**Part 1**

Software objects are modeled after real-world objects in that they have state and behavior as the real ones. A software object maintains its state in variables and implements its behavior with methods. You can represent real-world objects using software objects. However, you can also use software objects to model some abstract concepts.

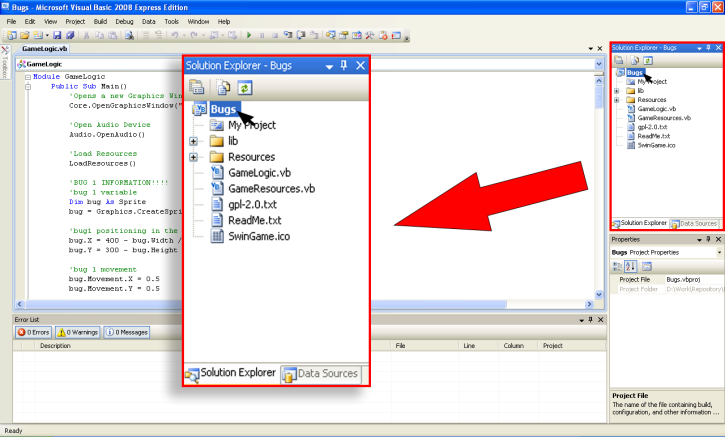
Object knows and can do certain things. To create a new object we need to create a class, which will be a template for all objects made from this class. This class can contain fields (things that an object knows) and methods (things that an object can do).

In our case we need to create a Bug class which will contain all behavior and all data associated with our object. To do so, we will need a constructor, which will set up the data that we need for our object such as creating a sprite, declare a position of the sprite, declare sprite movement. A constructor is a public sub, which has to have a name “New” and it will be called firstly when an object from this class is created.

Also, in order to create an object and see any result on the screen, we will need to have a Draw() and Update() methods.

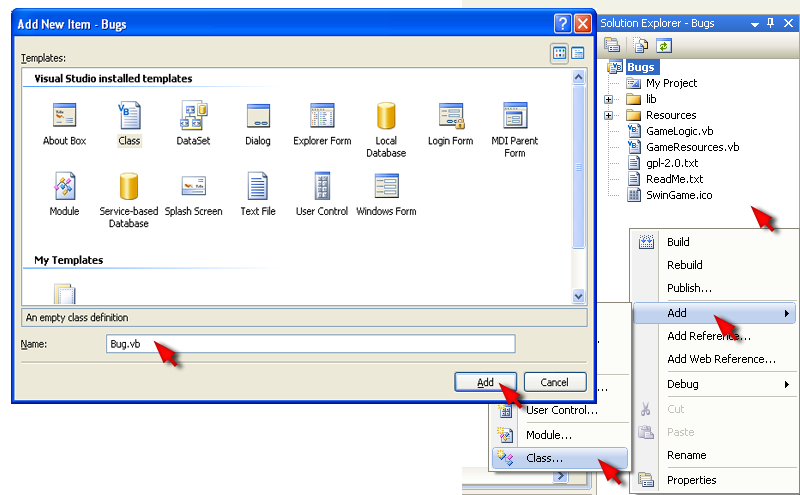
Firstly, we need to add a class to your game. To do so, follow the steps below:

1. In solution explorer, choose “Bugs” at the top of the list – this means thet you are choosing the whole game (Figure 1):



Figure

1. Right click inside the solution explorer, choose “Add” => “Class”, and type “Bug.vb” as class name and click “Add” button (Figure 2):



Figure

We will need to create three fields inside Bug class. First one will represent an alive sprite – the sprite that you see at the start of the game (Private AliveSprite As Sprite), second one will represent dead sprite (Private DeadSprite As Sprite) this is the animated sprite which will be played when the sprite was clicked, and the third one will be Alive fiels which will be Boolean (Boolean is simply true or false value of the variable - Private Alive As Boolean). Notice – all of them are private, which means you can access them only inside this class. In order to make them accessible from the outside we need to create a property. A property is a construction in programming language, that allows you to read or write a private field from the outside the class. To create a property for Alive variable you need to use the following construction:

|  |
| --- |
| Public Property IsAlive() As Boolean  Get  Return Alive 'allows to read the value  End Get  Set(ByVal value As Boolean)  Alive = value 'allows to assign a value  End Set  End Property |

*Exercise 1: Creating fields and a property*

cha 2 - worksheet.pngMake the following changes in your program and write your solutions to the worksheet:

1. In you Bug class, create AliveSprite, DeadSprite and Alive fields and a property for Alive field.

When we are creating a new object from this class, this object must have Alive = True, because we need to show alive Sprite at the beginning, it must know position of the AliveSprite and the movement. An object must also create a DeadSprite, in order to use it later, and for playing DeadSprite animation only ones, we need to put DeadSprite.EndingAction = SpriteEndingAction.Stop. All of this will be enclosed within a constructor.

To create a constructor you need to type Public Sub New() into your Bug class and press Enter. Assign Alive variable to true. Type AliveSprite = Graphics.CreateSprite(GameImage("sprite")) to create alive sprite. Now we need to declare the position of the sprite. It will be more interesting if we would have each time random starting position for our bug. To do so type

AliveSprite.X = Rnd() \* (800 - AliveSprite.Width)

AliveSprite.Y = Rnd() \* (600 - AliveSprite.Height).

Rnd() function generates random number between 0 and 1 (i.e. 0.002). We need to consider width and height of our sprite in order to find right position ( so the sprite will be always inside the game window).

Also it will be much more interesting in our bug will move in different directions