredBills | $100 |
| Label | lblFiftyBills | $50 |
| Label | lblTwentyBills | $20 |
| Label | lblTenBills | $10 |
| Label | lblFiveBills | $5 |
| Label | lblOneBills | $1 |
| Textbox | txtHundredBills |  |
| Textbox | txtFiftyBills |  |
| Textbox | txtTwentyBills |  |
| Textbox | txtTenBills |  |
| Textbox | txtFiveBills |  |
| Textbox | txtOneBills |  |
| Radio Button | rdoDollars  | Dollars |
| Radio Button | rdoPounds | Pounds |
| Radio Button | rdoEuros | Euros |
| Radio Button | rdoYen | Yen |
| Button | btnConvert | Convert |

1. When the program loads, the main window needs the following:
	1. The currency type text boxes and labels (Dollar, Pound, Euro, Yen) should be left aligned, with font size set to 20
	2. The US Dollar bill text boxes and labels (($100, $50, $20, $10, $5, $1) should be right aligned, with font size set to 20
	3. The currency type radio buttons and the “Convert” button should be center aligned, with the radio button’s font size set to 16 and the “Convert” button’s font size set to 20
	4. The “Convert” button must use a click event called “convertButton\_Click” which performs the following functions:
		1. Retrieves the user’s given amount from the text box specified by the user-selected radio button (if the “Dollars” radio button is selected, get the amount in dollarsBox)
			1. If the user enters an invalid value in a text box, a message box should appear alerting the user about the error (see end of instructions for reference images)
			2. If the user does not select one of the currency radio buttons, a message box should appear alerting the user about the error (see end of instructions for reference images)
		2. Convert the amount into the other three currency types, using the following currency rates. For example $123.45 multiplied by 0.76 is 93.82 Pounds and 789 Yen divided by 104.53 is $7.55
			1. 1 Dollar = 0.76 Pounds
			2. 1 Dollar = 0.84 Euros
			3. 1 Dollar = 104.53 Yen
		3. Write the newly calculated currency amounts to their respective text boxes. For example, the 93.82 Pounds calculated above should be set as the text for poundsBox
			1. Format the text so that is represents a valid currency amount, rounding to the nearest hundreds place (two decimal places)
		4. Calculate the **minimum** number of US bills necessary to represent the user’s given amount
			1. Round down the user’s given amount to the nearest dollar bill (round $123.45 down to $123.00)
		5. Write the newly calculated US bill counts to their respective text boxes.

**Sample Pictures:**



1: Example of the Message Box that should be displayed when the user inputs an invalid currency amount



2: Example of the Message Box that should be displayed when the user does not select a currency type radio button



3: User inputted 123.45 Euros, selected the "Euros" radio button, then clicked the "Convert" button.

You will have ninety (90) minutes to complete your work.

Your name and/or school name should not appear on any work you submit for grading. Remember to use your contestant # where appropriate.

Submit a copy your entire solution/project to the flash drive provided. You must submit your entire solution/project so that the graders may open your project to review the source code and/or build and execute your solution/project. **Submissions that do not contain source code will not be graded.**

Development Standards

* Standard name prefixes must be utilized for variables.
* All subroutines, functions, and methods must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any).

Your application will be scored on the following criteria:

**Solution and Project**

The project is present on the flash drive \_\_\_\_ 10 pts

The project is named according to the naming conventions \_\_\_\_ 10 pts

**Program Execution**

Code copied to USB drive and program runs from USB \_\_\_\_ 20 pts

*If the program does not execute, then the remaining items in the program executive section receive a score of zero.*

The title of the form is “Currency Calculator “ \_\_\_\_ 10 pts

The main form does not have minimize or maximize buttons

in the Form Handle \_\_\_\_ 10 pts

The form’s start position is the center of the screen and it cannot be resized \_\_\_\_ 10 pts

The window has the correct buttons, text boxes, radio buttons and

labels based upon the Figures \_\_\_\_ 20 pts

Clicking “Convert” button performs both calculations \_\_\_\_ 30 pts

Currency Converter: calculations are performed properly \_\_\_\_ 30 pts

Currency Converter: amounts are rounded to nearest hundredths place \_\_\_\_ 10 pts

Currency Converter: Message Box is shown for invalid currency amount \_\_\_\_ 20 pts

Currency Converter: Message Box is shown for no radio button selected \_\_\_\_ 20 pts

Bill Breakdown: calculations are performed properly \_\_\_\_ 20 pts

Bill Breakdown: dollar amount is rounded down to nearest dollar \_\_\_\_ 20 pts

**Source Code Review**

Code is commented at the top, for each function, and as needed \_\_\_\_ 10 pts

Code uses reasonable and consistent variable naming conventions \_\_\_\_ 10 pts

Currency Converter: code for “convertButton\_Click” method is present \_\_\_\_ 20 pts

Currency Converter: code for displaying Message Boxes is present \_\_\_\_ 10 pts

Currency Converter: code for calculating currency values is present \_\_\_\_ 20 pts

Bill Breakdown: code for rounding dollar amount down is present \_\_\_\_ 20 pts

Bill Breakdown: code for finding minimum number of each bill is present \_\_\_\_ 20 pts

Bill Breakdown: Code to stop calculation until proper data is present \_\_\_\_ 20 pts

**Total Points:** \_\_\_\_ **/ 370 pts**

**Potential Solution**

/\*

 \* CS\_330\_01\_2345\_6789

 \*/

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace CSharpProgrammingBPA

{

 public partial class CurrencyConverterForm : Form

 {

 private static readonly double dollarToPound = 0.76;

 private static readonly double dollarToEuro = 0.84;

 private static readonly double dollarToYen = 104.53;

 private static readonly int[] dollarBills = { 100, 50, 20, 10, 5, 1 };

 public CurrencyConverterForm()

 {

 InitializeComponent();

 }

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

 {

 var convertButton = sender as Button;

 if (convertButton != null)

 {

 var currencyInDollars = getSelectedCurrencyAsDollars();

 if (currencyInDollars == null) return;

 // Convert the dollar amount into each type of currency

 dollarsBox.Text = currencyInDollars.Value.ToString("#.##");

 poundsBox.Text = (currencyInDollars.Value \* dollarToPound).ToString("#.##");

 eurosBox.Text = (currencyInDollars.Value \* dollarToEuro).ToString("#.##");

 yenBox.Text = (currencyInDollars.Value \* dollarToYen).ToString("#.##");

 // Break the dollar amount into the smallest amount of dollar bills possible

 var dollarBillCounts = new List<int>();

 var currentAmount = currencyInDollars.Value;

 foreach (var bill in dollarBills)

 {

 var billCount = maximumNumberOfBills(currentAmount, bill);

 dollarBillCounts.Add(billCount);

 currentAmount -= bill \* billCount;

 }

 hundredBox.Text = dollarBillCounts[0].ToString();

 fiftyBox.Text = dollarBillCounts[1].ToString();

 twentyBox.Text = dollarBillCounts[2].ToString();

 tenBox.Text = dollarBillCounts[3].ToString();

 fiveBox.Text = dollarBillCounts[4].ToString();

 oneBox.Text = dollarBillCounts[5].ToString();

 }

 }

 /\*

 \* Find the maximum number of bills that a dollar amount can be broken into. For example, the amount 3.45 can be broken into three one dollar bills (dollarBill = 1)

 \*/

 private int maximumNumberOfBills(double amount, int dollarBill)

 {

 var billCount = 0;

 while (amount >= dollarBill)

 {

 amount -= dollarBill;

 billCount++;

 }

 return billCount;

 }

 /\*

 \* Retrieve the user given currency amount from the selected currency type, then convert into dollars

 \*/

 private double? getSelectedCurrencyAsDollars()

 {

 double amount;

 if (dollarsRadioButton.Checked)

 {

 if (double.TryParse(dollarsBox.Text, out amount)) return amount;

 else

 {

 MessageBox.Show("Cannot parse the given dollars amount, make sure a valid dollar amount was typed");

 return null;

 }

 }

 else if (poundsRadioButton.Checked)

 {

 if (double.TryParse(poundsBox.Text, out amount)) return amount / dollarToPound;

 else

 {

 MessageBox.Show("Cannot parse the given pounds amount, make sure a valid pound amount was typed");

 return null;

 }

 }

 else if (eurosRadioButton.Checked)

 {

 if (double.TryParse(eurosBox.Text, out amount)) return amount / dollarToEuro;

 else

 {

 MessageBox.Show("Cannot parse the given euros amount, make sure a valid euro amount was typed");

 return null;

 }

 }

 else if (yenRadioButton.Checked)

 {

 if (double.TryParse(yenBox.Text, out amount)) return amount / dollarToYen;

 else

 {

 MessageBox.Show("Cannot parse the given yen amount, make sure a valid yen amount was typed");

 return null;

 }

 }

 else

 {

 MessageBox.Show("No currency type selected, please select which type of currency is given");

 return null;

 }

 }

 }

}