**CS-200-1: Programming I**

**Fall 2018**

**Northeastern Illinois University**

**Programming Project: OpenGate**

**Introduction**

OpenGate is a two-player game with two significant names for each player: **player vs computer.**

This project has multiple purposes that are related to electricity physics (electrical power), and binary technology and how it is related to electrical circuits. To win this game or to open the gate, the user (player or computer) has to be entering 3 digits to (1st 2nd 3rd) to match the 3 digit secretive number from the **CodeTable(an electronic table that consists of 9 digits from 1 to 9 each digit could be considered as a button**), **and this is the first condition to win the game.**

**Second condition**, the user has to the keyboard to enter an electrical power of their choice, and if they enter a positive number, the binary circuit would equal 1, and that is the second condition to open the electrical gate. Otherwise, if they enter an electrical power of 0, means the circuit receives a binary sign of 0, which means the game has to be restarted because only one condition is met.

The rules as follows:

* Each player takes a turn entering their digits.
* **Player always goes first (computer second) --- Player has more chance to win because they start first!!**
* Assume the CodeTable always consists of **non-repeated 9 digits from 1 to 9 not necessarily in order.**
* The players **do not know the order of CodeTable digits.** Otherwise, it would be easy for them to know the Major Diagonal.
* The game ends when a player meets the two conditions mentioned above in the introduction.
* Winning combination is defined as Three diagonally adjacent symbols (**only the Major Diagonal** of the CodeTable (major diagonal of a table or 2D array **the diagonal that goes from left to right**).
* If neither player creates a winning combination when all **3 digits** are entered by **each** player, the game prints “Game over, no one wins!”.

**Java Project File Instructions**

* Create a class named OpenGate that has the following:

**Step1**

* Create a main method and leave it empty for now.

**Step2**

* Create a void method named NewGame() that does the following :

1. Prints the design of how a **new game** always starts as shown below:

|  |
| --- |
| --------- NEW GAME ---------   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Hello!! Welcome to the OpenGate game! Ready to play?   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

1. Calls the method TableLoop(). (this method explanation is coming forward)

**Step3**

* Create a method named PlayerCode that:
* Takes an integer and returns an integer.
* The method asks the keyboard to enter an integer and return that integer.
* The taken integer parameter references the order of digit (digit1, digit2, digit3)
* Sample output is shown below:

|  |  |  |
| --- | --- | --- |
| Sample Method Usage | Sample output | Return value |
| **int** player = kbd.nextInt();  **int** p =1;  PlayerCode(p); | Enter digit1 for player (1-9): 3 | 3 |
| **int** player = kbd.nextInt();  **int** p =3;  PlayerCode(p); | Enter digit3 for player (1-9): 7 | 7 |

**Step4**

* Create a method named ComputerCode that:
* Takes an integer and returns an integer.
* The method asks the keyboard to enter an integer from **1 to 9** and returns that integer.
* The taken integer parameter references the order of digit (digit1, digit2, digit3)
* Sample output is shown below:

|  |  |  |
| --- | --- | --- |
| Sample Method Usage | Sample output | Return value |
| **int** computer = kbd.nextInt();  **int** c =2;  ComputerCode(c); | Enter digit2 for computer (1-9): 9 | 9 |
| **int** computer = kbd.nextInt();  **int** c =1;  ComputerCode(c); | Enter digit1 for computer (1-9): 2 | 2 |

**Step5**

* Create a method named Power that:
* Takes an integer and does not return anything.
* The method asks the keyboard to enter an integer from **1 to 9** and returns that integer.
* The return integer returns the electrical power entered or guessed by the user.
* Sample output is shown below:

|  |  |  |
| --- | --- | --- |
| Sample Method Usage | Sample Output | Return Value |
| **int** n=kbd.nextInt(); | Guess the electrical power of the circuit >=0: 4 | 4 |

**Step6**

* Create a method named BinaryCircuit that:
* Takes an integer and returns an integer.
* The method asks the keyboard to enter an integer and return that integer.
* The method assigns 0 to the returned integer if the parameter passed equals 0. Otherwise, it assigns 1 to the returned integer if the parameter passed equals 1.
* The return integer returns the status of the binary circuit. (0 or 1)

**Step7**

* Create a method named AlmostWinner that:
* Takes a String and does not return anything.
* The method does a process after the user (player or computer) gets **all the three** digits **correct and** does some printing statements.
* The method checks if the electrical power entered by the user from the method Power

equals 0, and if it does, the status of the binary circuit will be printed out as 0 by calling the method BinaryCircuit, and a statement of restarting the game will be printed. Otherwise, if the power entered is greater than 0, the status of the binary circuit will be printed out as 1 by calling BinaryCircuit method, and a congratulations statement will be printed for the winner.

* The String parameter taken is just to be printed out as **player** or **computer** depending on whoever wins.
* The method calls at the end the method NewGame to automatically start **a new game** after **winning or restarting** the game.
* A sample output is shown below

|  |  |  |
| --- | --- | --- |
| Sample Output | Sample Method Usage | Sample Output |
| Wow!! you are about to win!! That is the right code!! make sure the electrical circuit is ON | Power()=0;  BinaryCircuit(0); | The status of circuit (binary 0 or 1): 0  Oops! sorry! Circuit needs to be turned on, restart the game |
| Wow!! you are about to win!! That is the right code!! make sure the electrical circuit is ON | Power()=5;  BinaryCircuit(5);  AlmostWinner(“player”); | The status of circuit (binary 0 or 1): 5  Congratulations player! You opened the gate! Winner! |
| Wow!! you are about to win!! That is the right code!! make sure the electrical circuit is ON | Power()=5;  BinaryCircuit(5);  AlmostWinner(“computer”); | The status of circuit (binary 0 or 1): 5  Congratulations computer! You opened the gate! Winner! |

**Step8**

* Create a method named TableLoop that:
* Creates two variables integers countPlayer and countComputer that keep track of **counting the digits entered for each player that match the Major Diagonal of The Code Table**.
* Calls the method WinningCodeTable (this method is very secretive, the players cannot see this method because it contains the CodeTable digits in order, the players must be guessing the digits from 1 to 9 to match the Major Diagonal of the table) to access the digits of the table.
* The method keeps comparing the major diagonal elements of the table code to the digits entered by either the player or computer and incrementing the counters (countPlayer or countComputer) at each digit entered matches the Major Diagonal element of the Code Table.
* If one of the counters (countPLayer goes first because player goes first) reach 3(3 correct digits), the method calls the method AlmostWinner to finish the process of reaching the edge of winning. Otherwise, Game is over since no one wins, a then calling NewGame method to **start a new game**.
* Sample output is shown below.

|  |  |  |
| --- | --- | --- |
| Sample Method Usage | Sample Output | Comments |
| **int** [] [] codeTable= {  {1,2,3}  , {4,5,6},  {7,8,9}};  (getting the array from WinningCodeTable method)  **int** i=1; (the parameter passed by either PlayerCode or ComputerCode -----index where the user start entering digits from---digit**1)** | Enter digit1 for player (1-9): 1 | /\* 159 is the winning code for this example  **The MajorDiagonal** \*/  PlayerCode(1)=1;(digit entered)  countPlayer =1 (it will increment from 0 to 1, because a [0][0],  which is the first element of the Major Diagonal, matches the digit entered by the player who has entered 1 (PlayerCode(1)=1) |
| **int** [] [] codeTable= {  {1,2,3}  , {5,4,6},  {9,8,7}};  (getting the array from WinningCodeTable method)  **int** i=1, i=2, i=3; (the parameter passed by either PlayerCode or ComputerCode---index where the user start entering digits from---digit**1,** digit**2**, digit**3)**  AlmostWinner(“computer”); | Enter digit1 for computer (1-9): 1  Enter digit2 for computer (1-9): 4  Enter digit3 for computer (1-9): 7  Wow!! you are about to win!! That is the right code!! make sure the electrical circuit is ON  Guess the electrical power of the circuit (>=0): 4  the status of circuit (binary 0 or 1): 1  Congratulations computer! You opened the gate! Winner! | /\* 147 is the winning code for this example  **The MajorDiagonal** \*/  PlayerCode(1)=1, PlayerCode(2)=4,  PlayerCode(3)=7;  (digit entered)  countPlayer =3 (it will increment from 0 to 3 three times, because all the Major Diagonal elements of the table  match all the digits entered by the computer |
| **int** [] [] codeTable= {  {1,2,3}  , {5,4,6},  {9,8,7}};  (getting the array from WinningCodeTable method)  **int** i=1, i=2, i=3; (the parameter passed by either PlayerCode or ComputerCode---index where the user start entering digits from---digit**1,** digit**2**, digit**3)** | Enter digit1 for player (1-9): 1 Enter digit1 for computer (1-9): 2  Enter digit2 for player (1-9): 4 Enter digit2 for computer (1-9): 3 Enter digit3 for player (1-9): 6 Enter digit3 for computer (1-9): 7  Game Over! No one wins! | /\* 147 is the winning code for this example  **The MajorDiagonal** \*/  PlayerCode(1)=1, PlayerCode(2)=4,  PlayerCode(3)=6;  ComputerCode(1)=2, ComputerCode(2)=3,  ComputerCode(3)=7;  (digit entered)  countPlayer =2 (it will increment from 0 to 2 two times, because two of the Major Diagonal elements of the table  match two of the digits entered by the computer  countComputer=1(it will increment from 0 to 1 since only one of the Major Diagonal elements of the table  match of the digits entered by the computer |

* Call the method NewGame from the main method to start the first new game (don’t worry about the coming games, it will call NewGame from other methods.
* Combine all the methods above appropriately.