605.201 Mini-Project 2:

Please do the following to complete this assignment.

Purpose:

The purpose of this project is to provide non-trivial practice in the use of Java programming constructs discussed from the beginning of the course through Module 08 and have a bit of fun doing it.

Resources Needed:

You will need a computer system with Java 7 or greater SE edition run-time and JDK. You may optionally use a Java IDE for example NetBeans, Eclipse, etc. However, application builders are not allowed.

Submitted Files:

Application Design:

This word-processed document is required to be written in APA style minus the Abstract section. The file formats accepted will be either .docx or .pdf. The length of the document should be between 1 and 2 pages, not including the UML class diagram. The following subjects should be discussed in this order:

1. Include a UML class diagram of your solution. Be sure to show the important associations between classes. The diagram should be pasted into a .docx or a .pdf file.
2. What alternative design approaches were considered and why were they rejected?

Source file:

Each public class must be contained in a separate Java source file. Only one source file will have a main() method and this source will be named BlackjackGameSimulator.java. Other source/class names are up to you following the guidelines specified so far in the course. The format of the Java source must meet the general Java coding style guidelines discussed so far during the course. Pay special attention to naming guidelines, use of appropriate variable names and types, variable scope (public, private, protected, etc.), indentation, and comments. Classes and methods should be commented with JavaDoc-style comments (see below). Please use course office hours or contact the instructor directly if there are any coding style questions. JavaDocs: Sources should be commented using JavaDoc-style comments for classes and methods. Each class should have a short comment on what it represents and use the @author annotation. Methods should have a short (usually 1 short sentence) description of what the results are of calling it. Parameters and returns should be documented with the @param and @return annotations respectively with a short comment on each. JavaDocs must be generated against every project Java source file. They should be generated with a - private option (to document all protection-level classes) and a –d [dir] option to place the resulting files in a javadocs directory/folder at the same level as your source files. See the JavaDocs demonstration for more details.

Submit file:

The submit file is to be a Zip file containing your design and analysis document, your Java sources, and your javadocs directory/folder. It should also contain a screen capture of your program executing. Any appropriate file name for this Zip file is acceptable. If you know how to create a standard Java JAR file, this is also acceptable for your source code. However, make sure you include the source code in your JAR file.

Collaboration:

It is encouraged to discuss technical or small design parts of this project with your fellow students. However, the resulting design and implementation must be your own. When in doubt, ask during office hours or contact your instructor.

Program Specification:

This project involves writing a program to simulate a blackjack card game. You will use a simple console-based user interface to implement this game.

A simple blackjack card game consists of a player and a dealer. A player is provided with a sum of money with which to play. A player can place a bet between $0 and the amount of money the player has. A player is dealt cards, called a hand. Each card in the hand has a point value. The objective of the game is to get as close to 21 points as possible without exceeding 21 points. A player that goes over is out of the game. The dealer deals cards to itself and a player. The dealer must play by slightly different rules than a player, and the dealer does not place bets. A game proceeds as follows: A player is dealt two cards face up. If the point total is exactly 21 the player wins immediately. If the total is not 21, the dealer is dealt two cards, one face up and one face down. A player then determines whether to ask the dealer for another card (called a "hit") or to "stay" with his/her current hand. A player may ask for several "hits." When a player decides to "stay" the dealer begins to play. If the dealer has 21 it immediately wins the game. Otherwise, the dealer must take "hits" until the total points in its hand is 17 or over, at which point the dealer must "stay." If the dealer goes over 21 while taking "hits" the game is over and the player wins. If the dealer's points total exactly 21, the dealer wins immediately. When the dealer and player have finished playing their hands, the one with the highest point total is the winner. Play is repeated until the player decides to quit or runs out of money to bet.

You must use an object-oriented solution for implementing this game.

Assessment:

Part	**70%**	**80%**	**90%**	**100%**	**% of Grade**
Design Document	All but one subject addressed with relevant, information. Few minor typographical issues. Document is close to assigned length. Many UML errors, and design is not object-oriented.	All assigned subjects address with mostly relevant information. Nicely formatted document. Document is close to assigned length. Design reflects poor application of object-oriented concepts. Few UML errors	All assigned subjects address with accurate and relevant. Nicely formatted document. Document is within assigned length. Fair to good object-oriented design concepts applied. No serious UML errors.	All assigned subjects address with accurate, relevant, and insightful information. Very nicely formatted. Document is within assigned length. Good to excellent object-oriented design concepts applied. No UML errors.	25%
Functionality	Majority of required function parts work as indicted in the assignment text. One major or 3	Most required function parts work as indicted in the assignment text above and	Nearly all required function parts work as indicted in the assignment text above and submitted documentation. One	All required function parts work as indicted in the assignment text above and submitted	35%
	minor defects. All major functionality at least partially working (example change provided but not correct). Design document does not fully reflect functionality.	submitted documentation. One major or 3 minor defects. All major functionality at least partially working ((example change provided but not correct).	to two minor defects.	documentation.	

Code	Majority of the code conforms to coding standards as explained and demonstrated so far in the course (ex. method design, naming, formatting, etc.). Five to six minor coding standard violations. Some useful comments. Some JavaDocs commenting. Code compiles with multiple warnings or fails to compile with difficult to diagnose error.	Most of the code conforms to coding standards as explained and demonstrated so far in the course (ex. method design, naming, formatting, etc.). Three to four minor coding standard violations. Mostly useful comments. Public class JavaDocs complete. Code compiles with one to two warnings.	Almost all code conforms to coding standards as explained and demonstrated so far in the course (ex. method design, naming, formatting, etc.). One to two minor coding standard violations. Appropriate level of useful comments. Public class JavaDocs complete. Code compiles. Code compiles with no errors or warnings.	All code conforms to coding standards as explained and demonstrated so far in the course (ex. method design, naming, formatting, etc.). Appropriate level of useful comments. Complete JavaDocs as specified. Code compiles with no errors or warnings.	30%
Submit package	More than one file submitted in incorrect format. Files not enclosed in the specified compressed file.	All but one file submitted in correct format. Files not enclosed in the specified compressed file.	All file submitted in correct format but not in the specified compressed file.	All file submitted in correct file formats and compressed as specified.	10%

If you have any questions about the specification of this project, contact your instructor *before* the project is due.