Module Module1

Sub Main()

Dim NewBingoGame As New BingoGame 'instantiation

NewBingoGame.Menu()

Console.ReadLine()

End Sub

End Module

Public Class BingoGame

Dim numbers As New NumberMachine 'composite aggregation

Public Sub New()

Console.WriteLine("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$")

Console.WriteLine("\* B-I-N-G-O S-I-M \*")

Console.WriteLine("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$")

End Sub

Public Sub PlayGame()

Dim playerCard As New BingoCard 'composite aggregation

Dim won As Boolean

playerCard.Displaycard()

Console.WriteLine("Eyes Down... ")

Do

Caller()

Console.WriteLine("Did you win?")

Console.WriteLine("Enter 1 for yes and 0 for no?")

won = Console.ReadLine()

Console.Clear() ' removes all text from console

playerCard.Displaycard()

Loop Until won 'loops until won = 1 (yes)

Console.WriteLine("You have matched " & playerCard.GameOver(numbers.getNumbers, numbers.getBack))

If playerCard.GameOver(numbers.getNumbers, numbers.getBack) = 15 Then

Console.WriteLine("Yes you have won!")

Else

Console.WriteLine("Sorry you stopped too early")

Console.WriteLine("You only matched " & playerCard.GameOver(numbers.getNumbers, numbers.getBack))

Console.WriteLine("GAME OVER")

End If

End Sub

Public Sub PlayBonusGame()

Dim playerCard As New BingoBonusCard

Dim won As Boolean

playerCard.Displaycard()

Console.WriteLine("Eyes Down... ")

Do

Console.WriteLine("\*\*BINGO BONUS\*\*")

Caller()

Console.WriteLine("Did you win?")

Console.WriteLine("Enter 1 for yes and 0 for no?")

won = Console.ReadLine()

Console.Clear()

playerCard.Displaycard()

Loop Until won

Select Case playerCard.GameOver(numbers.getNumbers, numbers.getBack)

Case 1

Console.WriteLine("Well Done Full House")

Case 2

Console.WriteLine("Well Done Two Lines")

Case 0

Console.WriteLine("Sorry you stopped too early, GAME OVER")

End Select

End Sub

Private Function Caller() As Integer

Caller = numbers.nextBall

Console.WriteLine("and the next ball is.....")

If Caller = 11 Then

Console.WriteLine("legs 11")

Else

Console.WriteLine(Caller)

End If

End Function

Public Function Menu() As Boolean

Dim choice As Integer

Do

MenuOptions()

choice = Console.ReadLine()

Select Case choice

Case 0

Console.WriteLine("Goodbye")

Case 1

PlayGame()

Case 2

PlayBonusGame()

Case 3

numbers.PracticeGame()

Console.Clear()

Console.WriteLine("Practice Game Initiated")

Case Else

Console.WriteLine("Not an Option")

End Select

Loop Until choice = 1 Or choice = 2 Or choice = 0

Return True

End Function

Public Sub MenuOptions()

Console.WriteLine("---------------------------------------")

Console.WriteLine("Choose 1 play Bingo")

Console.WriteLine("Choose 2 play Bingo Bonus")

Console.WriteLine("Choose 3 play practice game")

Console.WriteLine("Choose 0 to exit")

End Sub

End Class

Public Class BingoCard

Protected numbers(2, 8) As Integer ' numbers on card 'protect means only editable within that class

Public Sub New()

numbers = AssignNumbers()

End Sub

Private Function AssignNumbers() As Integer(,) 'assigns the numbers in the bingo card

'Randomize()

Dim row1(4) As Integer 'there are 5 numbers on each row and 5 spaces on each row

Dim row2(4) As Integer ' row() gets 5 numbers which gives the position on the row where the actual number to be called will be placed

Dim row3(4) As Integer

Dim cardnumbers(2, 8) As Integer 'refers to the whole card (3 rows, 9 columns)

row1 = AssignRowPlaces() ' number assigned location

row2 = AssignRowPlaces() 'number assigned location

row3 = AssignRowPlaces() 'number assigned location

For x = 0 To 4

Console.WriteLine(row1(x))

Next

For x = 0 To 4 'first row

cardnumbers(0, row1(x)) = repo.NewRandom(1, 8) + (10 \* row1(x)) 'the lower bound is 1, the upperbound is 8 - and so a random number is generated between 1 and 8. then 10 \* position of number is added to this number

Next

For x = 0 To 4 'second row

Dim base As Integer

base = (cardnumbers(0, row2(x)) Mod 10) + 1 'base is compared to the value in same index in array on row 1, mod 10 and then 1 is added

base = repo.NewRandom(base, 9) 'number generated between base calculated above and 9

cardnumbers(1, row2(x)) = base + (10 \* row2(x)) 'number in row 1 = base calculated above (between base and 9) + (10 \* position in row 2)

Next

For x = 0 To 4 'third row

Dim base As Integer

If cardnumbers(1, row3(x)) = 0 Then 'if the value in row 2 in same position of array in row 3 is 0 then

base = (cardnumbers(0, row3(x)) Mod 10) + 1 'it will use the number in row 1 of same position, mod 10 and then add 1

Else

base = (cardnumbers(1, row3(x)) Mod 10) + 1 'else it will use the value in row 3 in the same position, mod 10 and then add 1

End If

base = repo.NewRandom(base, 10) 'base is random number between base and 10

cardnumbers(2, row3(x)) = base + (10 \* row3(x)) 'number in row 3 = base calculated above + (10 \* position in row)

Next

Return cardnumbers

End Function

Public Sub Displaycard() 'prints the card

For x = 0 To 2

For y = 0 To 8

Console.Write(numbers(x, y) & ",")

Next y

Console.WriteLine()

Next x

End Sub

Private Function AssignRowPlaces() As Integer()

Dim numberCount As Integer 'automataily set to 0

Dim row(4) As Integer

Dim match As Boolean 'variable to check if there are duplicates

For x = 0 To 4

row(x) = -1 'this sets 5 numbers in each row to -1

Next

Do While numberCount <= 4

match = False

row(numberCount) = repo.NewRandom(0, 8) 'generates a random number beween 0 and 89 to be the random number on the card

numberCount += 1

For x = 0 To 4 'for loop and if statement below it to check if there are duplicates, then number count goes down 1 and the process repeats until it has 5 random numbers

If row(numberCount - 1) = row(x) And numberCount - 1 <> x Then

match = True

End If

Next

If match Then

numberCount -= 1

End If

Loop

row = rearrage(row, 4)

Return row

End Function

Private Function rearrage(ByVal dataSet As Integer(), ByVal size As Integer) As Integer()

' makes order on row of bingo card from smallest to largest

Dim i, j As Integer

For i = 0 To size - 1

For j = 0 To size - 1

If (dataSet(j) > dataSet(j + 1)) Then

Dim temp As Integer = dataSet(j)

dataSet(j) = dataSet(j + 1)

dataSet(j + 1) = temp

End If

Next

Next

Return dataSet

End Function

Public Overridable Function GameOver(ByVal calledNumbers As Integer(), ByVal tail As Integer) As Integer 'checks to see how mow many numbers were actually called

Dim matched As Integer

For x = 0 To 2

For y = 0 To 8

If numbers(x, y) <> 0 Then

For z = 0 To tail

If numbers(x, y) = calledNumbers(z) Then

matched += 1

End If

Next

End If

Next

Next

Return matched

End Function

End Class

Public Class BingoBonusCard

Inherits BingoCard 'inheritance

Public Overrides Function GameOver(ByVal calledNumbers As Integer(), ByVal tail As Integer) As Integer 'polymorphism 'tail pointer is the end pointer of the number card

Dim matched As Integer

matched = FullHouse(calledNumbers, tail)

If matched = 15 Then

Return 1 ' if all numbers matched then returns 1

End If

Console.WriteLine("You only matched " & matched)

Return 0 ' if no numbers matched returns 0

End Function

Private Function FullHouse(ByVal calledNumbers As Integer(), ByVal tail As Integer) As Integer

Dim matched As Integer

For x = 0 To 2 'goes through each row

For y = 0 To 8 'goes through each column

If numbers(x, y) <> 0 Then

For z = 0 To tail ' 0 to amount of umbers called

If numbers(x, y) = calledNumbers(z) Then

matched += 1 ' checks if numbers matched with calledNUmbers

End If

Next

End If

Next

Next

Return matched

End Function

End Class

Class NumberMachine

Private numberOrder(74) As Integer ' bingo balls

Private current As Integer ' current number of bingo balls shown

Private back As Integer = 74 ' number of bingo balls

Public Sub New()

Dim temp, num1, num2 As Integer

For x = 1 To 75

numberOrder(x - 1) = x ' sets numberOrder from 1 to 75

Next x

For x = 1 To 1000

num1 = repo.NewRandom(0, 74) ' random number assigned to num1

num2 = repo.NewRandom(0, 74) ' random number assigned to num2

' switches numberOrder(num1) with numberOrder(num2)

temp = numberOrder(num1) ' integer in numberOrder with position of num1 is assigned to temp

numberOrder(num1) = numberOrder(num2) ' integer in numberOrder with position of num2 is assigned to numberOrder(num1)

numberOrder(num2) = temp ' integer in temp is assigned to numberOrder(num2)

Next

End Sub

Public Sub PracticeGame() ' set order of bingo balls

numberOrder(0) = 4

numberOrder(1) = 11

numberOrder(2) = 5

numberOrder(3) = 57

numberOrder(4) = 65

numberOrder(5) = 33

numberOrder(6) = 48

numberOrder(7) = 58

numberOrder(8) = 68

numberOrder(9) = 78

numberOrder(10) = 47

numberOrder(11) = 18

numberOrder(12) = 50

numberOrder(13) = 59

numberOrder(14) = 80

numberOrder(15) = 81

End Sub

Public Function nextBall() As Integer

If current < back Then ' if current is more than the original number it will stop showing

current += 1 ' number of bingo balls shown

Return numberOrder(current - 1) ' shows next bingo ball

Else

Return -1 ' returns -1 if ran out of bingo balls

End If

End Function

Public Function getNumbers() As Integer()

Return numberOrder

End Function

Public Function getBack() As Integer

Return current

End Function

End Class

Public Class repo ' class to assign random values

Public Shared Function NewRandom(ByVal lowerbound As Integer, ByVal upperbound As Integer) ' upperbound = the highest value you want, lowerbound = lowest value you want

Return CInt(Math.Floor((upperbound - lowerbound + 1) \* Rnd())) + lowerbound ' does the randomization

End Function

End Class