**Reading Assignment 11 – Chapter 9 Questions**

|  |  |
| --- | --- |
| Name: |  |

Instructions:

* Read pages 315-352 of your text.
* Type answers to the questions below (don’t remove questions).
* Submit work on BlazeView.
1. Consider the application discussed on pages 318-322. State four reasons that this code is not good.
2. Consider the application discussed on pages 318-322. Which class will be hard to maintain? Why?
3. Consider the application as described through page 325, what is *varying* (thus, we want to consider encapsulating)? What is the solution?
4. (Omit) Describe in detail how Java’s Iterable interface works. How does Java’s Iterable interface fit into the JCF?
5. (Omit) What is one danger associated with the remove() method when using multiple threads that have iterators over the same collection of objects?
6. Define these: External Iterator, Internal Iterator, Polymorphic iteration.
7. How does use of the iterator pattern adhere to the design principle: A class should have only one reason to change?
8. (Omit) Define cohesion in your own words.
9. (Omit)What does it mean to say that HashMap (Hashtable) indirectly supports the Iterator interface? What about the Keys-how are they accessed? Can you critique this approach?
10. (Omit) What is the problem with the application as described through page 351? How do they fix the problem?
11. (Omit) On page 332, it says that we will get a lot of leverage out of using the Java *Iterator* interface. What is this leverage?
12. What is wrong (missing) from the UML on page 358 (or 359)?
13. (Omit) On page 360, why didn’t they make the MenuComponent class an interface?
14. (Omit) Why can’t the print method be implemented in the abstract class MenuComponent?
15. Consider the implementation through page 366. What if the DinerMenu and CafeMenu need to both use the DessertMenu? How would you handle this?
16. How does the Composite pattern violate the Single Responsibility design principle in the context of the problem in the text? Describe the design tradeoff between *transparency* and *safety* (see page 367).
17. (Omit)Why is it OK to have the methods: add, remove, getChild available in the leaf nodes?
18. What is a Composite Iterator?
19. (Omit)Why is the Composite Iterator more complex than the print method?
20. Why is the Null Iterator useful? Argue that it provides transparency and safety.
21. (Omit)Explain how the stack is used in the CompositeIterator?