**Casino Assignment 2**

**Due, Monday, October 2, 2020 for maximum 100%**

**Tuesday, October 3, 2020 for maximum 90%**

**Wednesday, October 4, 2020 for maximum 80%**

**Thursday, October 5, 2020 for maximum 70%**

**Assignment Scope**

1. Netbeans IDE familiarity
2. Update a class
3. Add custom constructor to class
4. Add member variables to class
5. Add methods to class
6. Use class Random
7. Use arrays
8. Compile and run a project
9. Compress a project and submit to Webcourses
10. Decompress compressed project and verify it is a Netbeans project

**References**

1. Netbeans.docx
2. Setting up a project in Netbeans.docx
3. Netbeans right click menu help.docx

**Deliverables**

To complete this assignment you must submit your **compressed Netbeans project** to Webcourses.

**Tasks**

|  |  |
| --- | --- |
| Activity | |
| Casino project | Copy previous assignment to a new project (i.e. copy the project folder and modify the new project) |
| casino package |  |
| Casino class | 1. Update the switch statement for case label **Constants.SLOTS** to do the following    1. Add the member variable of class **Player** as an argument to the constructor for class **Slots**    2. Call method **play** in class **Slots** |
| Player class | 1. Update class **Player** to do the following    1. Add member variables with access level modifier **private**       1. **String** **name**       2. **int** **cash**       3. **Scanner** **scan**    2. Generate getters/setters for member variables **name** and **cash**    3. Write custom constructor **Player** to do the following       1. Access level modifier **public**       2. Instantiate the member variable instance of class **Scanner**       3. Declare local variable **name** data type class **String**       4. Declare local variable **money** data type integer       5. Prompt the player to enter their name       6. Set variable **name** equal to member variable **scan** method **next**()       7. Prompt the player to enter their cash to play with       8. Set variable **cash** equal to member variable **scan** method **nextInt()**       9. Call setter for member variable **name**, passing **player** as an argument       10. Call setter for member variable **cash**, passing **money** as an argument |
| constants package |  |
| Constants class | Update class to do the following:   1. Add constant **BET** initialize to value **5** 2. Add constant **PAIR\_PAYOUT** initialized to value **5** 3. Add **contant** **TRIPLE\_PAYOUT** initialized to value **50** |
| slots package |  |
| Slots class | Update class to do the following:   1. Add member variables with access level modifier **private**    1. Constant integer **SLOTS** initialized to value **3**    2. Constant character array **SYMBOLS** initialized to **'$', '%', '&', '#', '@', '!'**    3. Character array **spins**    4. Class **Random rand**    5. Class **Player player**    6. **boolean play**    7. Class **Scanner scan** 2. Write custom constructor **Slots** to do the following    1. Access level modifier **public**    2. Parameter list receives class **Player**    3. Set member variable **player** equal to parameter    4. Instantiate member variable class **Scanner**, pass **System.in** as an argument    5. Initialize member variable **play** with value **true**    6. Instantiate member variable class **Random**    7. Instantiate member variable **spins** with size **3** 3. Write method **play** to do the following:    1. Access level modifier **public**    2. Return type **void**    3. Empty parameter list    4. Declare variable **input** data type integer    5. Output to the screen game rules similar to **Figure 4**    6. Write an **if** statement to check if the player’s cash is less than the minimum $5 bet; if true, output to the screen that the player doesn’t have even cash to play and set variable **play** equal to **false**    7. Write a **while** loop based on the condition that variable **play** is true       1. Set the player’s cash equal to the current value minus the bet amount       2. Output to the screen that the slots are spinning similar to **Figure 5**       3. Write a for loop to iterate through the size of the **spins** array (i.e. constant **SLOTS**)          1. Set the current index of the array equal to method call **randomSymbol**          2. Output to the screen the symbol stored in the current index of the array similar to **Figure 5**       4. Write an **if/else if/else** statement to evaluate the following          1. Outputs to the screen should be similar to **Figure 6**          2. All three symbols matched; output to the screen that the player won $50; update the player’s cash to add $50          3. Two of the three symbols matched; output to the screen that the player won $5; update the player’s cash to add $5          4. None of the symbols matched; output to the screen that the player didn’t receive any money       5. Output to the screen the player’s current cash balance similar to **Figure 7**       6. Write an **if** statement to evaluate if the player’s cash balance is at least $5 to continue playing          1. If true, output to the screen to prompt the player if they want to spin again similar to **Figure 7**          2. Store player’s response in variable **input** set equal to class **Scanner** method **nextInt()**             1. Write an **if** statement to evaluate if variable **input** is equal to 1; if **true**, set variable **play** equal to **true**; otherwise set variable **play** equal to **false**             2. **else**, **break** out of the loop    8. Output to the screen an exit statement similar to **Figure 8** 4. Write method **randomSymbol** to do the following    1. Access level modifier **private**    2. Declare variable **num** data type integer    3. Declare variable **symbol** data type character initialized to an empty value in **‘ ‘**    4. Set variable **num** equal to class **Random** method **nextInt** passing as an argument constant array **SYMBOLS** property **length**    5. Write a **switch** statement to evaluate the variable **num**       1. Each **case** label should set the variable **symbol** equal to the corresponding index in the array **SYMBOLS** (e.g. case 0: symbol = SYMBOLS[0];)    6. Return variable **symbol** |
| Casino application |  |
| Test Case 1 | Test Case 1 passes |
| Test Case 2 | Test Case 2 passes |
| Test Case 3 | Test Case 3 passes |
| Test Case 4 | Test Case 4 passes |
|  | Source compiles with no errors |
|  | Source runs with no errors |
|  | Source includes comments |
| Total |  |

**Perform the following test cases**

|  |  |  |
| --- | --- | --- |
| Test Cases | | |
|  | **Action** | **Expected outcome** |
| Test Case 1 | **Project view** | Completed project view should look like Figure 1 |
| Test Case 2 | **Run executable**  **User enters a name and clicks <Enter>**  **User enters a cash value as an integer and clicks <Enter>** | The console window should look similar to **Figure 2**  then **Figure 3** |
| Test case 3 | **User enters value 3 and clicks <Enter>** | The console window should look similar to **Figure 4** |
| Test case 4 |  | The console window should look similar to **Figure 5** |
| Test case 5 |  | The console window should look similar to **Figure 6** |
| Test case 6 |  | The console window should look similar to **Figure 7** |
| Test cast 7 | **User enters the value 1 <Enter>** | The console window should look similar to **Figures 5, 6, 7** |
| Test case 8 | **User enters the value 0 <Enter>** | The console window should look similar to **Figure 9** |
| Test case 9 | **User’s cash balance is less than $5** | The console window should look similar to **Figure 8** |
| Test case 10 | **User enters a value less than 1 or greater than 3** | The game menu is displayed again until the user enters a valid value |

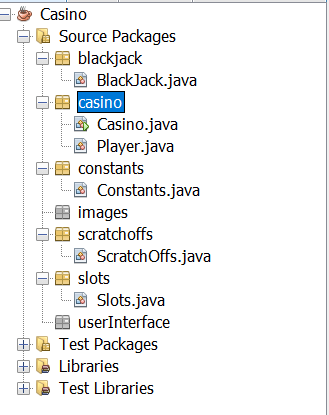


Figure 1 Project View

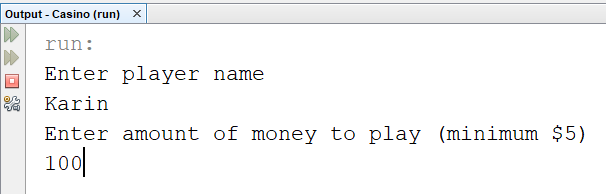


Figure 2 Class Player constructor prompts

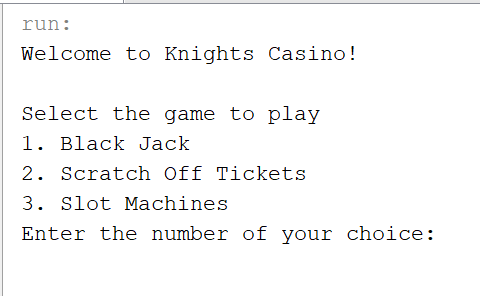


Figure 3 displayMenu method output in console window

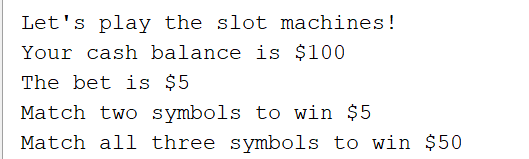


Figure 4 Slots game rules

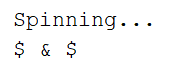


Figure 5 Random symbols



Figure 6 Comparison evaluation

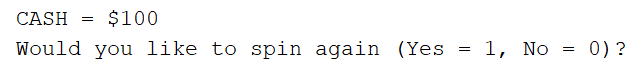


Figure 7 Cash balance and continue

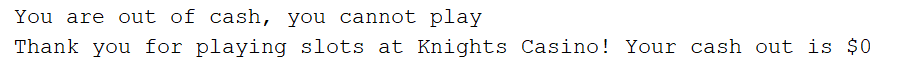


Figure 8 Cannot play



Figure 9 Exit statement