Homework #2: Object-oriented Analysis and Design

Due Thursday, September 18th at 11:59 p.m.

This assignment consists of two parts. In the first part you will analyze a problem domain (the card game Blackjack) to produce a domain model, an object model, several interaction diagrams, and the behavioral contracts for one domain scenario. In the second part of this assignment you will practice developing software based on artifacts of the design process by implementing software for a coffee shop from a UML class diagram of the software's object model.

Your goals for this assignment are to:

- Successfully use object-oriented analysis. Specifically you should be able to analyze a problem domain based on a textual description, develop a domain model for that domain, and develop an object model and interaction diagrams based on the domain model and textual descriptions of use-case scenarios.
- Demonstrate mastery of the distinctions between a domain model and object model.
- Correctly use basic UML notation to create UML class diagrams and interaction (either sequence or communication) diagrams.
- Identify and describe behavioral contracts for interactions with a system.
- Interpret an object model (as a UML class diagram) and accurately represent its design in Java code.
- Demonstrate mastery of the learning goals from previous homework assignments, such as using good Java coding practices and style.

This document continues by describing the object-oriented design task (part 1) and then the implementation task (part 2).

Part 1: Object-oriented analysis and design

In this part of the assignment you will analyze the card game Blackjack to produce a domain model, an object model, interaction diagrams for several use-case scenarios, and the behavioral contracts for one of those scenarios. Specifically, you will analyze a simple variant of the game where a single player plays against an automated dealer:

> The objective of Blackjack is to obtain a score higher than the dealer. In each round the player and the dealer are initially dealt 2 cards, with one of the dealer's cards dealt face down. Each numerical card (2-10) is worth its face value, aces are worth either 1 or 11 (it is the player's choice), and all face cards (jack, queen, and king) are worth 10.
>
> At the beginning of each round the player must place a bet; this amount will not change for the rest of the round. After the bet, the player repeatedly decides whether to *hit* (be dealt a new card, up to 5 cards in the player's hand) or *stand* (stop being dealt cards). If the player scores more than 21 points, the player loses his bet and the round.
>
> After the player stands, the dealer exposes his face-down card and must hit (be dealt cards) until the value of his cards is more than 16. If the player's score is more than the dealer's final score (but less than or equal to 21) then the player wins an amount equal to his bet. The player also wins if he is dealt 5 cards worth 21 or fewer points. Otherwise, the player loses his bet.

See the Wikipedia's Blackjack page for more information. You should not model advanced game details, such as doubles and splits, that are not mentioned above.

Object-oriented analysis

Start building a vocabulary by creating a domain model for this Blackjack variant. Document all relevant concepts from the problem domain and include important attributes and associations. Your domain model should be visualized by a UML class diagram.

Object-oriented design: modeling use-case scenarios

To start designing a solution create an interaction diagram (either a UML sequence diagram or a UML communication diagram) for each of the three scenarios below, and also explicitly describe the behavioral contracts for the third scenario. Consider how the game flows from an initial action (e.g., a user clicking a button, or part of the system requesting some information) to the individual steps following that action.

The three scenarios are:

1. Determine the number of points of the player's current hand.
2. Given a bet, initialize a new round. This includes updating scores and dealing cards.
3. The player takes the 'hit' action during the game. This includes dealing additional cards, checking whether the game is over, possibly updating scores, etc. For this scenario also explicitly describe the set of behavioral contracts for the 'hit' operation in addition to its interaction diagram.

Object-oriented design: creating an object model

Based on your domain model and interaction diagrams, create an object model for this Blackjack variant. Your object model should describe the software classes of the system and use the design principles (such as information hiding and cohesion) discussed in class. If you encounter nontrivial design decisions, carefully consider those decisions and explore alternative solutions. Your object model should be visualized as a UML class diagram, and the methods and relationships in your object model should correspond to the methods and interactions in your interaction diagrams.

Part 2: Implementing software from an object model

For this part of the assignment you will implement software for a coffee shop to print recipes and calculate the cost of drinks. Below is a list of requirements, and in an appendix we've included an object model (as a UML class diagram) representing a possible system. The provided design was developed to make it easy to add new features such as ingredients, beverages, recipes, or sizes to the coffee shop application. You should implement the system according to the provided object model.

When implementing the system you may change the names and signatures of methods and you may add methods as necessary. The high-level features of your implementation (such as classes and relationships between classes) should match the object model's design. We have provided a small amount of code (including a main method) to help you begin your implementation. See our `README.md` file for some details.

Requirements for the coffee shop

1. The coffee shop offers both Coffee and Tea beverages.
   a. The basic coffee beverages are House Blend, Espresso, and Decaf.
   b. The basic tea beverages are Green Tea, Red Tea, and White Tea.
2. Each beverage comes in three sizes: small, medium, and large.
3. The software takes input in the following format:

   ```
   <beverage name> <size> [<ingredient> <ingredient> ...]
   ```

   For example, a large green tea with milk and ginger would entered in as:

   ```
   greentea large milk ginger
   ```

4. The software should calculate the cost of each order according to the size and type of the beverage, plus an additional charge for ingredients:

   a. **Beverage**

      i. Espresso, Green Tea, White Tea: 100 cents

      ii. House Blend, Red Tea: 80 cents

      iii. Decaf: 50 cents

   b. **Additional Ingredients**

      i. Chocolate, Milk, Whipped Cream: 30 cents

      ii. Jasmine: 50 cents

      iii. Ginger: 60 cents

   c. **Sizes**

      i. Coffee

      - Small: 40 cents
      - Medium: 70 cents
      - Large: 100 cents

      ii. Tea

      - Small: 20 cents
      - Medium: 50 cents
      - Large: 70 cents

5. Every beverage will be prepared using one of the following recipes:

   - Coffee recipe:

     ```
     add coffee
     process ingredients
     add ingredients
     ```

   - Tea recipe:

     ```
     add water
     add tea bags
     add all ingredients
     ```

   - Coffee with milk recipe:

     ```
     steam milk
     add coffee
     mix steamed milk and coffee
     process non-milk ingredients
     add all ingredients
     ```

- Tea with milk recipe:

```
add water
add tea bags
add non-milk ingredients
add milk
```

6. The output format of the program should be:

   The total cost of your order is: *cost*
   The beverage is prepared as follows:
   recipe steps

   where *cost* is the total cost of the beverage and the *recipe steps* are the steps of the recipe as described above.

Turning in your work

For the first part (object-oriented analysis and design) of the assignment, turn in the following files:

- `domain.pdf`: Your domain model for Blackjack.
- `interaction_1.pdf`: Your response to our scenario about getting the number of points in the player's current hand.
- `interaction_2.pdf`: Your response to our scenario about initializing a new round.
- `interaction_3.pdf`: Your response to our scenario about the 'hit' action.
- `behavioral_contracts.pdf`: A textual description of the system's behavioral contracts for the 'hit' action.
- `object.pdf`: Your object model for Blackjack.

All documents may contain text and figures.

For the second part (implementing software from an object model) of the assignment, turn in all source code as an Eclipse project in your homework directory. Your project should be configured to, and successfully, build on Travis-CI even though this assignment has no testing requirements.

Evaluation

Overall this homework is worth 100 points. To earn full credit, we expect:

- A domain model that describes the vocabulary of the problem and accurately models concepts and relationships in the problem space, not the solution space, of the design.
- An object model and interaction diagrams that describe the key components of your design, following the design principles from class. Your interaction diagrams should clearly describe how the scenarios could be implemented, and the methods and relationships in your interaction diagrams should correspond to the methods and relationships in your object model.
- Behavioral contracts for the third ('hit') scenario that accurately describe the requirements for that scenario.
- An implementation of the coffee shop that closely matches the provided object model and correctly builds using Ant on Travis-CI.

We will grade your work approximately as follows:

- Design documents demonstrating an understanding of the design process: 10 points
- A domain model, object model, and interaction diagrams that accurately model Blackjack and follow design principles: 45 points
- Clear design documents using correct notation: 5 points
- Accurate behavioral contracts for the third use-case scenario: 5 points
- Faithful implementation of the coffee shop software: 25 points
- Documentation, style, and other learning goals: 10 points

Appendix: The coffee shop's object model