Class Assignment – Hare and Tortoise Part E:

1 Introduction

In this part of the assignment you will complete the functionality of the Square subclasses to achieve a working prototype of the board game as described in Part A.

2 Additional functionality for The EffectOnPlayer methods in the Square subclasses

In **Part D, Section 3** each of the subclasses of **Square** had a method **EffectOnPlayer** which changed the amount of money a player had should they finish on that type of square.

In this section, each of these methods will be enlarged to change the final location of a player who lands on that type of square.

In the **Win_Square** class the **EffectOnPlayer** method will add $15 to the player's money and the player will roll the dice again to move to a new square on the board. This movement may cause the player to land on another "special" square which will affect the player accordingly. It is possible, though highly unlikely that a player after landing on a **Win_Square** may as a result of subsequent roll land on another **Win_Square** and therefore be required to roll the dice again.

To implement the action of **"rolling the dice again",** it is suggested that you add a public parameterless method to **Player** class which will then call the **"roll the dice"** method that was implemented in **Part C, Section 3.1**

In the **Lose_Square** class the **EffectOnPlayer** method will deduct $25 from the player's money and the player will move back three (3) squares. So if the player had landed on square 10 they will move to square 7.

In the **Chance_Square** class the **EffectOnPlayer** method will either add or deduct $50 from the player's money with a 50% chance of one of the options occurring. If the player receives an additional $50 they will move forward by five (5) squares. If the player is deducted $50 they will back by five (5) squares.

3 Simple Animation

This section should only be done once you have the rest of this assignment working correctly. If there are issues with your game playing 100% correctly up to this point do not waste your time attempting this animation, it will not be marked.

You should maintain a separate version of your assignment which plays the game up to this stage in the event that you do not complete this additional functionality. So that you have a working program which can be submitted rather than an incomplete program which no longer works correctly.

The animation involves moving each player *one square at a time*. For example, if player One is on the Start square and rolls 8, then player One is displayed moving to square 1, then to square 2, and so , until they reach square 8, rather than moving from the Start Square to square 8 in one movement. Should the player land on a "special" square the resulting movement will also be from square to square.

It is recommended that you do not use the method which you wrote in Part C Section 1.2 which moved all the players at the same time. Nonetheless that method will give you some ideas how to implement this animation.

Obviously you will need to *redisplay the GUI board* each time you move the player from one square to the next. When you run your program you will probably see that the players' tokens appear to move altogether in one go. The command to *redisplay the GUI board* doesn't actually get an opportunity to complete it activity while the rest of your code is executing. (The redisplaying of the GUI is a separate thread to the main execution thread of your program)

To overcome this small issue, insert the statement `Application.DoEvents();` immediately after the command to *redisplay the GUI board*. This method call gives all of the GUI controls the opportunity to redisplay themselves while other code is executing.

Depending upon the speed of your computer, you may find that the tokens still move very rapidly from square to square. You can slow them down by adding the following statement after the call to `Application.DoEvents();`

```
Thread.Sleep(100); //Sleep for 100 milliseconds.
```

This will cause your computer to pause for at least 100 milliseconds between moves. You can make this time-delay bigger or smaller, but do not make it too big or your game will be very slow to play.