

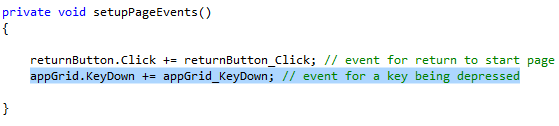
Systems Development: Object Oriented Programming

(H171 35)

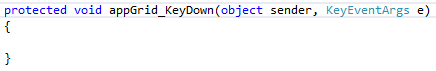
WPF Application

Step 5 – responding to key presses

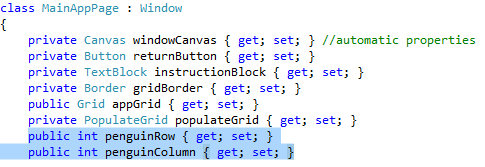
* The next stage is to move the penguin around the screen using the arrow keys
* This app is going to respond to key presses made by the user. To capture a keypress we will be using the “KeyDown” event. This occurs when a key on the keyboard is pressed. We will then use “KeyEventArgs” class which provides data for the [UIElement.KeyUp](https://msdn.microsoft.com/en-us/library/system.windows.uielement.keyup%28v=vs.110%29.aspx) and [UIElement.KeyDown](https://msdn.microsoft.com/en-us/library/system.windows.uielement.keydown%28v=vs.110%29.aspx) events to determine which key was pressed and then perform the appropriate action
  + First we need to add the KeyDown event to our MainAppPage, we already have a method called setupPageEvents(), so let’s add it in here



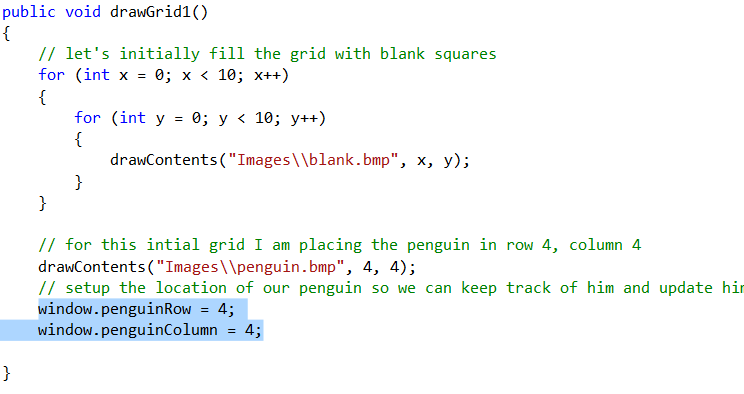
* Now we need to write the event handler in MainAppPage.cs. We will create the stub for this method and come back and fill in the implementation later



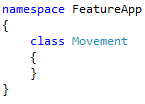
* We need to track is the location of our penguin so we can successfully move it around. We will store the penguins row and column values in the MainAppPage



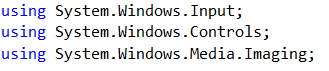
* These values will then be initially set when the grid is drawn in the PopulateGrid class



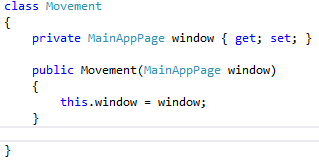
* This event handler is going to move the penguin image around the screen, in the direction of any arrow keys the user presses. Implementing movement involves several instructions so we are going to create a separate Movement class. (You may eventually want to expand this to add conditions for the movement)



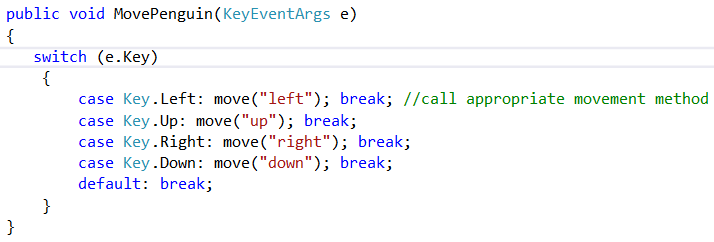
* Make sure you add these libraries



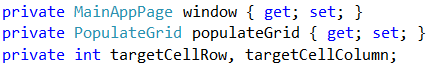
* This movement class needs to know about the window in which it is applying the movement to so we have a MainAppPage class member which is initialised by the constructor with the window being passed as a parameter



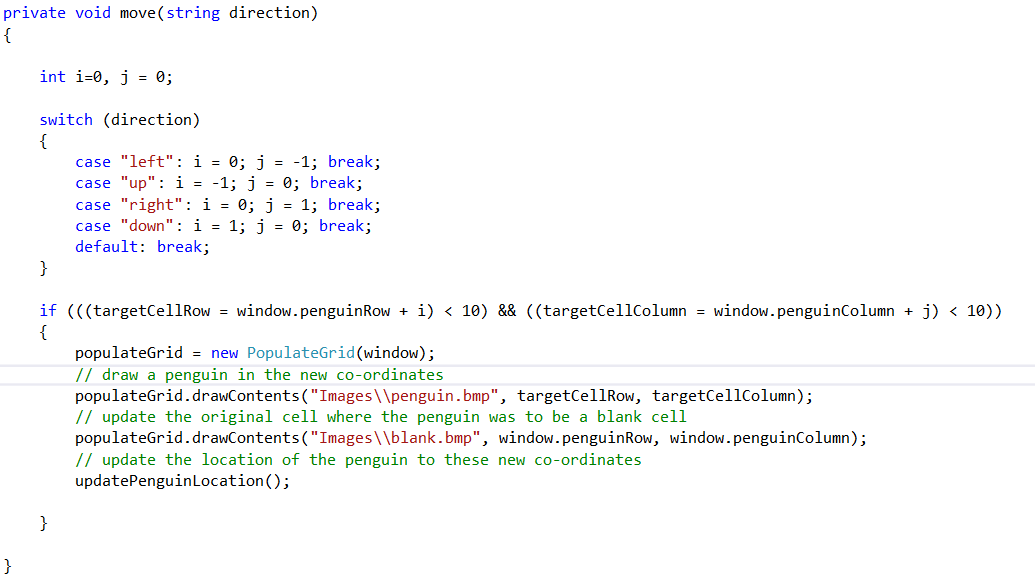
* Now we are going to write a method called MovePenguin which analyses the keypress and decides which direction the penguin is going to move:



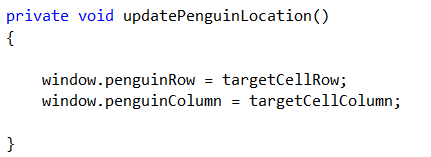
* We need to take note of the target cell we want the penguin to move to, therefore we will add 2 new member variables to the Movement class. We will also be populating the grid with a new location for the penguin, therefore we will create a PopulateGrid member variable:



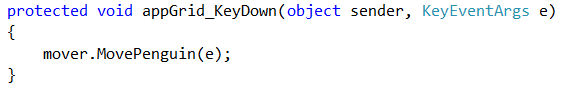
* We now need to write the move method used above, which gets passed into it the direction of the move
  + We need to update the row and column positions based on the direction of movement, where “i” is for row and “j” is for column
  + If the new row and column positions are within the boundary of the grid then we can go ahead
    - Pass our window we want updated to the populateGrid instance
    - Draw the penguin in it’s new location on the grid
    - Fill the cell where the penguin originally was with a blank
    - Update the location of the penguin to be this new cell



* We need to write the updatePenguinLocation method as used above. This simply updates the row and column of the penguins location to the new details:



* Our final step is to revisit the event handler we created in MainAppPage.cs which handles the “KeyDown” event and call the method to move the penguin when a key is pressed:



NOW RUN AND TEST THE APPLICATION – You should be able to move the penguin around the screen using the arrow keys and each cell behind I will be filled with a blank.