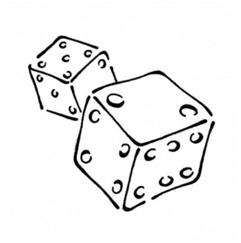
Business Application Programming INFO102

VICTORIA UNIVERSITY
OF WELLINGTON
Te Whare Wananga o te Upoko
o te Ika a Maui

School of Information Management (SIM)
Te Kura Whakaipurangi Korero

Assignment 2

Trimester 2 2011



INFO102 Dice Game

<u>Due Date</u>: **Monday 12 September 2011 - 5pm**

Overview

This C# assignment requires you to create a small prototype system for a dice game. A company is considering installing such an application on their employees' computer which they will be able to use during their break in order to relax.

The user for this system is the actual player which means that the interface must be attractive and easy for users to play.

Please note:

- *Read the whole assignment before you begin.*
- *The best source of information to help you complete this assignment are:*
    - o *Textbook: Bradley and Millspaugh, Chapters 1 to 6.*
    - o *Lectures 1 to 12.*
    - o *Workshops 1, 2, 3, and 4.*

Learning Objectives

In this assignment you will be showing an understanding of the following:

- C# built-in classes, objects, and methods.
- General methods.
- Interface design principles.
- Interface controls including menus, textboxes, combo boxes, labels, buttons, and picture boxes.
- Programming constructs: decision structures, property and variable, exception. handling, method definition and calls.
- Variables, constants, and calculations.
- Multiple forms, About forms, ColorDialog dialog box.
- Passing data between forms by creating class properties.

Warnings

1. You must write your program in C# 2008. Programs written in C# 2005 or C# 2010 will not be marked.
2. You are responsible for backing up your own work.
3. Plagiarism is not accepted in this course. Please read the course outline for guidelines on the University plagiarism policy.

Submission Instructions

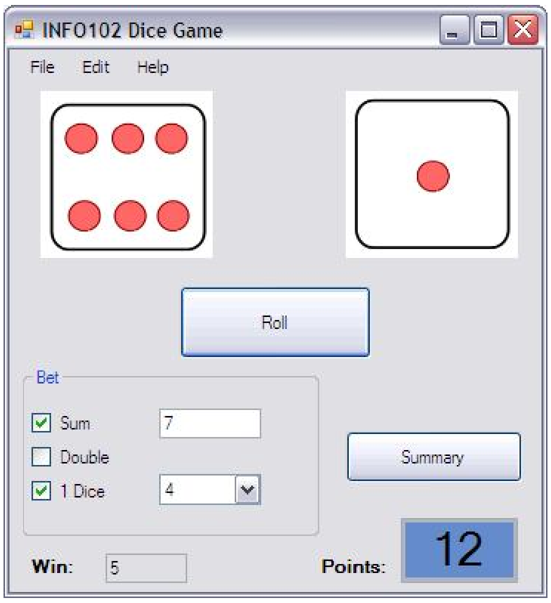
1. Name your project using this format: StudentID_DiceGame_A2 (e.g. 3000123456_DiceGame_A2).
2. You will need to zip your project folder and then use file transfer protocol (FTP) to submit the zipped folder to the SIM web server. Instructions for zipping project folders and for using FTP are available on Blackboard. Ensure you know how to do this because if you do not submit your assignment correctly we cannot mark it.
3. The submission system closes at 5pm on the due date. You are strongly advised to submit your work one or two hours before this time.

Technical Notes

1. You can use C# Visual Express 2008 or Visual C#.Net 2008 Professional edition to create this application. Both versions are suitable for this assignment. Instructions for downloading these products for home use are available on Blackboard. For downloading at home you will find the Express edition is smaller.

Instructions – Dice game system

1. You are required to create a C# application that is a prototype for a dice game application.



- [✓] Sum: 7
- [ ] Double
- [✓] 1 Dice: 4

Win: 5
Points: 12

2. The goal of the game consists of scoring the highest possible score based on guessing the outcomes from consecutive 2-dice rolls.

3. Each game starts will a total of 10 points. If a user loses all his/her points, then a message box indicates that the game is over and that the user has no points left. The application then closes.

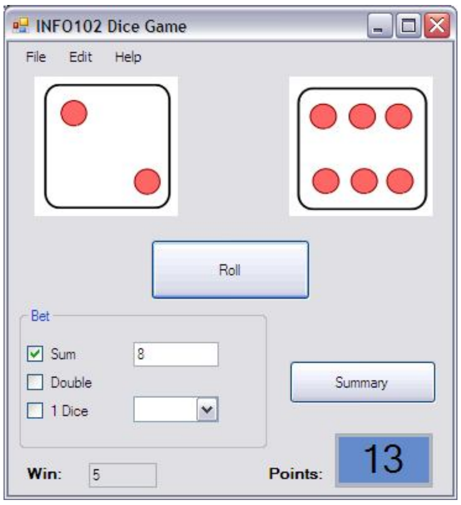
4. The user can base his/her guess on three possible options: guessing the right sum, guessing the outcome will be a double, or else guessing one of the two dice. The user is allowed to use one of the options, two options, or even the three options at the same time for each roll of the dice.

   If the user correctly guesses:
   - a sum: he/she wins 5 points.
   - a double: he/she wins 3 points.
   - one of the dice: he/she wins 1 point.

For each selected option which was wrong, the user looses one point.

Examples: Suppose the following configurations:

A. A user currently has 8 points. He/she selects one option: sum = 8



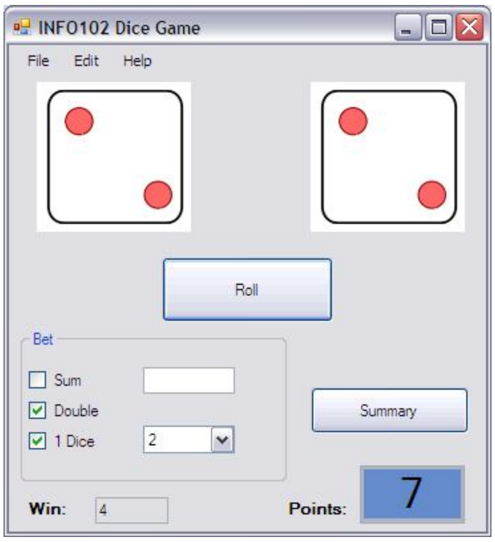
☑ Sum: 8
☐ Double
☐ 1 Dice

Win: 5

Points: 13

Since the user's guess was right, he/she wins 5 points and now has 13 points. If the sum of the dice had not been 8 then he/she would have lost one point, giving a total of points of 9.

B. A user currently has 3 points. He/she selects two options: double, and at least one dice is a 2.



Bet

☐ Sum

✓ Double

✓ 1 Dice 2

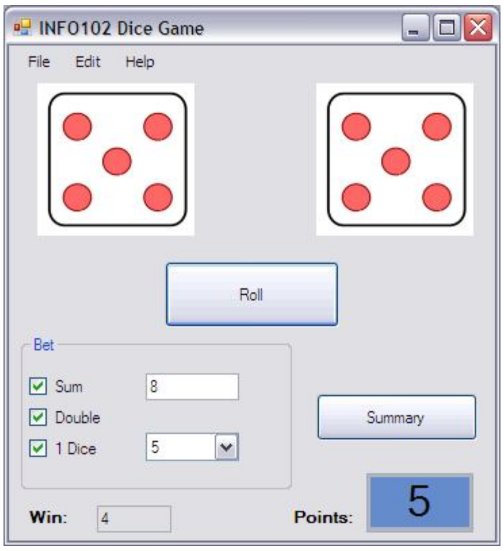
Win: 4

Points: 7

Both guesses were right, the user then wins 3 + 1 = 4 points (3 for the double and 1 for guessing that one dice is a 2).

Suppose the outcome had been a double 3. The user would have won 3 points for guessing a double <u>but</u> lost 1 point since there is no 2 in the outcome. In other words he would have won 2 points.

C. Suppose a user has one point. 3 options were selected: the sum is equal to 8, a double, and at least one dice is a 5.



Bet

✓ Sum 8

✓ Double

✓ 1 Dice 5

Win: 4

Points: 5

Two guesses were right: the double, and getting at least a 5. The user then wins 3 points for guessing a double, 1 point for getting at least a 5, and loses one point for guessing that the sum would be equal to 8 (which was not the case). The user then wins 3 + 1 – 1 = 3 points.

5. Here are some special cases that must be handled by the system:
    - If a user has 2 points left, he/she cannot select 3 options.
    - If a user has 1 point left, he/she can only use one option for the next roll.
    - If a user selects the sum check box and does not enter any value, a message pops up, asking the user to enter a value for the sum.
    - Similarly, if a user selects the 'one dice' check box and does not enter any value, a message pops up, asking the user to enter a value for the dice.
    - If a user enters a sum that does not make any sense (-15, 13, 9999), this is no special case. The user looses one point.
    - If a user does not select any option, he/she can still roll the dice and the number of points will not be affected.

<u>Requirements are as follows:</u>

0	***Activity Diagram*** • You are expected to provide an activity diagram that will explain the steps followed by the algorithm used when a user clicks on the 'Roll' button. • Your activity diagram must be done electronically (using appropriate software) or else you can draw it on a piece of paper, scan it, and add the file when submitting your assignment.
1	**Coding** All program objects including variables, constants, and methods are named following good programming standards: code layout, naming convention, comments (as specified in the text book).
2	**Forms** The application will contain 4 forms: 1. `DiceGameMainForm`: the main form in which the user will play. 2. `SummaryForm`: a form which will provide a summary of the results of the game. 3. A `ColorDialog` form: it is a standard dialog box which will allow to select the background colour of **<u>all</u>** the other forms. 4. An about form which inherits from the `AboutBox` form provided in the Visual C# templates.

3 **Form properties:**

Property settings for all forms:

Property	**Setting**
Control box	false
Start position	CenterScreen
Form border style	Fixed3D
Minimize box	false
Maximize box	false

4 **Menu of `DiceGameMainForm`**

- The menu on the main dice game form will have the following structure (including hot keys).

```
File            Edit            Help
 New game        Color           About
 Exit
```

- The shortcut keys for the menu entries must be:
  - New Game: CTRL + N
  - Exit: CTRL + E
  - Color CTRL + C
  - About: ALT + A

5 **Controls of `DiceGameMainForm`**

The main form must contain the controls displayed on the screen captures used in this document. There must be:

- 2 picture boxes: each one will be used to display a dice.
- A label and a textbox for the number of points won by a user for a given roll (this textbox will only display the number of points won thus excluding the points lost for wrong guesses. Follow examples on pages 4 and 5).
- A label and a textbox for the total number of points.
- A menu (see previous requirement).
- A button to display the summary form.
- The 'guess options' will be in a group box. Each of them can be selected using a check box. Sums are entered in a separate text box. A combo box contains the possible values (from 1 to 6) when the 'at least one dice' option is selected.

You are free to improve the appearance of the form but the mentioned picture boxes, labels, and textboxes must have the same location as the above requirements.

6 **`myColorDialogForm`**

- The menu entry Edit->Colour opens a ColorDialog dialog box in which the user will select the background colour for **all** the forms of the application (main form, summary, and about box).

7	**New Game** The menu entry File->New Game resets the number of points to ten and resets all the variables (or properties) which store the current number of games, wins …
8	**`SummaryForm`** • `SummaryForm` is a form that will contain the summary of the results that a user has obtained while playing. It opens when the user clicks on the Summary button of the main dice game form. • This form will be used as a modal form. • The form will contain two buttons: o One button entitled 'Exit' to exit the application. o One button entitled Continue Playing' to return to the main form and to continue playing (in such a case, the `SummaryForm` will be closed, but not destroyed since it will continue updating the summary of the results). • The form will display: o Current number of points. o Number of times the dice were rolled. o Total number of times the user has correctly guessed a sum. o Total number of times the user has correctly guessed a double. o Total number of times the user has correctly guessed the value of one dice. All the above information must be stored as **properties** in the `SummaryForm` class. • All the above properties are reset to zero if a user selects File->New Game. • There are no specific requirements for the controls used in this form. You may choose the controls which seem the most appropriate to display the information mentioned above.
9	**About Form** • The AboutForm will be designed from the AboutBox template provided by Visual C#. • It will contain a picture and display information about the application (e.g. creator, date, description…)
10	**Extra Feature** • You are expected to add an extra feature of your choice. You are free to add any feature which you think would improve the application. • The extra feature must be briefly explained in a multiple line comment at the top of the Program.cs file.

Important Help

- **Generating and displaying the dice**

To decide which image to display for each dice, you will use a method that returns a random number between 1 and 6. Each number will be associated with the corresponding image files:

  - 1 → Dice1.png
  - 2 → Dice2.png
  - 3 → Dice3.png
  - 4 → Dice4.png
  - 5 → Dice5.png
  - 6 → Dice6.png

To do so, use the following method in `DiceGameMainForm`:

```
private int generateRandomDice()
        {
            Random random = new Random();
            System.Threading.Thread.Sleep(350);
            return random.Next(1, 7);
        }
```

This method will randomly return either the numbers 1, 2, 3, 4, 5, or 6. Do not try to change the code of this method. Each line has a specific use.

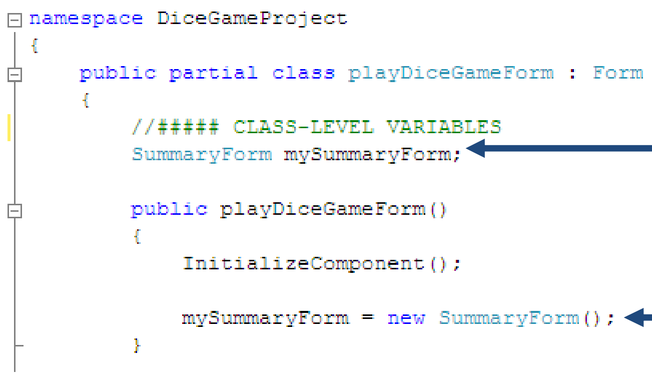
Write another method which returns the name of the file associated with a number given as a parameter (this number will contain the generated random number provided by the `generateRandomDice` method.

This method should have the following signature:

```
private string getDiceFile(int diceResultInt) {
//you are STRONGLY recommended to use a SWITCH structure in this method
in which each case will return the appropriate picture file name.
}
```

- **Use of a class-level variable for the summary form.**

It is strongly recommended to declare your summary form as a class-level variable within `DiceGameMainForm.` The same way you would use an int, decimal, or string variable, you can declare the following class-level variable:



```
namespace DiceGameProject
{
    public partial class playDiceGameForm : Form
    {
        //##### CLASS-LEVEL VARIABLES
        SummaryForm mySummaryForm;

        public playDiceGameForm()
        {
            InitializeComponent();

            mySummaryForm = new SummaryForm();
        }
```

Declaration of the class-level variable which type is `SummaryForm`.

Do not forget to add this line of code, to create your `SummaryForm`.

Once this is done, you simply need to use the `ShowDialog()` method to display the `SummaryForm` when it is needed.

- **<u>A strategy for proceeding:</u>**

Recommended process for developing your solution

1. Sketch the two forms: the main form and the summary form (no need to sketch the ColorDialog dialog box or the about form).
2. Draw the activity diagram associated with the algorithm that will be used when a user clicks on 'Roll'. Consider all possible cases (including exceptions).
3. Create the designs for the two forms in C#.
4. Set the properties of all forms and all objects on the forms.
5. Add the menu in `DiceGameMainForm`.
6. Write the code needed to get all of the forms to open and close.
7. Write the code for `DiceGameMainForm`.
    - Start with generating two separate numbers, one for each dice, and try to display the corresponding picture files (using `generateRandomDice` and `getDiceFile`).
    - Implement each 'guess option' separately. First make sure that an option works perfectly well before starting to implement the next one.
    - Once all the options are implemented (and work), you may add the code handling all the exceptional cases.
    - Write the code in `SummaryForm`. Declare the class properties one by one. You may start with the total number of times the player has rolled the dice and display it in the form. It also means that you need to update your code in `DiceGameMainForm` so that each time a user rolls the dice, the corresponding number of rolls is increased by one.

8. Write the code for the Menu->New Game menu entry.
    - Do not forget to reset the `SummaryForm` properties.

9. Write the code for the ColorDialog dialog box.
    - Make sure the selected color is applied to all your forms.

Indicative Marking Guide

	Guidelines	Total Mark
1	Activity diagram • The activity diagram uses correct UML standards (as seen in class). • The diagram is correct and does not omit any important case.	/3
2	Overall application • Program executes on opening. • Each form opens and closes according to requirements.	/2
3	`DiceGameMainForm` • Correct controls (including correct location). • Menu follows the requirements. • Images are properly displayed according to the generation of random numbers. • Points are correctly calculated. • Number of points is updated appropriately when a user is playing. • The 'New game' menu entry resets the game results correctly. • All potential exceptions are handled.	/12
4	Summary information: • Design of the page, use of appropriate controls • Modal / Modeless mode used according to requirements. • Displays correct summary information. • Exit and continue buttons work. • Summary information is correctly updated in case a user returns to the game and display the summary again.	/5
5	ColorDialog dialog box • Color is changed in the main form when a user selects a background color. • The color change applies to all forms.	/2
6	About form • Appearance is correct. • Appropriate values are displayed. • Correct template is used.	/2
7	Methods, events, and properties • Method calls are correct. • Properties are correctly defined in classes. • Property values are properly transferred from one form to the other one. • Correct events are used.	/4

8	Code comments are present and appropriate • Main comment • Code and method comments	/2
9	Naming standards correct and consistent throughout • Form and other controls • Variables • Constant	/2
10	Clean code • Correct layout (indentation) • No empty methods • Proper use of blank lines and code blocks	/2
11	Extra feature • The suggested extra feature is relevant. • The suggested extra feature provides value to the application.	/4
		TOTAL: 40 marks

20% of total course mark