**Assignment 2**

**Due, Wednesday, June 6, 2018 for maximum 100%**

**Thursday, June 7, 2018 for maximum 90%**

**Friday, June 8, 2018 for maximum 80%**

**Saturday, June 9, 2018 for maximum 70%**

**Assignment Scope**

1. Add member variables to classes
2. Generate getters/setters for member variables
3. Write custom constructors
4. Add method signatures to an interface
5. Implement an interface so that methods are declared as abstract
6. Implement abstract methods in inherited classes
7. Write methods to accomplish specific tasks for a class
8. Create enumerations as constants
9. Compile and run a project
10. Compress a project and submit to Webcourses
11. 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 | |
| Euchre project | Create a new Java Application project named Euchre  allowing Netbeans IDE to create the main class called Euchre.java |
| Euchre class | Update main() method to do the following:   1. Call static method System.out.println() output to the console “Welcome to Euchre!” 2. Call static method JOptionPane.showMessageDialog() using the method signature that receives two arguments; the first argument should be null, the second argument should be explicit text “Let’s Play Euchre!” 3. Instantiate an instance of class Game calling the no-argument constructor |
| constants |  |
| Constants class | Add the following:   1. A constant for the number of AI players set to the value of 3 2. A constant for the number of cards in a Euchre deck set to the value of 24 3. A constant for the number of cards each player is dealt set to the value of 5 4. A constant for the number of rounds in a hand set to the value of 5 5. An enumeration for the color of cards so it includes: RED and BLACK 6. An enumeration for the suit of cards so it includes: CLUBS, DIAMONDS, HEARTS, SPADES 7. An enumeration for the face value of cards so it includes: NINE, TEN, JACK, QUEEN, KING, ACE |
| core package |  |
| AiPlayer class | Extend class Player.java  Implement methods inherited from class Player:   1. public Card playCard(); 2. public int makeTrump(); |
| Card class | 1. Add member variables (Hint: you will have to explicitly import each enumeration, example: import constants.Constants.Color; ): 2. Data type enumeration Face to represent the face value of the card 3. Data type enumeration Suit to represent the suit of the card 4. Data type enumeration Color to represent the color of the card 5. Generate getters/setters for member variables |
| Deck class | 1. Add member variable    1. Data type Set<Card> to represent the deck of cards 2. Generate getter/setter for member variables |
| Game class | 1. Add member variables    1. Data type enumeration Suit to represent the trump suite of Euchre    2. Data type class Player to represent the lead player for each hand    3. Data type class Player to represent the dealer for each hand    4. Data type class Player to represent which player won the current trick/hand    5. Data type int to represent the current round of the game    6. Data type ArrayList<Team> to represent the two teams of the game    7. Data type class Deck to represent the deck of cards for the game    8. Data class Scanner to get information from the user 2. Generate getters/setters for the member variables 3. Write a custom constructor that    1. Has no parameters    2. Calls method createTeams() which will be defined within this class    3. Calls method outputTeams() which will be defined within this class 4. Write method createTeams() that    1. Has no parameters    2. Has return type void    3. Instantiate the member variable of type ArrayList<Team>    4. Instantiates two instances of class Team       1. one for TeamOne       2. one for TeamTwo       3. set names for each team as appropriate       4. add each instance to the ArrayList<Team> member variable    5. Instantiate the member variable of class Scanner passing “System.in” as the argument to the constructor    6. Using System.out.println() static method, prompt the user for the human player’s name    7. Instantiate an instance of class String set equal to the reference object of class Scanner using its method .next()    8. Instantiate an instance of class HumanPlayer    9. Call method .setName() on the reference object of class HumanPlayer passing the value stored in the variable from step g. above    10. Add the reference object of class HumanPlayer to the instance of class Team representing Team One    11. Write a for loop to generate three AiPlayer instances        1. Generate a unique name for each AiPlayer and call method .setName() passing the unique name as an argument        2. Add one AiPlayer instance to the instance of class Team representing Team One        3. Add the other two AiPlayer instances to the instance of class Team representing Team Two 5. Write method outputTeams() so that    1. It has no parameters    2. Has a return type of void    3. Uses an enhanced for loop to loop through the collection of member variable of type ArrayList<Team>       * 1. Calls method outputTeam() in class Team |
| HumanPlayer class | Extend class Player.java  Implement methods inherited from class Player:   1. public Card playCard(); 2. public void makeTrump(); |
| IPlayer interface | Add method signatures:   1. public Card playCard(); 2. public void makeTrump(); |
| Player class | 1. Declare abstract class Player so that it implements interface IPlayer 2. Add member variables:    1. Data type String to represent player name 3. Generate getters/setters for the member variables 4. Declare abstract the methods inherited from interface IPlayer    1. public abstract Card playCard();    2. public abstract void makeTrump(); |
| Team class | 1. Add member variables    1. Data type ArrayList<Player> to represent a team    2. Data type int to represent the team’s score    3. Data type int to represent the team’s tricks taken for each hand    4. Data type String to represent the team’s name 2. Generate getters/setters for the member variables 3. Write a custom constructor that instantiates the member variable ArrayList<Player> 4. Write method outputTeams() so that    1. It has no parameters    2. Has a return type of void    3. Outputs the current Team’s name    4. Uses an enhanced for loop to loop through the collection of class Team’s member variable ArrayList<Player>       1. Outputs the current Player’s name |
| userinterface package | Create package userinterface |
| Euchre application |  |
| Test Case 1 | Test Case 1 passes |
| Test Case 2 | Test Case 2 passes |
| Test Case 3 | Test Case 3 passes |
|  | Source compiles with no errors |
|  | Source runs with no errors |
|  | Source includes comments |

**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 application** | The console window should look like figure 2 |
| Test case 3 | **Regression testing; Run application** | The JOptionPane.showMessageDialog() method call should look like figure 3 |

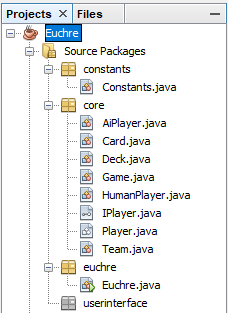


Figure 1 Project View

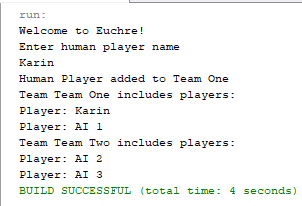


Figure 2 Output in console window

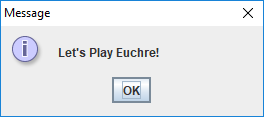


Figure 3 Display from JOptionPane.showMessageDialog() method