HW 4: Simple21

(Deadline as per Canvas)

This assignment deals with the following topics:

- Loops
- Strings
- Classes & Methods

General Idea of the Assignment

This is a simplified version of a common card game, "21". Blackjack is a better-known variant.

In this game, the dealer deals two "cards" to each player in the very first round, one hidden, so that only the player who gets it knows what it is, and one face up, so that everyone can see it. In the following rounds, subsequent cards are added to the visible cards in hand for both players, one card per round, until the user passes. That is to say, each user will get only one hidden hard, which is assigned to them in the first round. (Actually, what the other players see is the total of each other player's visible cards, not the individual cards. Users can know all their own cards and the sum of them.)

There are two players: one human player (the person playing the game) and one computer player. To play, the players take turns requesting cards, trying to get as close to 21 as possible, but not going over 21. A player may pass (ask for no more cards). Once a player has passed, he or she cannot later ask for another card. When all players have passed, the game ends.

The winner is the player who has come closest to 21 without exceeding it. In the case of a tie, or if everyone goes over 21, no one wins.

The game is only played once (so it's actually just one "hand").

The "cards" are the numbers 1 through 10 and they are randomly generated, not drawn from a deck of limited size. The odds of returning a 10 are four times as likely as any other value (because in an actual deck of cards, 10, Jack, Queen, and King all count as 10).

Provided program implementation:
We have provided a *Simple21.java* class file which includes:

Main method

- Creates instance of the Simple21 class and runs the program by calling the run() method
- Do not modify this method - leave as given in the starter code

void run()

- This method runs and controls the overall flow of the program.
- Prints the game instructions by calling printInstructions()
- Gets and sets the user's name dynamically based on user input
- Sets the computer's name to “Computer”
- While the game is running
    - Runs the game by calling play()
    - Asks the user if he/she wants to play the game again by calling askYesOrNo("Play again? (y/n) ")
        - If yes:
            - Asks the user if they want to modify their username by calling askYesOrNo("Modify username? (y/n) ")
            - Runs the game again by calling play()
        - If no:
            - Exits the program

void play(String username, String computerName)

- This method controls the game and logic for the given user and computer.
- Gives each player, in turn, a chance to take a card, until both the user and computer have passed. Prints a message when a player passes. Once a player has passed, that player is not given another chance to take a card.
- First, assign the user two random cards, one hidden, and one visible. Print the user’s status.
- Then, assign the computer two random cards, one hidden, and one visible. Print its status.
- While the game is not over, ask the user/computer if they want to take random cards until both of them pass.
- If both of them passed, print the winner, and the play() method will return.
- Notice that, once a player has passed, that player is not given another chance to take a card. And the user can choose to ask for more cards even though they have more than 21 points.

void printInstructions()

- Prints out instructions for the game.

boolean askYesOrNo(String prompt)

- Displays the given prompt and asks the user for input. If the user's input starts with “y”, returns true. If the user's input starts with “n”, returns false.
- For example, calling askYesOrNo("Do you want to play again? (y/n) ") would display "Do you want to play again? (y/n) ", wait for user input that starts with “y” or “n”, and return true or false accordingly.

To check whether a string starts with specific character(s) (e.g. “y” or “Y”), you can use the *startsWith* method. For example, this checks if “someString” starts with “some” and stores the result in a boolean variable:

```
boolean startsWith = “someString”.startsWith(“some”);
```

- Other requirements:
  - If the user types in “y” or “yyyy”, the method returns true anyway.
  - "Yes" or "No" are case-insensitive. That’s to say, “YES” and “Yes” are equal. They’ll both return true.
  - The method needs to strip leading and trailing whitespace (including “\t”, “\n”) of the user input. The word “   Yes   ” with whitespace will be considered the same as “Yes”.

    You can use the *strip* method to remove all leading and trailing white spaces from a string. For example, this returns the string “   yes\n” as a new string “yes”:

    ```
    boolean strippedString = “    yes\n”.strip();
    ```

  - If the user’s input is invalid, prompt the question again until a valid input.

int nextCard()

- Returns a random "card", represented by an int between 1 and 10, inclusive.
- The "cards" are the numbers 1 through 10 and they are randomly generated, not drawn from a deck of limited size. The odds of returning a 10 are four times as likely as any other value (because in an actual deck of cards, 10, Jack, Queen, and King all count as 10).
- Hint: Think about how to write code that will generate a random number from 1 - 13, but will only return a number from 1 - 10 from this method.

boolean takeAnotherCard(int computerTotalPoints, int userVisibleCard)

- Strategy for computer to take another card or not.
- According to the computer’s own given total points (sum of visible cards + hidden card) and the user's sum of visible cards, you need to design a game strategy for the computer to win the game.
- Returns true if the strategy decides to take another card, false if the computer decides not to take another card.
- Notice that, if the computer decides not to take a card, he will not be able to take the card in the following rounds until the game is restarted. (Same as the user, if the user decides not to take cards, he will be marked as passed and will not take cards until the game is restarted.)

boolean isGameOver(boolean isUserPassed, boolean isComputerPassed)

- Determines if the game is over or not. If the given isUserPassed is set to true, the user has passed. If the given isComputerPassed is set to true, the computer has passed.

- This method returns true if both the user and the computer have passed, and false if either of them has not yet passed.

void printStatus(boolean isUser, String name, int hiddenCard, int visibleCard, int totalPoints)

- In each turn, prints out the current status of the game.
- If isUser is set to true, the given player is the user. In this case, print out the user's given name, his/her hidden card points, visible card points, and total points.
- If isUser is set to false, the given player is the computer. In this case, print out the computer's given name, and his/her visible card points.
- For example, calling *printStatus(true, "Brandon", 4, 15, 19)* would print:
```
Brandon has 4 hidden point(s).
Brandon has 15 visible point(s).
Brandon has 19 total point(s).
```
- As another example, calling *printStatus(false, "Computer", 1, 19, 20)* would print:
```
Computer has 19 visible point(s).
```

void printWinner(String username, int userTotalPoints, String computerName, int computerTotalPoints)

- Determines who won the game and prints the game results in the following format:
    - User's given name and the given user's total points
    - Computer's given name and the given computer's total points
    - The player who won the game and the total number of points he/she won by, or if it's a tie, nobody won.

Program Output

Print out what the program is doing as it goes along. We have provided a *template_behavior.txt* file which shows some sample runs of the program -- yours should provide similar information.

Submission

Your submission should include:

- *Simple21.java* - the source code for your game
    - Make sure this class file is named exactly "*Simple21.java*"

  This file must include a header, in the form of javadoc comments at the top of your class that contains
  (each on its own line):

- Your name
- Your Penn ID
- Statement of work. Either:
    - A list of resources you used and/or people you received help from (including TAs/Instructor)
    - A statement that you worked alone without help

Evaluation

Correctness - 18 pts

Does the game work as expected? Did you follow the directions exactly? Did you implement all the required methods correctly? Does your program properly handle user input? For example, when the user is asked if they want to take another card, and they type "yes", will the program behave as expected? Another example, is the nextCard() method four times as likely to return a 10?

Javadocs/Comments – 5 pts

Does every method have Javadoc comments defined? Do they appropriately describe what the method does? Are all non-trivial lines of code commented? Did you provide the necessary information in the form of Javadoc comments at the top of your class?

Style – 5 pts

The style includes things like variable names, comments, method names, and general readability of your code. Are variables named based on the information they store? Are your user-defined methods named based on what they do?