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

Grading for Homework 3: Behavioral Design Patterns

CCIS-ID:	aflores
Repo Name:	https://github.ccs.neu.edu/cs5500/student-49-F18
Commit-ID:	a19b09a
Repo Commit Date (UTC)	Oct 29, 2018, 1:41 AM EDT
Graded By:	Praveen
Total Score:	72

1 Iterators

1. **QUESTION:** Did the three methods use an iterator object in any form?

   Yes, used an iterator object

---

2. **QUESTION:** Did the code implement a from scratch iterator or did it use an external resource (e.g. java.util interface or implementation)?

   Implemented from scratch

---

3. **QUESTION:** Does it include at least these two methods: hasNext() and getNext()?

   Yes include at least these two methods

---

4. **QUESTION:** Did it include anything like remove (element)?

   Yes, included

---

5. **QUESTION:** Did the code compile without errors ?

   Yes, compiles without errors

---

6. **QUESTION:** Does the code have smells other than complaints about package regex'es?

   Minus Points: 1

---

7. **QUESTION:** Does the code: Extend Hand by creating NewHand? Not having these specific new methods. Not overriding hasCard(Card)?

   Yes

---

8. **QUESTION:** Any signs of sloppiness? (e.g. TODO comments, obsolete code that was commented out, duplicated code, etc.)

   Minus Points: 0

---

9. **QUESTION:** Was Maven used? Did Maven run without error? No missing dependencies allowed.

   Minus Points: 0

---

10. **QUESTION:** Does the code pass the previous tests, including negative path, on their Hand Impl?

    Minus Points: 0

---

11. **QUESTION:** Does the code include positive path tests on occurencesInHand() for both rank and card, namely:1.On either type of input, where there's a case for 0 occurrences, 1, and more than 1 occurrence?

    Yes

---

2 BONUS

12. **QUESTION:** Does the code play the game of Go Fish as an at least two-player game where at least one side is automated?

    +1pts Attempted with Good Effort

---

13. **QUESTION:** Did the code use NewHand from the core problem AND that code has an appropriate iterator?

    No

---

14. **QUESTION:** Does the code have smells other than complaints about package regex'es. Any signs of sloppiness? (e.g. TODO comments, obsolete code that was commented out, duplicated code, etc.)

    Minus Points: Satisfactory

---

15. **QUESTION:** Was Maven used? Did Maven run without error? Is this a discrete project?

    Minus Points: +2pts Yes

---

16. **QUESTION:** Are there any tests?

    Satisfactory

---

3 Adapt a Card

17. **QUESTION:** Did the code use a class adapter model or an object adapter model to obscure the adaptee?

    Yes

---

18. **QUESTION:** Did the adapter provide all the function of Hand: 1.accept(Card) 2. Boolean hasCard(Card) 3.Card pullCard() 4.List¡Card¿ showCards()5. Sort(String) 6. Shuffle() and Card: 1. Rank getRank() 2. Suit getSuit().?

    Minus Points: 0

---

19. **QUESTION:** Throwing an exception when asking for a sort by suit or by both?

    Throwing no Exception

---

20. **QUESTION:** Does it expose diErBao details to the client (not counting class based adapter offering the interface on the Chinese side)?

    Minus Points: 0

---

21. **QUESTION:** Implementing methods themselves what they could have gotten from the adaptee. What counts is the implementation. Also, there should be no changes to the adaptee.?

    Minus Points: 0

---

22. **QUESTION:** Did the code compile without errors?

    No Errors

---

23. **QUESTION:** Does the code have smells other than complaints about package regex'es?

    Minus Points: 1

---

24. **QUESTION:** Does the code pass the previous tests, including negative path, on their Hand Impl and on their Card (or Rank/Suit) Impl?

    Minus Points: 0

---

25. **QUESTION:** Was Maven used? Did Maven run without error? Is this a discrete project?

Minus Points: 0

---

4 Play BlackJack

26. **QUESTION:** Does the code recognize the hand (or player) as the object that has state?

    Satisfactory

---

27. **QUESTION:** Does the code make use of a state pattern to organize decisions and transitions?

    Minus Points: 9

---

28. **QUESTION:** Does the code use a singleton pattern to minimize state instances?

    Minus Points: 2

---

29. **QUESTION:** Does the code require counting the final value of the hand by re-examining the cards in the hand rather than looking up the state?

    -2pts: Yes

---

30. **QUESTION:** Did the code compile without errors?

    Yes, no errors

---

31. **QUESTION:** Does the code have smells other than complaints about package regex'es?

    Minus Points: 1

---

32. **QUESTION:** Is changeState() method tested across all possible state changes?

    Minus Points: 5

---

33. **QUESTION:** For each state, for each possible draw, is the correct action verified?

    Minus Points: 5

---

34. **QUESTION:** Is the creation of every possible state tested? The pattern uses a static construction method?

    -2pts Not fully tested

---

35. **QUESTION:** Is there a test for a round – meaning the play of at least one player, one dealer, and an assessment of winning?

    No

---

36. **QUESTION:** Is the dealer's behavior tested? Does the dealer hit until 17 – hard or soft?

    -2pts Both

---

37. **QUESTION:** Was Maven used? Did Maven run without error? Is this a discrete project?

    Minus Points: 1

---

5 General Comments

38. **QUESTION:** Comments:

    Please set the Java Version in your pom.xml, else the build will fail in a PC other than yours (which does not have the same environment).

---

39. **QUESTION:** Missing all of part 1. Iterators?

    No

---

40. **QUESTION:** Missing all of part 2. Adapt a Card?

    No

---

41. **QUESTION:** Missing all of part 3. BlackJack aka State?

    No
