Northeastern University
CS5500 – Managing Software Development
Fall 2018 Professors Jump, Katz, and Weintraub

Homework 3: Behavioral Design Patterns

The purpose of this homework is to give you experience with behavioral design patterns.

Your work will be assessed for:

1. Whether it works for your tests.
2. Whether you have defined a robust test set.
3. Design efficiency, efficacy, and robustness.
4. Code style (documentation and code style). You should create files in such a way so that someone else can use, test, or extend your files without you being present.

In all work, you should run `sonarlint` and must eliminate all major issues.

REQUIREMENT: You must build your project using Maven so we can use Maven as part of grading your project.

REQUIREMENT: You must use `junit4` or `junit5`. No earlier versions are accepted.

REQUIREMENT: You must use Java version 1.7 or later. No earlier versions are accepted.

REQUIREMENT: You must work by yourself; your work must be original which means you may not use any internet resources other than Java language/library documents, junit documents, and the like. You may not receive or find any algorithmic or data structure help except from the teaching team or materials they provide. All violations of these rules are academic misconduct and will be treated as described by the syllabus.

Turn in your answers (maven projects) by pushing them to your course repository in the homework 3 directory.

Problem 1 - Iterators

(30 points)
Extend the `Hand` interface you implemented in the previous homework with an interface named `newHand` whose implementation involves a bespoke (home built) iterator. Specifically, `newHand` extends `Hand` by adding or overriding:

1. `Boolean hasCard(Card cardToFind)` // overriding the existing method
2. `int occurrencesInHand(Card cardToFind)` // New!
3. `int occurencesInHand(Rank rankToFind)` // New!

Implement and test your own iterator. You must define an iterator for a hand. You may not use a standard iterator. You must also provide an adequate `junit` test suit for your solution.

As a hint on the desired functionality, all three new methods are variants of what `ArrayList.contains(Object o)` does.[1] You may not use this function nor may you use any built-in iterator. You must construct your solution from scratch.

Do this as a separate project. Because you are using `Maven`, when you create the project please use the following conventions:

1. Under groupID, use `edu.northeastern.ccs.cs5500`.
2. For artifact id, use `homework3`.

Bonus
(15 points)
Implement a two player version of *Go Fish* using `newHand` and the factory you built in homework 2. You must play the game using your iterator where appropriate. While you can play *Go Fish* with any type of deck `standard` is the default `Deck` type. Do this as a separate project. See the appendix for the rules of the game.
Do not create a fancy graphical interface for the game. We will not be looking at any GUI's.

[1] See `https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#contains(java.lang.Object)`

Problem 2 - Adapters

(30 points)

Because of the success of your work in part , a Chinese conglomerate has decided form a partnership. A requirement is that your code must use its `diErBao` package which contains two classes, `Ka` (card) and `Shou` (hand). Your task is to adapt the system you built for part and use these classes instead of your own. The java files are provided off the web site. You may not alter these files in any way. Everything you've done so far must continue to work as before. That is, all the tests you did for homework 2 (either part 1 or part 2 - the factory version) and for problem 1 on this homework related to Hand or Card must pass using your adaptation of the `diErBo` classes.

Do this as a separate project. Because you are using `Maven`, when you create the project please use the following conventions:

1. Under groupID, use `edu.northeastern.ccs.cs5500`.
2. For artifact id, use `homework4`.

Problem 3 - State

(40 points)

Now let's try a complicated game.

Let's play a more complicated game, blackjack aka 21. From Wikipedia:

> *Blackjack, also known as twenty-one, is a comparing card game between usually several players and a dealer, where each player in turn competes against the dealer, but players do not play against each other. It is played with one or more decks of 52 cards. [our* `standard` *deck] The objective of the game is to beat the dealer in one of the following ways:*
>
> - *Get 21 points on the player's first two cards (called a "blackjack" or "natural"), without a dealer blackjack;*
> - *Reach a final score higher than the dealer without exceeding 21; or*
> - *Let the dealer draw additional cards until their hand exceeds 21.*

Playing a Hand

Each player is dealt two cards. The dealer is also dealt two cards, the first one is up (exposed to all players) and one down hidden. The value of cards two through ten is their pip value (2 through 10). Face cards (Jack, Queen, and King) are all worth ten. Aces can be worth one or eleven. [Prof. note: the player decides whether an ace counts as one or 11. You do not finalize this decision until you decide your hand is "done."] A hand's value is the sum of the card values. Players are allowed to draw additional cards [Prof note: from the deck] to improve their hands. A hand with an ace valued as 11 is called "soft", meaning that the hand will not bust by taking an additional card; the value of the ace will become one to prevent the hand from exceeding 21. Otherwise, the hand is "hard".
Once all the players have completed their hands, it is the dealer's turn. The dealer hand will not be completed if all players have either busted or received Blackjacks. The dealer then reveals the hidden card and must hit [take a card from the deck] until the cards total 17 or more points. (At most tables the dealer also hits on a "soft" 17, i.e. a hand containing an ace and one or more other cards totaling six.) Players win by:

1. not busting [your hand <= 21] and having a total higher than the dealer,
2. not busting and having the dealer bust,
3. getting a blackjack without the dealer getting a blackjack.

If the player and dealer have the same total (not counting blackjacks), this is called a "push", and the player typically does not win or lose money on that hand. Otherwise, the dealer wins.
For now for the blackjack players among us, we will ignore doubling down. We will do splitting, which if you draw a pair, you can convert each card into a separate hand and each hand gets an additional card. If you get another pair, you can split again. Since we're students, we'll play at the $1 table, meaning if you win a hand, you get $1 if you beat the dealer and you lose $1 if you lose the hand. Splits count as two separate hands.

If only one 52 card deck is used in the game, if the deck runs out of cards before the dealer's hand is complete, the hand is tossed (no winners, no losers) and a new deck is used. If a larger deck is used (say a deck composed of six or seven decks which is typical at most casinos), a random spot in the deck (usually in the range of halfway to two-thirds toward back of the deck) is chosen and marked. When the next card

to be played hits that point, a new large deck is shuffled, a random spot is selected, and play continues from there.

The assignment

Design and implement a game of blackjack with one to five players using the Deck and Hand solutions you built in homework 1 part 2. Similarly, you should reuse the iterator from problem 1, should you need an iterator over a hand.

In your game, each player will play a hand. The deck being used will be built from six (6) standard 52 card decks.

Using the State pattern, implement the basic strategy given in Figure 1 (or you can look it up at `https://en.wikipedia.org/wiki/Blackjack` under Blackjack Strategy). Define a `junit` test suit to confirm your work.

Extra Credit (10 points): implement a card counting strategy. The bonus points is valid *only if* your solution to the standard blackjack strategy is correct and working.

Because you are using `Maven`, when you create the project please use the following conventions:

1. Under groupID, use `edu.northeastern.ccs.cs5500`.
2. For artifact id, use `homework5`.

Player hand	Dealer's face-up card									
	2	3	4	5	6	7	8	9	10	A
Hard totals (excluding pairs)										
17–20	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	SU	SU	SU
15	S	S	S	S	S	H	H	H	SU	H
13–14	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh
10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
5–8	H	H	H	H	H	H	H	H	H	H
Soft totals										
	2	3	4	5	6	7	8	9	10	A
A,9	S	S	S	S	S	S	S	S	S	S
A,8	S	S	S	S	Ds	S	S	S	S	S
A,7	Ds	Ds	Ds	Ds	Ds	S	S	H	H	H
A,6	H	Dh	Dh	Dh	Dh	H	H	H	H	H
A,4–A,5	H	H	Dh	Dh	Dh	H	H	H	H	H
A,2–A,3	H	H	H	Dh	Dh	H	H	H	H	H
Pairs										
	2	3	4	5	6	7	8	9	10	A
A,A	SP	SP	SP	SP	SP	SP	SP	SP	SP	SP
10,10	S	S	S	S	S	S	S	S	S	S
9,9	SP	SP	SP	SP	SP	S	SP	SP	S	S
8,8	SP	SP	SP	SP	SP	SP	SP	SP	SP	SP
7,7	SP	SP	SP	SP	SP	SP	H	H	H	H
6,6	SP	SP	SP	SP	SP	H	H	H	H	H
5,5	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
4,4	H	H	H	SP	SP	H	H	H	H	H
2,2–3,3	SP	SP	SP	SP	SP	SP	H	H	H	H

Key:

S = Stand
H = Hit
Dh = Double (if not allowed, then hit)
Ds = Double (if not allowed, then stand)
SP = Split
SU = Surrender (if not allowed, then hit)

Figure 1: Basic Strategy for Blackjack

APPENDIX: How to play Go Fish

A popular game with little kids and doesn't require much thinking is *Go Fish*. According to Bicycle[2],

> *The goal is to win the most "books" of cards. A book is a four of a kind... (meaning four of the same rank)*
>
> *The cards rank from ace (high) to two (low). The suits are not important, only the card numbers are relevant, such as two 3s, two 10s, and so on.*
>
> *Any player deals one card face up to each player. The player with the lowest card is the dealer. The dealer shuffles the cards, and the player on his right cuts them. The dealer completes the cut and deals the cards clockwise one at a time, face down, beginning with the player to his left. If two or three people are playing, each player receives seven cards. If four or five people are playing, each receives five cards. The remainder of the pack is placed face down on the table to form the stock.*
>
> *The player to the left of the dealer looks directly at any opponent and says, for example, "Give me your kings," usually addressing the opponent by name and specifying the rank he wants, from ace down to two. The player who is "fishing "must have at least one card of the rank he asked for in his hand. The player who is addressed must hand over all the cards requested. If he has none, he says, "Go fish!" and the player who made the request draws the top card of the stock and places it in his hand.*
>
> *If a player gets one or more cards of the named rank he asked for, he is entitled to ask the same or another player for a card. He can ask for the same card or a different one. So long as he succeeds in getting cards (makes a catch), his turn continues. When a player makes a catch, he must reveal the card so that the catch is verified. If a player gets the fourth card of a book, he shows all four cards, places them on the table face up in front of him, and plays again.*
>
> *If the player goes fishing without "making a catch" (does not receive a card he asked for), the turn passes to his left.*
>
> *The game ends when all thirteen books have been won. The winner is the player with the most books. During the game, if a player is left without cards, he may (when it's his turn to play), draw from the stock and then ask for cards of that rank. If there are no cards left in the stock, he is out of the game.*

Key to playing the game of *Go Fish* is being able to look through a hand to identify the presence of a particular rank, which means you should use newHand methods extensively.

[2] `https://www.bicyclecards.com/how-to-play/go-fish/`