 [Department of Computer Science](http://www.cs.umd.edu/)

[CMSC132: Fall 2016](http://www.cs.umd.edu/class/fall2016/cmsc132-04XX/)

**Project:** Blackjack

**Due Date:** Thursday Sep 8, 11:00 pm

## Overview

For this project you will write a class named **Blackjack** that implements the game logic associated with a blackjack card game. The Blackjack class implements the **BlackjackEngine** Java interface whose definition you can find in the [project's javadoc documentation](http://docs.google.com/doc/index.html).

**Unlike other class projects, for this project you are allowed to discuss or receive help from your classmates, but you may not exchange any code. Every student must implement their own code. If you are assisted by a student add the name of the student to the top of the Blackjack.java file.**

## Objectives

This project will give you practice using classes, ArrayList and other fundamental Java Constructs.

## Grading

* (55%) Public Tests
* (15%) Release Tests
* **(20%) Secret Tests**
* (10%) Style

## Good Faith Attempt

Remember you need to satisfy a good faith attempt for every project. The good faith attempt for this project is the public tests. The deadline to satisfy this requirement will be posted later on.

## Code Distribution

The project's code distribution is available by [checking out the project](http://www.cs.umd.edu/eclipse/manage.html#check-out) named **132Fall2016Project1**. The code distribution provides you with the following:

* A package called **blackjackGUI -** This package implements the game's Graphical User Interface.  You don't have to modify or add anything to this package.  We have provided the source code in case you are interested to see how it works.  In this package you will find a class named **PlayBlackjack** which is the class you will execute if you want to play the game through the GUI.
* A package named **blackjack** - In this package you will find the shell for a class named **Blackjack** that implements the blackjack game logic.  This is the class you need to implement for this project.  In the blackjack package you will also find the following support classes: **Card**, **CardSuit** **and CardValue**.  You will use these classes during the implementation of the the Blackjack class; you should not modify these classes.
* **PublicTests.java** - This class represents the set of JUnit public tests. The expected results for each test can be found in the text files with a "pub" prefix.  This class can be found in the **tests** package.
* A package named **tests** - Includes the public tests (**PublicTests.java)**  and a shell for a class (StudentTests.java).

## Specifications

### Blackjack Rules

If you have been to Vegas then you are familiar with the rules to play blackjack (also known as 21).  However, what happens in Vegas stays in Vegas, therefore the rules for blackjack in this project are different.  If you want to familiarize yourself with the game, check the online blackjack game available at <http://www.247blackjack.com/>.  **Note that the rules we will use are not exactly the same ones used in this online game.**

In our blackjack version we have a dealer (person who shuffles and distribute cards) and only one player.  The game's objective is for the player to beat the dealer by generating a hand of cards whose value is higher than the dealer's hand without exceeding a total value of 21.  The game starts by the dealer shuffling cards (one or more decks) and dealing two cards to the player and herself/himself (the actual order is described in the [project's javadoc documentation](http://docs.google.com/doc/index.html)).  One of the dealer's cards will be face down.  At this point the player will ask for cards until he/she believes he/she can beat the dealer with the current hand and as long the cards' total value does not exceed 21.  If the hand of cards does not exceed 21, and the player stops requesting cards (what is referred to as "stand"), then the dealer will flip the card that was face down, and proceed to deal cards to himself/herself as long as the cards' value is less than 16 and does not exceed 21.   If a value greater than 21 is generated the player wins.  Otherwise whoever (player or dealer) has the hand with the highest value will win the game.  The following provides additional information about the game:

* The term "bust" refers to the scenario where the player or dealer cards' value exceeds 21.
* The term "blackjack" refers to the scenario where in a hand of cards we have an Ace("1") along with a "10", Jack, Queen or King.
* In our blackjack version the player cannot split a hand of cards.  If you don't know what splitting is disregard this comment.
* The value of cards "2" through "10" correspond to the numeric value associated with the card face value.
* "1" (Ace) could be worth either 1 or 11.
* The Jack ("J"),  Queen("Q")  and King("K") are worth 10 points each.
* The card's suit (i.e., "SPADES", "DIAMONDS", "HEARTS", "CLUBS") has no bearing on a card's value.

## Requirements

* The state of the game is defined by the values: BlackjackEngine.DEALER\_WON, BlackjackEngine.PLAYER\_WON, and BlackjackEngine.GAME\_IN\_PROGRESS.
* Your program should handle more than one deck of cards.
* **When creating a deck of cards, load the suits in the following order: SPADES, DIAMONDS, HEARTS and CLUBS.  At the same time, each suit should be loaded starting with the ACE, followed by cards with a face value of "2" through "10", followed by the "JACK", "QUEEN" and "KING".**
* **When creating more than one deck of cards, you must load one deck at a time using the order specified above.  After all the cards have been loaded, then you can proceed to shuffle them.** For instance, a game with 2 decks of cards is initially created as: Spade Ace, Spade 2, Spade 3,... Spade Jack, Spade Queen, Spade King, Diamond Ace ... Diamond King, Heart Ace ... Heart King, Club Ace ... Club King, Spade Ace ... Spade King, ... Club King
* Cards that are in the deck should remain "face up". (Sorry if this is not intuitive, but that's how we're doing it.)
* When dealing a card from the deck, remove the card at position 0 (don't remove cards from the other end of the deck).
* **You must use ArrayList objects to represent the player and dealer's cards, and the deck.**
* You must use the **shuffle** method of the **java.util.Collections** class to perform the data shuffling.  You must use the shuffle method that takes a list followed by a Random object.  Notice that the list parameter is defined as List<?>.  You can ignore that definition.  The method will work with ArrayList objects.
* A player has an account with an initial value of 200 chips.
* The default initial bet is 5 chips.
* When a new game is dealt the bet amount is deducted from the account.
* When a player wins he receives twice the bet amount.
* When a player draws he receives the original bet amount.
* Immediately after checking out the code distribution, make sure you can submit your project, even if you have not implemented the project.  **Do not wait until the day the project is due to submit your project for the first time.**
* Verify that your project passes the submit server tests ([https://submit.cs.umd.edu/](https://submit.cs.umd.edu/fall2016))
* Although we will not grade them, you are welcome and encouraged to provide your own tests in the in the file **StudentTests.java** provided with the code distribution.
* Keep in mind that for this project you have three release tokens in the submit server. Information about the test types for CS course can be found at [Test Types](http://www.cs.umd.edu/~nelson/classes/utilities/TestTypes.html).
* Do not define any supporting classes.
* **IMPORTANT** → If you have a problem with your code and need assistance during office hours, you need to have a student test(s) that illustrates the problem you are experiencing.
* See [StudentTests.html](http://www.cs.umd.edu/class/fall2016/cmsc132-04XX/content/resources/StudentTests.html) for information regarding the implementation of student tests for this project.
* See [Style Guidelines](http://www.cs.umd.edu/class/fall2016/cmsc132-04XX/content/resources/StyleGuidelines.html) for information regarding style.
* We cannot provide any information regarding release nor secret tests. Once your project has been graded, you can see a TA if you would like to find out why you failed a release or secret test.

## Sample Run

[Game Video](http://docs.google.com/gamevideo.mp4) illustrates the functionality associated with the application.

## Suggestions on How To Start the Project

* Study the public tests before starting implementing your project.
* Notice that you can implement this project without using the GUI at all.

## Submission

Submit your project using the "Submit Project" option (available by right clicking on the project folder).

## Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the [Office of Student Conduct](http://www.jpo.umd.edu/).

[Web Accessibility](https://www.umd.edu/web-accessibility)