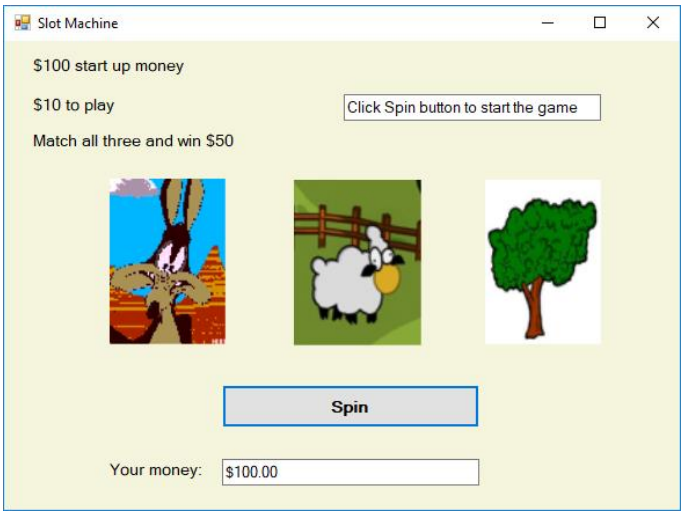
Multiple Class Instances

- Make a Spinner class.
- Create 3 Spinner objects (instances of the Spinner class).
- Use the Spinner objects, by calling the Spinner's Spin() method.

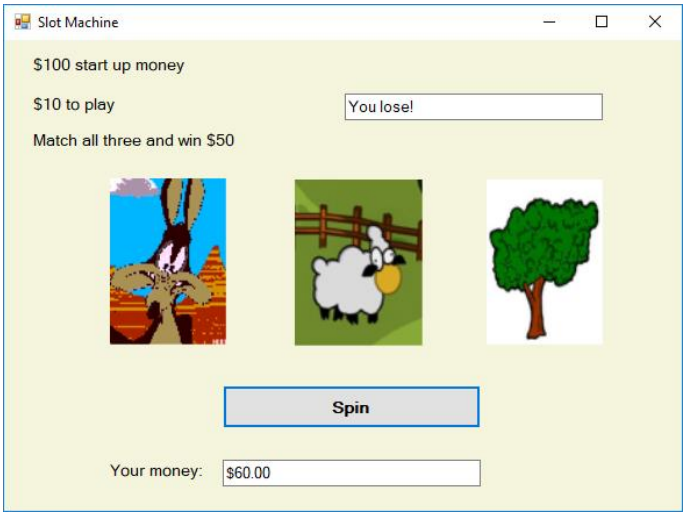
Slot Machine ******* CHECKPOINT 4 *******

Your game should provide a *Spin* button. The user is given $100 at the start of the game. It costs $10 for each click of the *Spin* button. When the user clicks this *Spin* button, a random sequence of 20 images is displayed in each of the three locations. Each image should be a tree, a coyote, or a sheep (these .bmp files are available on Teams). If all three images are the same, the player wins $50. Your game should keep track of the player's current total money. Your game should also tell the player, at each spin, if he has won or lost.

A possible screen layout is shown below:



At the start of the game



During the game

Implementation Suggestions:

1. **Set up the Form** with three PictureBoxes, four Labels, two TextBoxes and one Button. Consider colours, fonts, sizes, proportions and alignment.

2. **Make a Spinner class.**
   It will need an integer **constant** for the number of images that will be used.

   The ***fields*** should contain an array to hold the three images, an integer representing the index in the array of the image to be loaded (I called this *imageNumber*), a Random object and a reference to its PictureBox on the form.

   Its ***constructo***r is used to initialise all the field values.

   There should be a ***method*** called *Spin* which generates a random number and uses the random value to decide which of the three images (sheep, tree or coyote) will be loaded into the PictureBox.
   The *imageNumber* will be used to compare the three images and determine whether the player has won or lost.

   In the Spinner class, create a ***property*** for the *imageNumber* field. A property is a method and is used to **set** or **get** a field value **from outside the class**. This is necessary since we always define the fields as private to their own class. To **get** is to access or read the value that has been stored in the field. To **set** is to change or write to a field. Properties can also be used to check that legal values are used or to control how the data is manipulated. Properties can be Read-Write, Read-only or Write-only.

   Alternatively, from the Spinner class, RightClick on any field, select *Quick Actions and Refactoring*. Or click on the lightbulb in the left hand gutter. Choose the second option: *Encapsulate field: 'field name' (but still use the field)* and the property will be created for you. The property name will be exactly the same as the field name except it starts with a capital letter. This is a C# convention.

   You should reorganise your code so that you list in the following order: constants, fields, constructor, programmer-defined methods, events, properties.

3. **In the Form1 class, create 3 Spinner objects.**
   In the Form1 class, create three Spinner objects (I called these *spinner1*, *spinner2* and *spinner3*). Alternatively create an array of Spinner objects.

7. **Use the Spinner objects.**
   Add fields to store the user's current total money (I called this *winnings*), and a Random object.
   Create a *Click* handler for the *Spin* button, where $10 is deducted. In a *for* loop of 20 iterations, each spinner spins in turn and the final outcomes are compared. If there is a match, the user should be credited $50. Remember to use constants.

8. Modify your game so that if the player runs out of money (i.e. his total winnings go to 0), the game ends, and the feedback box reports "You're out of money". (To test this, you may want to change your win/loss amounts so that the user runs out of money quickly.)

9. Separate the Slot Machine functionality from the *button1_Click* event. Create methods to hold all the Slot Machine game function. Then in the *button1_Click* event handler, call these methods.