eLab Book

Fundamentals of Programming

Assignment:

WE21, Codebreaker assignment,

Digital Skills Academy

Author: **Gabriel Garus**

Student ID: D09122844

e-mail: gabriel.garus@webelevate.ie

Date : 14 July 2013

Version: *02*
Status: *Draft*

Table of Contents

1. Overview

Based on the Java programming language and the javabook class library, write a Codebreaker game in Java.
The game will have a purely text based interface, no graphics, as depicted in the detailed brief section.

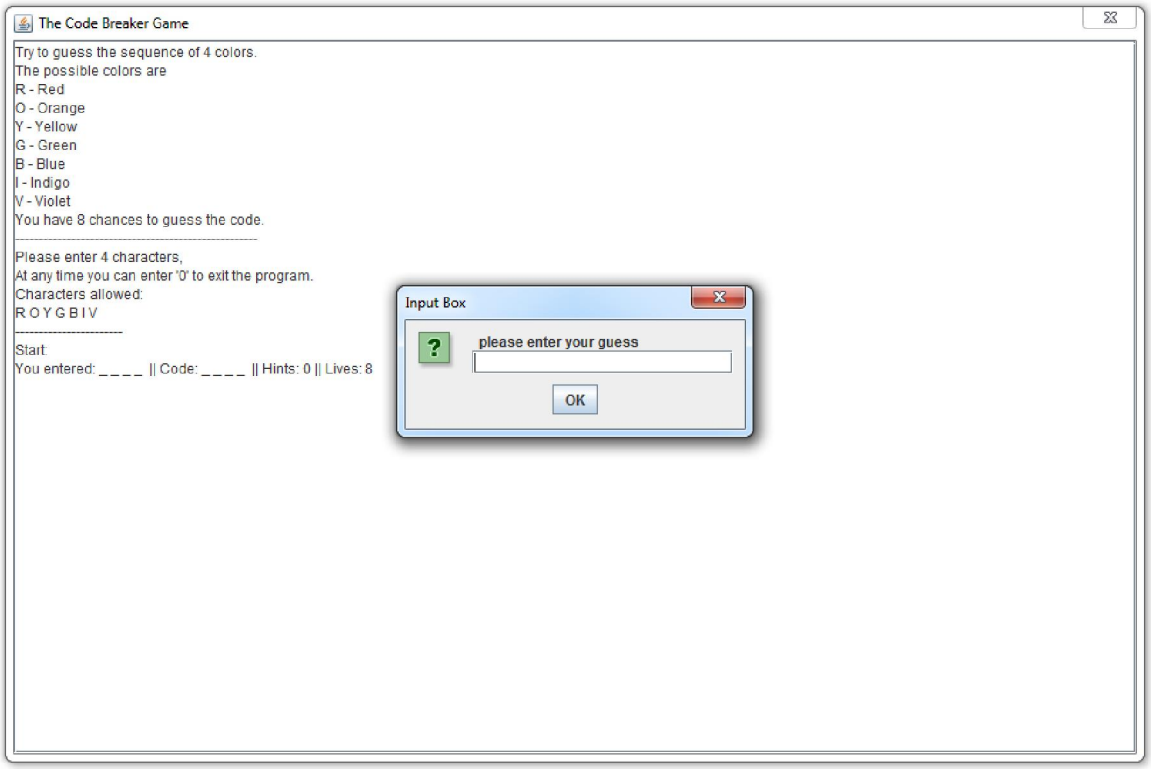
The game starts by choosing the code patch which is a sequence of four colours from the following available colours:

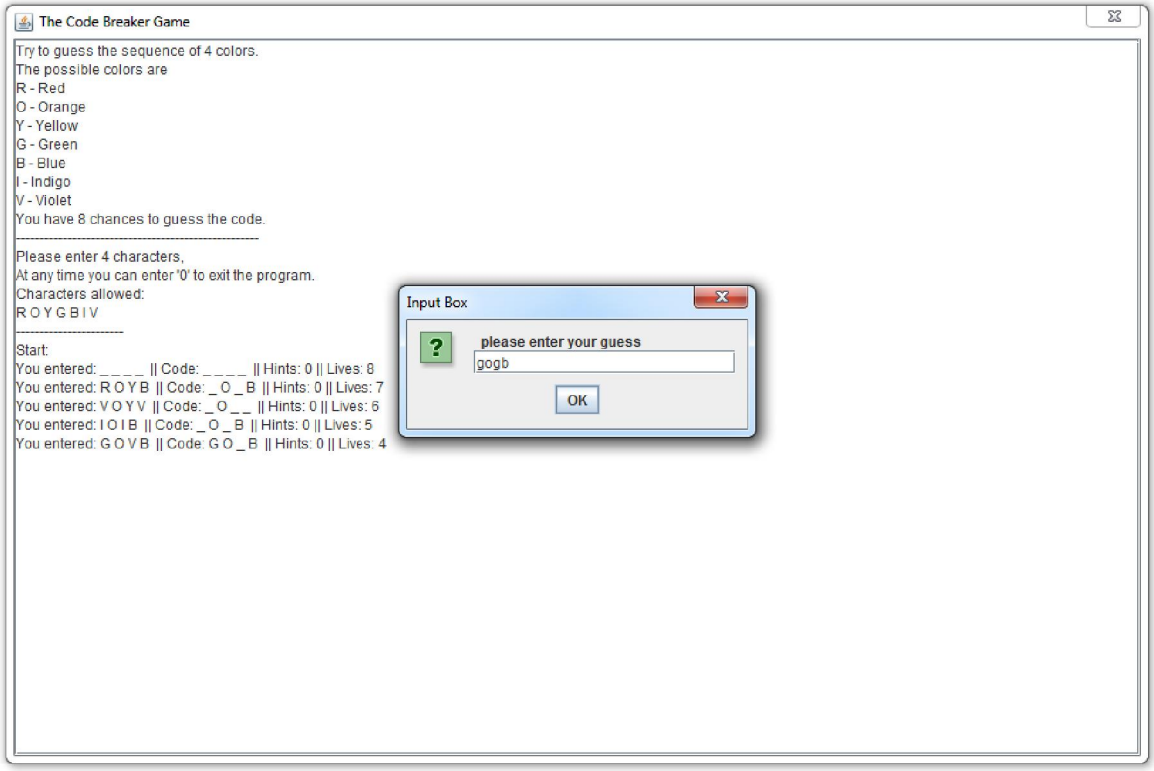
- red(R),
- orange (O),
- yellow (Y),
- green (G),
- blue (B),
- Indigo (I),
- violet (V).

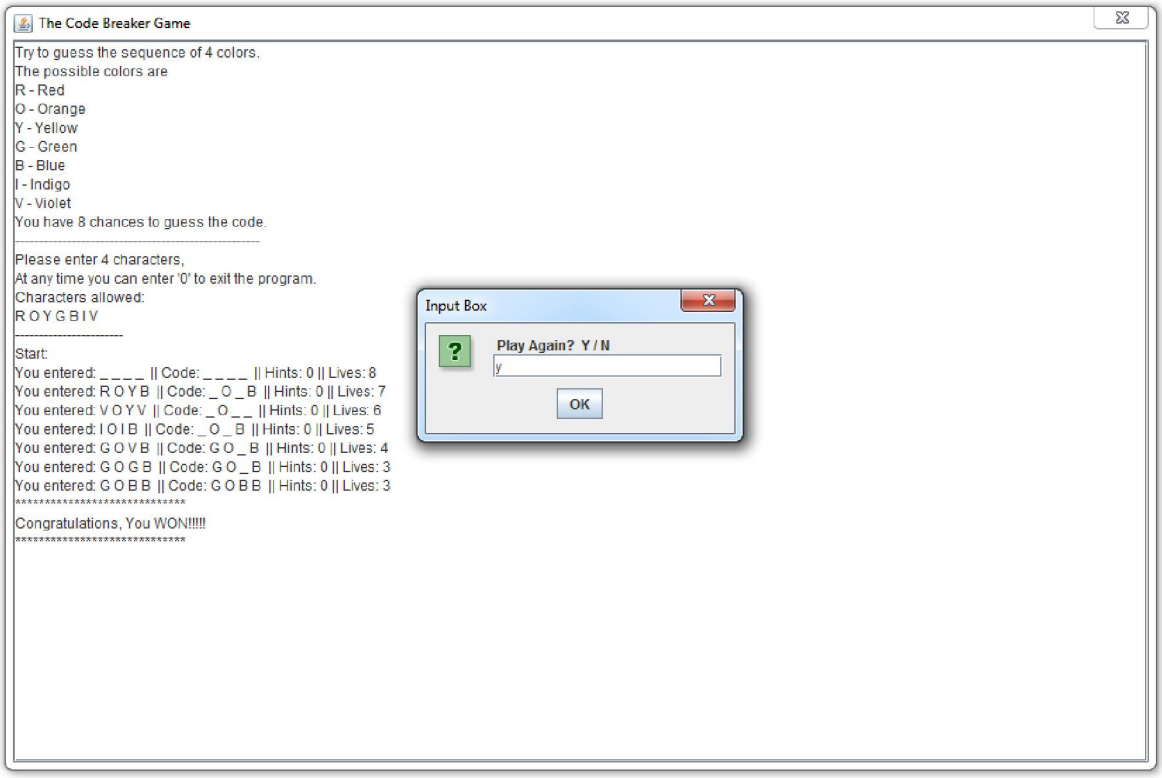
The code patch is not displayed to the user. Only 4 lines are displayed (_ _ _ _) and lives = 8.
The user enters a string of four characters which is their guess at the sequence of 4 letters choose by the computer.
The user input is compared to the code patch and the following feedback is provided:

- If the two code patches match, the game says YOU WIN, do you want to play again (Y/N)?
- If one or more colours between the two colour patches match, display the positions where the colours match.
- If the colour is correct but the position is wrong for one or more colours give the user a clue as to how many colours there are in the in the users patch that are not in the correct position.
- If the same colour is used twice in the users patch and twice in the computers patch but the positions are wrong the clue will have a value of 2.
- if the colour patch has not been guessed then a life is lost and the user is asked : Enter a sequence a 4 character sequence from ROYGBIV or 0 to exit:
- If the number of lives is zero following this guess and the user has not won, then display, the code patch and. YOU LOOSE, do you want to play again (Y/N)?
- If instead of entering in a code sequence the user enters 0, or there are 0 in the code patch entered, exit the game (boss kill switch)
- If the same sequence is entered twice or more, inform the user that duplicate patches are not allowed and ask then to re enter a new code patch. No life is lose for a duplicate entry.

2. Screens

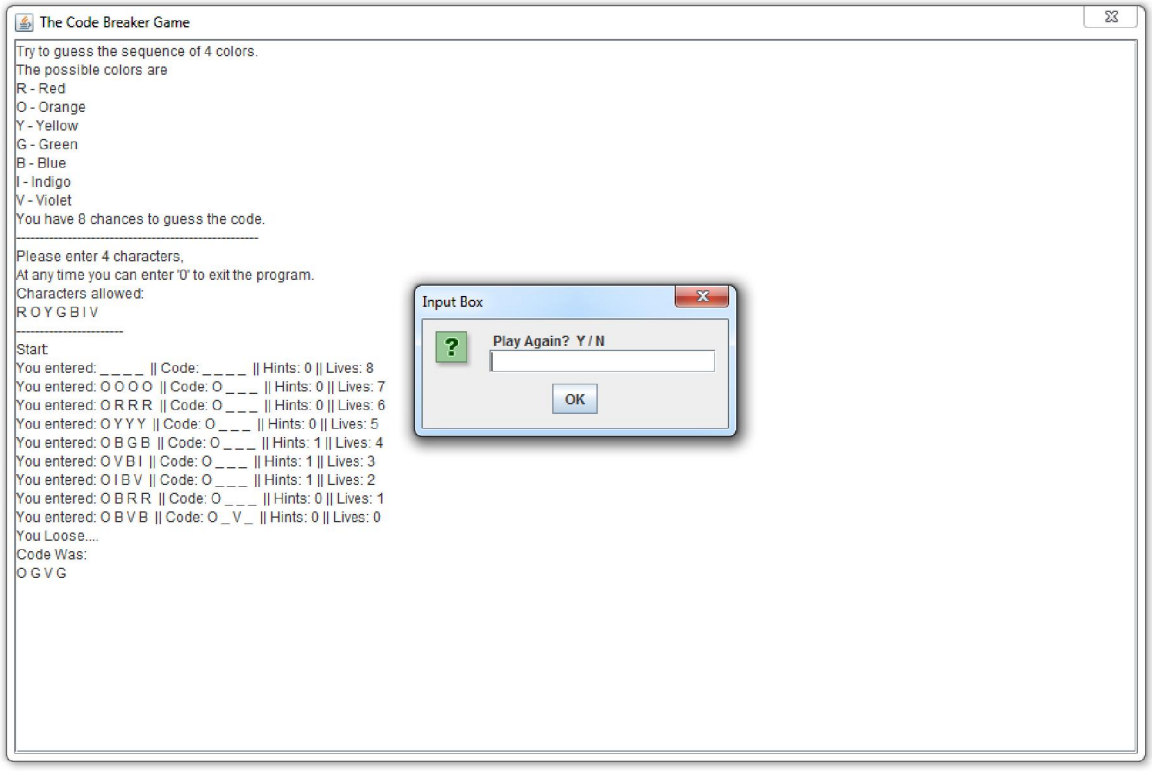

The Code Breaker Game
Try to guess the sequence of 4 colors.
The possible colors are
R - Red
O - Orange
Y - Yellow
G - Green
B - Blue
I - Indigo
V - Violet
You have 8 chances to guess the code.
Please enter 4 characters,
At any time you can enter '0' to exit the program.
Characters allowed:
R O Y G B I V
Start:
You entered: _ _ _ _ || Code: _ _ _ _ || Hints: 0 || Lives: 8
Input Box
please enter your guess
OK


The Code Breaker Game
Try to guess the sequence of 4 colors.
The possible colors are
R - Red
O - Orange
Y - Yellow
G - Green
B - Blue
I - Indigo
V - Violet
You have 8 chances to guess the code.
Please enter 4 characters,
At any time you can enter '0' to exit the program.
Characters allowed:
R O Y G B I V
Start:
You entered: _ _ _ _ || Code: _ _ _ _ || Hints: 0 || Lives: 8
You entered: R O Y B || Code: _ O _ B || Hints: 0 || Lives: 7
You entered: V O Y V || Code: _ O _ _ || Hints: 0 || Lives: 6
You entered: I O I B || Code: _ O _ B || Hints: 0 || Lives: 5
You entered: G O V B || Code: G O _ B || Hints: 0 || Lives: 4
Input Box
please enter your guess
gogb
OK


The Code Breaker Game
Try to guess the sequence of 4 colors.
The possible colors are
R - Red
O - Orange
Y - Yellow
G - Green
B - Blue
I - Indigo
V - Violet
You have 8 chances to guess the code.

Please enter 4 characters,
At any time you can enter '0' to exit the program.
Characters allowed:
R O Y G B I V

Start:
You entered: _ _ _ _ || Code: _ _ _ _ || Hints: 0 || Lives: 8
You entered: R O Y B || Code: _ O _ B || Hints: 0 || Lives: 7
You entered: V O Y V || Code: _ O _ _ || Hints: 0 || Lives: 6
You entered: I O I B || Code: _ O _ B || Hints: 0 || Lives: 5
You entered: G O V B || Code: G O _ B || Hints: 0 || Lives: 4
You entered: G O G B || Code: G O _ B || Hints: 0 || Lives: 3
You entered: G O B B || Code: G O B B || Hints: 0 || Lives: 3
****
Congratulations, You WON!!!!!
****
Input Box
Play Again? Y / N
y
OK


The Code Breaker Game
Try to guess the sequence of 4 colors.
The possible colors are
R - Red
O - Orange
Y - Yellow
G - Green
B - Blue
I - Indigo
V - Violet
You have 8 chances to guess the code.

Please enter 4 characters,
At any time you can enter '0' to exit the program.
Characters allowed:
R O Y G B I V

Start:
You entered: _ _ _ _ || Code: _ _ _ _ || Hints: 0 || Lives: 8
You entered: O O O O || Code: O _ _ _ || Hints: 0 || Lives: 7
You entered: O R R R || Code: O _ _ _ || Hints: 0 || Lives: 6
You entered: O Y Y Y || Code: O _ _ _ || Hints: 0 || Lives: 5
You entered: O B G B || Code: O _ _ _ || Hints: 1 || Lives: 4
You entered: O V B I || Code: O _ _ _ || Hints: 1 || Lives: 3
You entered: O I B V || Code: O _ _ _ || Hints: 1 || Lives: 2
You entered: O B R R || Code: O _ _ _ || Hints: 0 || Lives: 1
You entered: O B V B || Code: O _ V _ || Hints: 0 || Lives: 0
You Loose....
Code Was:
O G V G
Input Box
Play Again? Y / N
OK

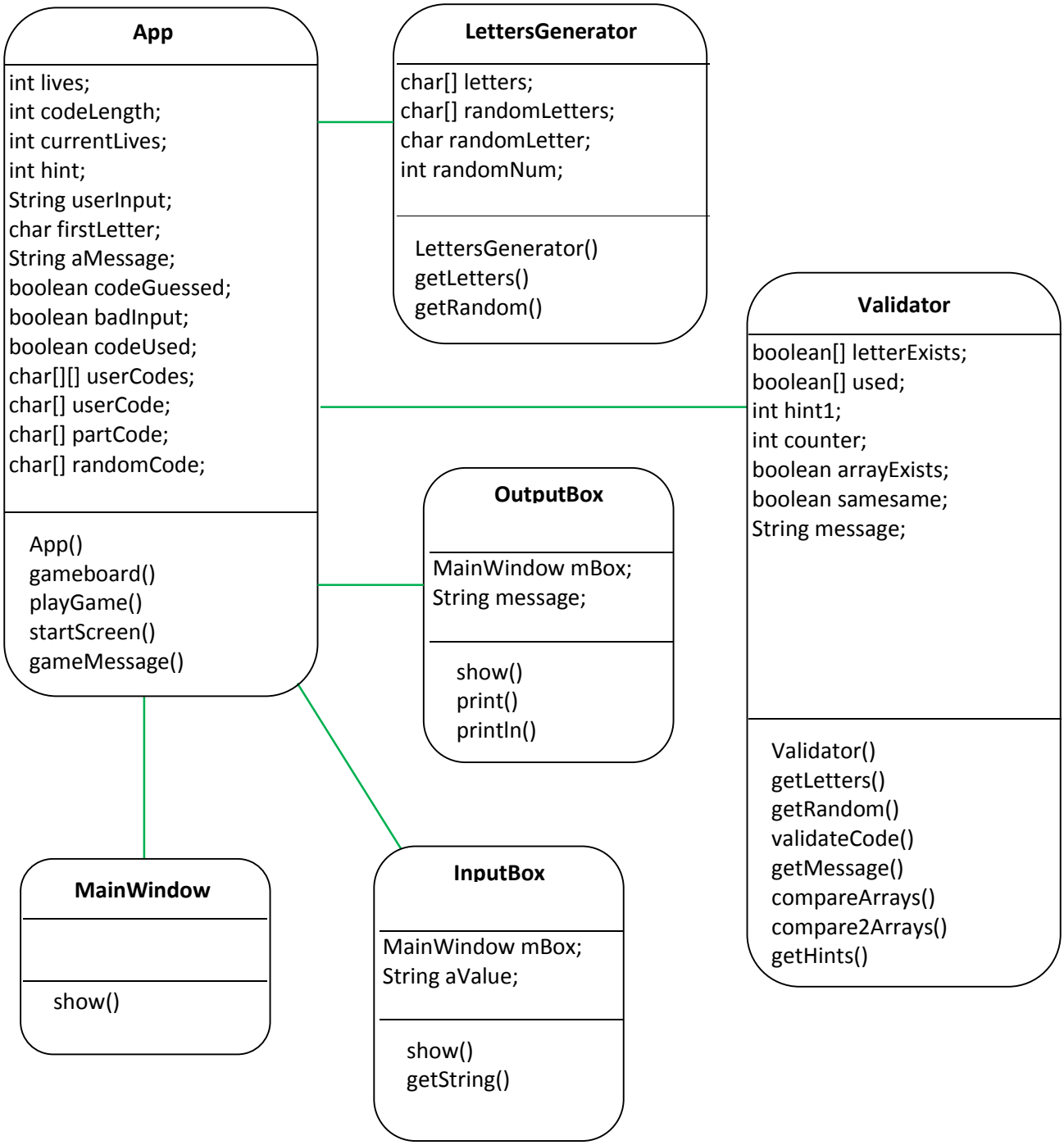
3. Documentation

3.1. How to use the program documentation

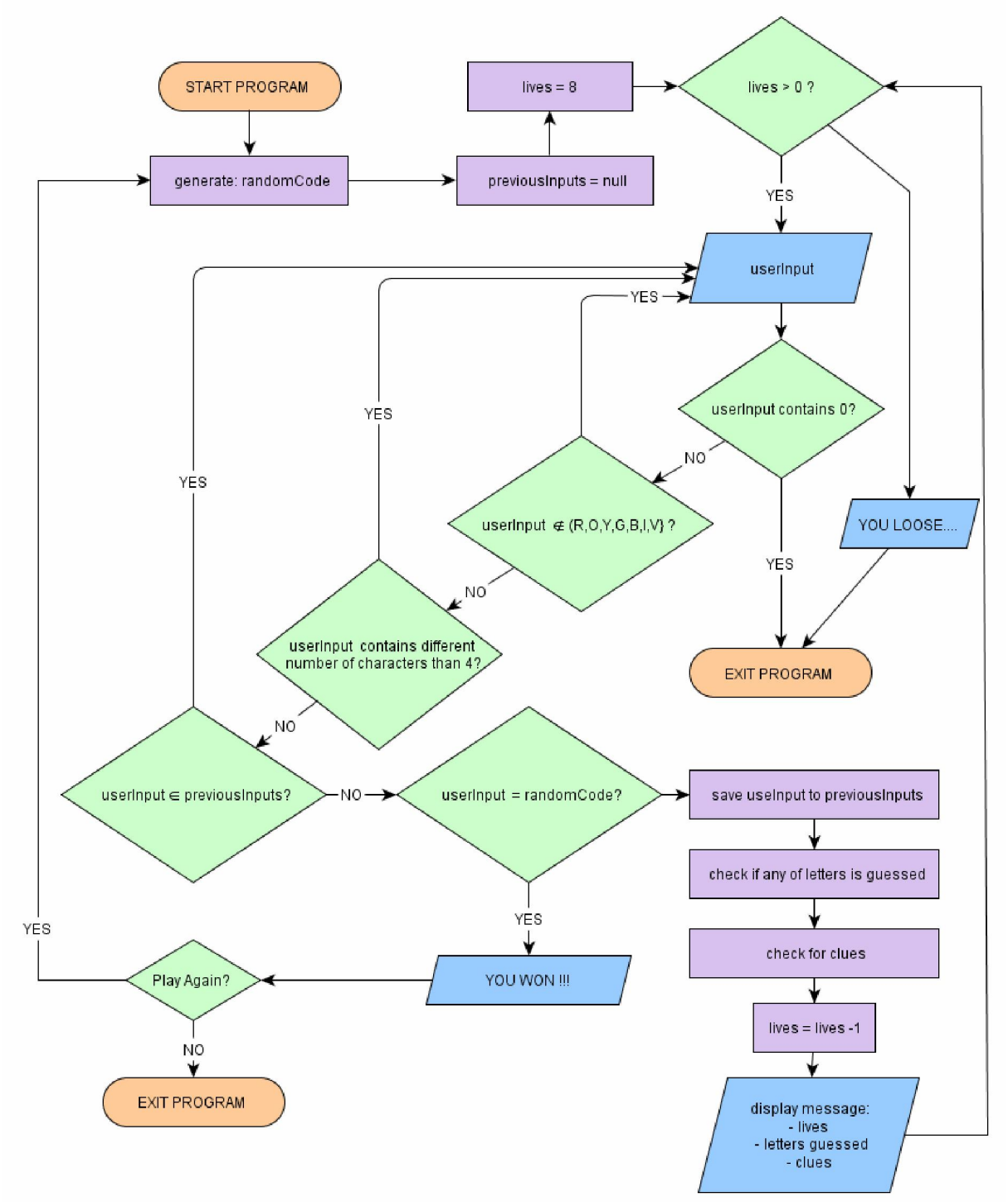
This documentation describes design of the game. This might be helpful for future upgrading/redesigning the game.

3.2. Object and class diagrams showing the implementation

Class Diagram of CodeBreaker game:



Flowchart representing logic behind game design.


START PROGRAM
generate: randomCode
previousInputs = null
lives = 8
lives > 0 ?
YES
userInput
YES
userInput contains 0?
NO
YES
userInput ∉ {R,O,Y,G,B,I,V} ?
NO
YES
YOU LOOSE....
YES
EXIT PROGRAM
userInput contains different number of characters than 4?
NO
userInput ∈ previousInputs?
NO
userInput = randomCode?
save useInput to previousInputs
check if any of letters is guessed
check for clues
lives = lives -1
display message:
- lives
- letters guessed
- clues
YES
YOU WON !!!
Play Again?
YES
NO
EXIT PROGRAM

3.3. Database or data definitions used by the code

3.3.1. App class

- int **lives**
  Initial number of lives - game resolutions until it ends, unless guessed a code.
- int **codeLength**
  Number of characters in code patch, as well as number of characters user should input, when guessing
- int **currentLives**
  Current number of lives - it's decreasing with every Input that doesn't match a code.
  When it reaches 0, game ends
- int **hint**
  Number of clues - characters user entered, that are present in the code, but in different location
- String **userInput**
  Stores user Input
- char **firstLetter**
  first letter of user Input, to determine if user wants to play again.
- String **aMessage**
  message displayed, when user's input is incorrect
- boolean **codeGuessed**
  States if users input matches random code
- boolean **badInput**
  States if user Input is Valid
- boolean **codeUsed**
  States if user Input has been already entered this game
- char[][] **userCodes**
  An Array to store all user Inputs in this game
- char[] **userCode**
  User Input string converted to array
- char[] **partCode**
  An array which allows to display letters which have been guessed in the code
- char[] **randomCode**
  An array that stores random code of letters

3.3.2. LettersGenerator Class

- int **randomNum**
  randomly generated number.
- char **randomLetter**
  randomly generated letter.
- char[] **letters**
  An array to store allowed characters.
- char[] **randomLetters**
  An array to store random Code - set of randomly picked letters from allowed set.

3.3.3. Validator Class

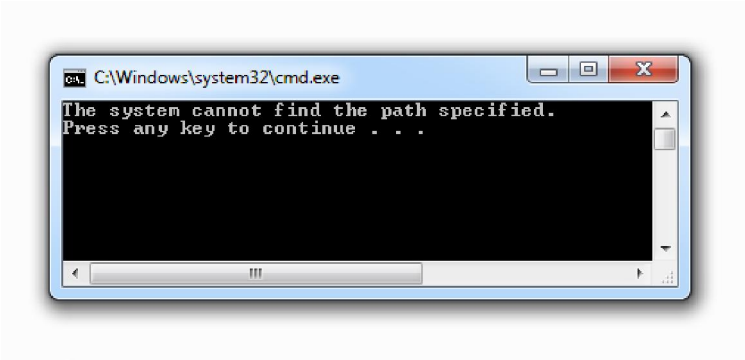
- boolean[] **letterExists**
  an Array used to check if code patch have been already used - comparing arrays
- boolean[] **used**
  an Array used when checking for clues.
- int **hint1**
  counts number of clues
- int **counter**
  used to check if code patch have been already used - comparing arrays
- boolean **arrayExists**;
  states if given array exists in set of arrays - used to check if code patch have been already used - comparing **arrays**
- boolean samesame;
  states if two compared arrays are the same.
- String **message**
  stores message to display when validating user input

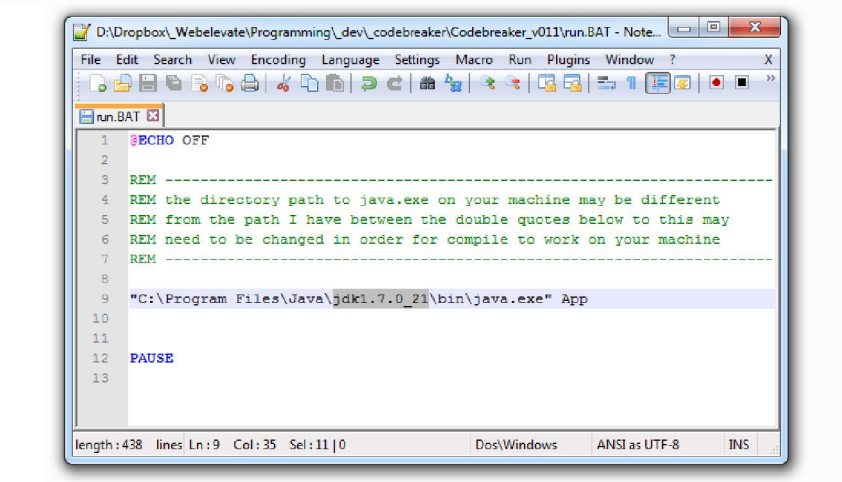
3.4. Error Handling and limitations of the design

There is Input validation functionality implemented in the code.
When character which does not belong to set of allowed characters is entered, user is prompted of 'invalid character' and asked to enter his guess again.
That allows to handle any random keys entered by user.

There is only one exception - when Input field is left blank and 'ok' button is hit, then Input box disappears.
This might be due to Javabook InputBox limitations.

Due to different java versions, sometimes it might be necessary to edit run.bat file and give the correct path to actual system specific java folder.





3.5. File structure of the source and build code

Game directory contains only one subfolder which contains Javabook library.
Other classes and editable .java files are stored directly in main game directory/

3.6. Installation instructions

There is no need for installation. Just execute 'run.bat' file.

3.7. Maintenance instructions

3.7.1. Configuration options

Number of lives and Code Length can be changed by changing value of two variables in App Class:

- lives
- codeLength

```
17
18    int lives = 8;
19    int codeLength = 4;
20
```

3.7.2. Housekeeping requirements

Please do not change any file names inside game folder.

The game is designed to be ready to play multiple times - there is no need for any special garbage collectors.
All Arrays and variables are reseted with every new game.

3.7.3. Extending the code, best approaches

Game can be easily extended by adding new letters into allowed pool.
It can be done by adding new positions to **'letters'** array in **LetterGenerator** class.

```
public LettersGenerator()
{
  letters = new char[8];
  letters[0] = 'R';
  letters[1] = 'O';
  letters[2] = 'Y';
  letters[3] = 'G';
  letters[4] = 'B';
  letters[5] = 'I';
  letters[6] = 'V';
 letters[7] = 'X';
}
```

4. Code

4.1. Methods in App Class

4.1.1. App()

```
public App()
	{

		mBox = new MainWindow();
		iBox = new InputBox(mBox);
		oBox = new OutputBox(mBox,980,650, "The Code Breaker Game");

		mBox.show();
		gameBoard();

		System.exit(0);
	}
```

4.1.2. gameBoard()

The only purpose of this method is to ask player at the end if he/she wishes to play again.
If Yes - new game is launched, if not - program exits.

```
private void gameBoard()
	{
			boolean playAgain = false;
			String userInput;
			char firstLetter;
		do
		{
			oBox.clear();
			playGame();

			userInput = iBox.getString("Play Again?  Y / N");

			firstLetter = userInput.charAt(0);
			if (firstLetter == 'Y' || firstLetter == 'y')
				{
					playAgain = true;
				}
			else
			{
				playAgain = false;
			}

		}
		while (playAgain);

	} //end of GameBoard
```

4.1.3. playGame()

This is the main game method.
It's main functionality is as follows:

- user Input
- pass user Input to Validator Class for validation
- save user string into array
- pass user code array to Validator Class for checking if it has been already used
- pass user code array to Validator Class for checking if it maches random code
- ask Validator Class for clues
- management of currentlives
- displaying appropriate messages

```
public void playGame()
    {
    boolean codeGuessed = false;
    boolean badInput = false;
    boolean codeUsed = false;
    String aMessage = " ";

    // reseting arrays
    for (int q=0; q<codeLength; q++)
    {
       userCode[q] = '_';
       partCode[q] = '_';
    }
    // reseting current Lives
    currentLives = lives;


    //get random code array
    randomCode = new char[codeLength];
    randomCode = code.getLetters(codeLength);

   // for testing purposes print generated code
    System.out.println(randomCode);


    mBox.setVisible(true);
    oBox.setVisible(true);

     startScreen();


  for (int x=0; x<lives; x++)
  {
      gameMessage();

      aMessage = " ";
    do
       {

          //get user input
          userString = iBox.getString(aMessage + "please enter your guess");
```

```
      //validate input
      badInput = hal.validateCode(userString, code.letters.length, codeLength);
      aMessage = hal.getMessage();
      if (badInput == false)
      {
        //save input string into Array 'userCode'
         for (int i=0; i<codeLength; i++)
        {
            userCode[i] = Character.toUpperCase(userString.charAt(i));
        }

        //check if userCode has been used already
        codeUsed = hal.compare2Arrays(userCode, userCodes);
        if(codeUsed)
        {
           badInput = true;
           aMessage = "Code already used. ";
        }
      }
    }
   while(badInput);



//check if WIN - Compare userCode Array with randomCode Array
 codeGuessed = hal.compareArrays(userCode,randomCode);


// check if any of the letters is correct (partial guess)
  for (int a=0; a<codeLength; a++)
  {
     if(userCode[a] == randomCode[a])
     { partCode[a] = randomCode[a]; }
     else
     { partCode[a] = '_';}
  }

 // check for hints
 hint = hal.getHints(userCode,randomCode,codeLength);


//IF win - end Game Loop
if (codeGuessed)
{break;}
else
{
   // save userCode into array with all entered codes
    for (int i=0; i<codeLength; i++)
      {
         userCodes[x][i] = userCode[i];
      }

    currentLives--;
```

```
        }

    } // end of main game loop

    gameMessage();

      if (codeGuessed)
      {
         oBox.println("*****************************");
         oBox.println("Congratulations, You WON!!!!!");
         oBox.println("*****************************");
      }
     else
      {
         oBox.println("You Loose....");
         oBox.println("Code Was: " );
         for (int i=0; i<codeLength; i++ )
             {
               oBox.print(randomCode[i] + " ");
             }
      }

}
```

4.1.4. startScreen()

Generating Initial message with game instructions.

```
Public void startScreen()
{
  oBox.clear();
    oBox.println("Try to guess the sequence of 4 colours.");
    oBox.println("The possible colours are");
    oBox.println("R - Red");
    oBox.println("O - Orange");
    oBox.println("Y - Yellow");
    oBox.println("G - Green");
    oBox.println("B - Blue");
    oBox.println("I - Indigo");
    oBox.println("V - Violet");
    oBox.println("You have " + lives + " chances to guess the code.");
     oBox.println("-----------------------------------------------------");
    oBox.println("Please enter " + codeLength + " characters,");
    oBox.println("At any time you can enter '0' to exit the program.");
    oBox.println("Characters allowed:");
    for (int t=0; t<code.letters.length; t++)
     { oBox.print(code.letters[t] + " ");}

    oBox.println(" ");
    oBox.println("-----------------------");
    oBox.println("Start: ");
     }
```

4.1.5. gameMessage()

Generating in-game messages with guessed code status, actual lives, clues, etc.

```
public void gameMessage()
 {
    oBox.print("You entered: ");
      for (int u=0; u<codeLength; u++)
      {
         oBox.print(userCode[u] + " ");
      }
        oBox.print(" || ");

        oBox.print("Code: ");
        for (int k=0; k<codeLength; k++)
          {
            oBox.print(partCode[k] + " ");
          }
        oBox.print(" || ");

        oBox.print("Hints: " + hint);
        oBox.print(" || ");

        oBox.print("Lives: " + currentLives);
        oBox.println(" ");
 }
```

4.1.6. main()

Just to lunch App

```
public static void main(String[] args)
 {
      App codeBreaker = new App();
 }
```

4.2. Validator Class

4.2.1. validateCode()

Validating user entry. Returns boolean value informing if users entry meets requirements.

```
public boolean validateCode(String userString, int lettersPool, int aCodeLength)
  {
    LettersGenerator letGen = new LettersGenerator();
    boolean invalidInput = false;

     //exit at 0
      for (int d=0; d<userString.length(); d++)
      {
       if (userString.charAt(d) == '0')  { System.exit(0); }
      }

       // check string length
       if (userString.length() != aCodeLength)
          {
           message = "You are supposed to enter 4 colours. ";
           invalidInput = true;
          }
       else
       {
          // check if used allowed characters
          for (int i=0; i<aCodeLength; i++)
          {
            char userLetter = Character.toUpperCase(userString.charAt(i));

             for (int j=0; j<lettersPool; j++)
             {
                if (userLetter != letGen.letters[j] )
                {
                   invalidInput = true;
                   message = "Invalid character used. ";

                }
                else {invalidInput = false; break;}
             }
             if (invalidInput == true) {break;}
          }
       }
    return invalidInput;
  }
```

4.2.2. getMessage()

Passes message about incorrect entry to main game class (App)

```
public String getMessage()
  {
   return message;
  }
```

4.2.3. compareArrays()

Compares two given arrays, returning bollean 'true' if they are the same.

```
public boolean compareArrays(char[] array1, char[] array2)
  {
    boolean samesame = false;
    if (array1.length != array2.length)
    {System.exit(0);}
    else
    {
      for (int i=0; i<array1.length; i++)
      {
        if (array1[i] == array2[i])
        {
          samesame = true;
        }
        else
        {
        samesame = false;
        break;
        }
      }
    }

   return samesame;
  }
```

4.2.4. compare2Arrays()

Comparing given array, with given 2-dimensional array.
Returns bollean value - 'true' if given array exists inside 2-dimensional array.

```
public boolean compare2Arrays(char[] arrayA, char[][] arrayB)
  {
    boolean arrayExists = false;
    boolean[] letterExists = new boolean[arrayA.length];
    int counter = 0;

    for (int i=0; i<arrayB.length; i++)
    {


      for(int j=0; j<arrayA.length; j++)
        {
          if(arrayA[j] == arrayB[i][j])
```

```
                {
                  letterExists[j] = true;
                }
                else
                {
                  letterExists[j] = false;
                }
            }
            counter = 0;

            for (int e=0; e<letterExists.length; e++)
              {
                 if(letterExists[e] == true)
                     {counter = counter + 1;}
              }
            if (counter == letterExists.length)
              {
                 arrayExists = true;
                 break;
              }
              else
              {
                 arrayExists = false;
              }
           }
          return arrayExists;
     }
}
```

4.2.5. getHints()

Calculate value of hints and passes it back to main game class.

```
public int getHints(char[] userCode, char[] randomCode, int codeLength)
{
  int hint1=0;
      boolean[] used = new boolean[]{false,false,false,false,};

      for (int b=0; b<codeLength; b++)
      {
         if (userCode[b] != randomCode[b])
         {
          for (int c=0; c<codeLength;c++)
          {
             if (userCode[b] == randomCode[c] && randomCode[c] != userCode[c] && used[c] == false)
             {
                hint1++;
                used[c] = true;
                break;
             }
          }
         }
      }
 return(hint1);
}
```

4.3. LettersGeneratorClass

4.3.1. LettersGenerator()

This array contains collection of allowed characters.

```
public LettersGenerator()
   {
    letters = new char[7];
    letters[0] = 'R';
    letters[1] = 'O';
    letters[2] = 'Y';
    letters[3] = 'G';
    letters[4] = 'B';
    letters[5] = 'I';
    letters[6] = 'V';
   }
```

4.3.1. getRandom()

Generates random number.

```
public int getRandom(int sides)
{
return( 1 + (int) (Math.random() * sides));
}
```

4.3.2. getLetters()

Generates array of random characters.

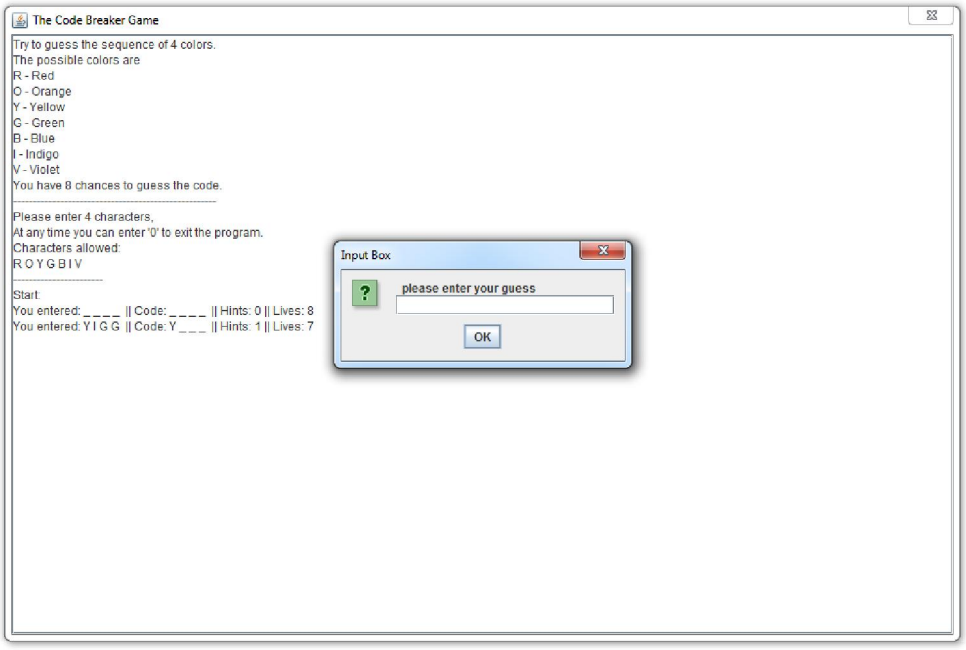
```
   public char[] getLetters(int aCodeLength)
   {
      char[] randomLetters;
      randomLetters = new char[aCodeLength];

      char randomLetter;

      for (int i=0; i<aCodeLength; i++)
        {
        int randomNum = getRandom(letters.length)-1;
        randomLetter = letters[randomNum];
        randomLetters[i] = randomLetter;
        }


      return randomLetters;
   }
```

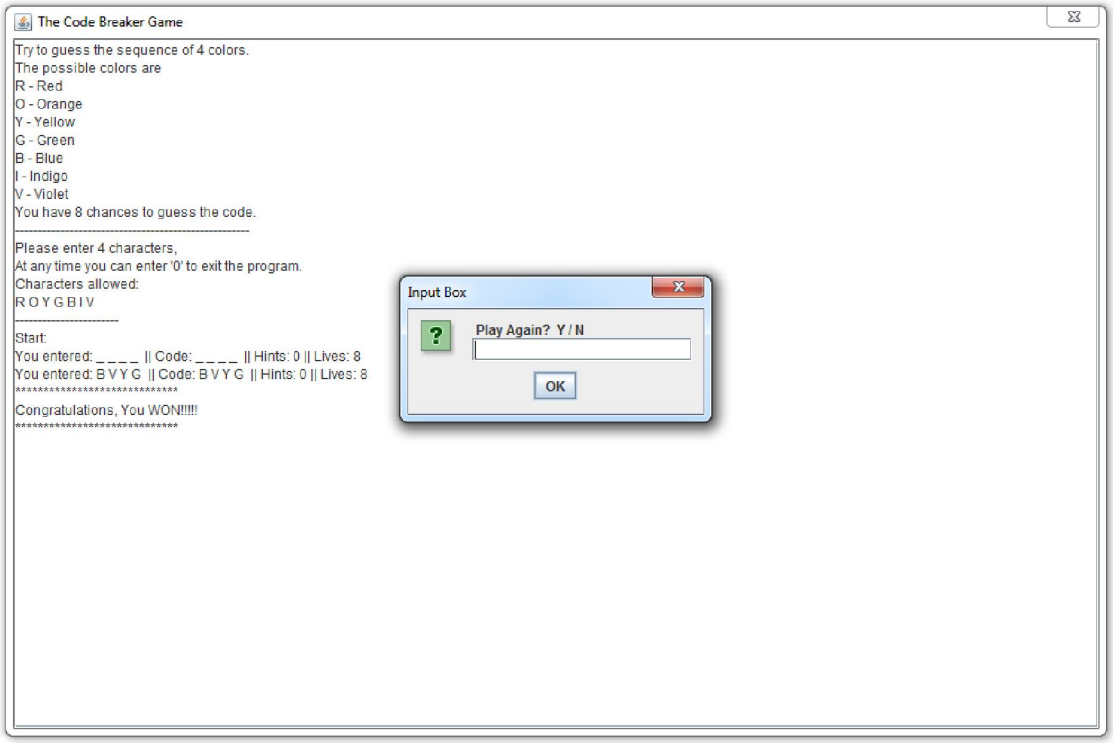
5. Test Records

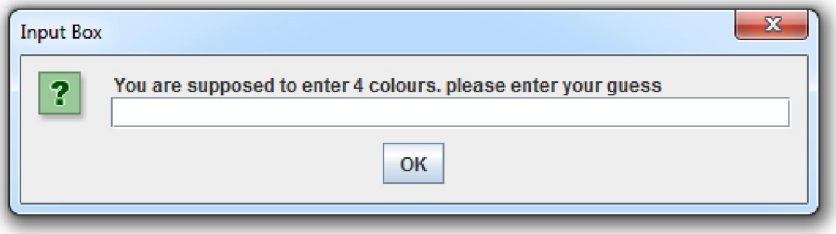
TESTING

Code: WE21 CodeBreaker
Version: 11
Author: Gabriel Garus
Tester: Gabriel Garus
Test Date: 2013-07-14

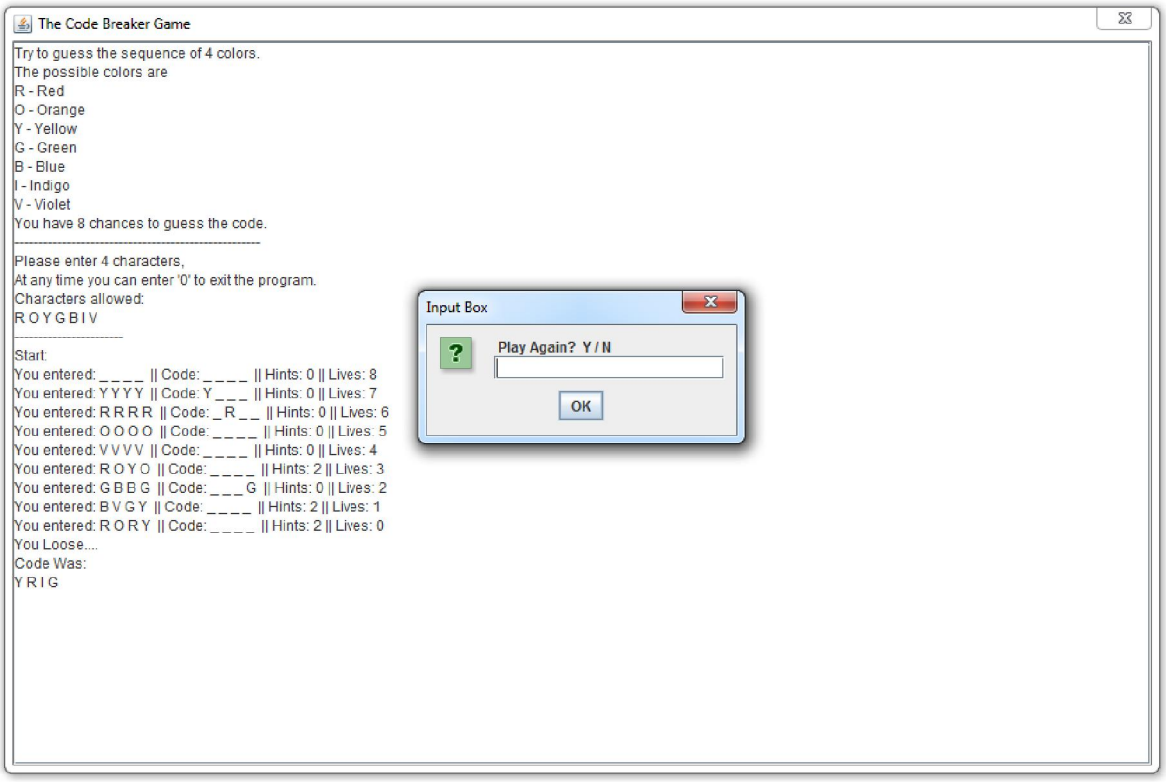
Test Case	Entered Value	Random Code	Expected Result	Actual Result	test result	Comments
010	rRybOV	n/a	please try again	please try again	pass	
020	Y	n/a	please try again	please try again	pass	
030	F	n/a	please try again	please try again	pass	
040	!%|	n/a	please try again	please try again	pass	
050	0	n/a	exit game	exit game	pass	
060	rb0	n/a	exit game	exit game	pass	
070		n/a	please try again	InputBox disappears	fail	javabook limitations
080	bvyg	BVYG	You Won!	You Won!	pass	
090	yigg	YROI	Y___, clues: 1	Y___, clues: 1	pass	
010	YYYY	YRIG	Y___, clues: 0	Y___, clues: 0	pass	
011	YYYY	YRIG	code used	code used	pass	
012	8th trial, oooo	YRIG	You Loose	You Loose	pass	

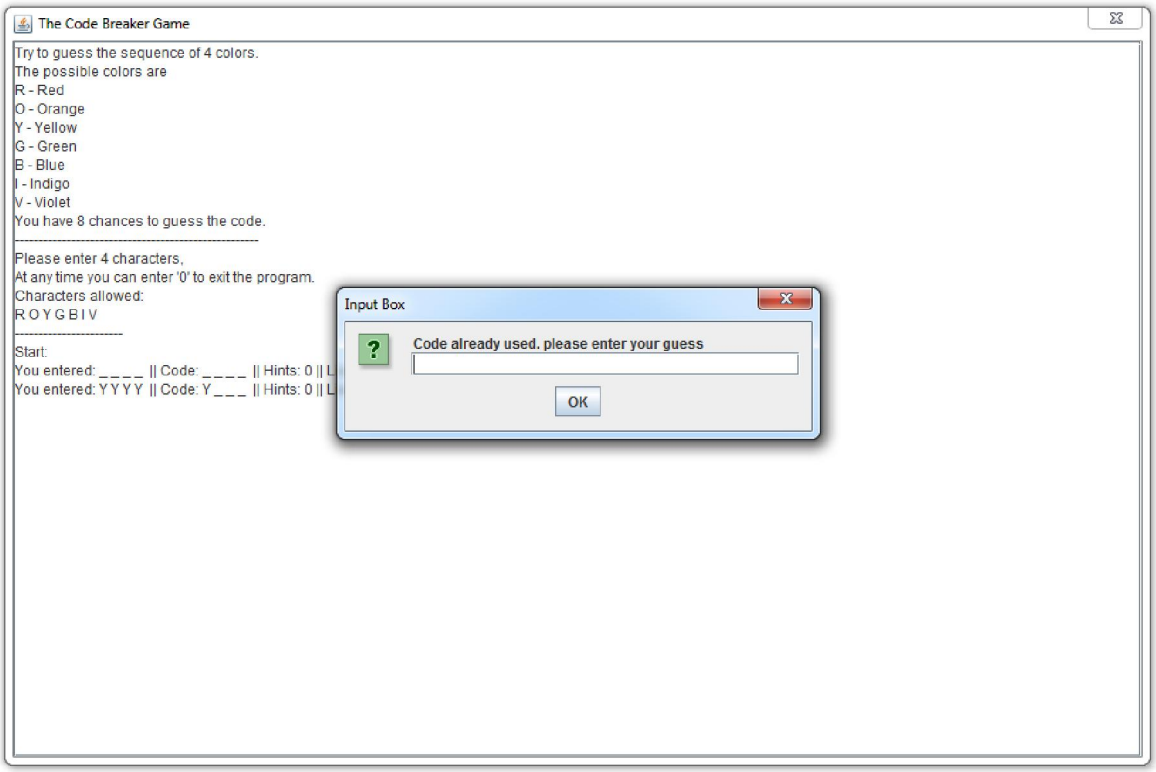



The Code Breaker Game
Try to guess the sequence of 4 colors.
The possible colors are
R - Red
O - Orange
Y - Yellow
G - Green
B - Blue
I - Indigo
V - Violet
You have 8 chances to guess the code.
Please enter 4 characters,
At any time you can enter '0' to exit the program.
Characters allowed:
R O Y G B I V
Start:
You entered: _ _ _ _ || Code: _ _ _ _ || Hints: 0 || Lives: 8
You entered: B V Y G || Code: B V Y G || Hints: 0 || Lives: 8
Congratulations, You WON!!!!!
Input Box
Play Again? Y / N
OK


Input Box
You are supposed to enter 4 colours. please enter your guess
OK


Input Box
Invalid character used. please enter your guess
OK


The Code Breaker Game
Try to guess the sequence of 4 colors.
The possible colors are
R - Red
O - Orange
Y - Yellow
G - Green
B - Blue
I - Indigo
V - Violet
You have 8 chances to guess the code.
Please enter 4 characters,
At any time you can enter '0' to exit the program.
Characters allowed:
R O Y G B I V
Start:
You entered: _ _ _ _ || Code: _ _ _ _ || Hints: 0 || Lives: 8
You entered: Y Y Y Y || Code: Y _ _ _ || Hints: 0 || Lives: 7
You entered: R R R R || Code: _ R _ _ || Hints: 0 || Lives: 6
You entered: O O O O || Code: _ _ _ _ || Hints: 0 || Lives: 5
You entered: V V V V || Code: _ _ _ _ || Hints: 0 || Lives: 4
You entered: R O Y O || Code: _ _ _ _ || Hints: 2 || Lives: 3
You entered: G B B G || Code: _ _ _ G || Hints: 0 || Lives: 2
You entered: B V G Y || Code: _ _ _ _ || Hints: 2 || Lives: 1
You entered: R O R Y || Code: _ _ _ _ || Hints: 2 || Lives: 0
You Loose....
Code Was:
Y R I G
Input Box
Play Again? Y / N
OK


The Code Breaker Game
Try to guess the sequence of 4 colors.
The possible colors are
R - Red
O - Orange
Y - Yellow
G - Green
B - Blue
I - Indigo
V - Violet
You have 8 chances to guess the code.
Please enter 4 characters,
At any time you can enter '0' to exit the program.
Characters allowed:
R O Y G B I V
Start:
You entered: _ _ _ _ || Code: _ _ _ _ || Hints: 0 ||
You entered: Y Y Y Y || Code: Y _ _ _ || Hints: 0 ||
Input Box
Code already used, please enter your guess
OK