CSE461 – Software Engineering

Unit 1 Questions (Due February 3rd, 5:00 pm)

Please answer the following questions about this unit's patterns. Diagrams are highly recommended when answering these questions (please use appropriate UML notation).

Create a *word/pdf* document named **Unit1-Questions.doc**, containing the questions and your team's answers. Deposit this file into the **U1 Questions** drop box in the Courses portal. Only one person per group should submit the file. However all the team members names should be listed on the first page of the submission.

1. As a lead developer in an organization, you are in-charge of training the new comers about Software design. One of the new comers asks the following question: "If we are going to refactor the design anyway, why should we worry about coming up an initial design"? How would you answer this question?

2. Does pair programming reduce the need for refactoring? Support your argument with appropriate evidence.

3. In "No Silver Bullet -- Essence and Accidents of Software Engineering" by Fred Brooks, the author claims "there is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement within a decade in productivity, in reliability, in simplicity". List arguments to support/contradict his claim.

4. Explain the statement "An adapter is a glue or wrapper class."

5. The C++ implementation of class Adapter specifies the use of multiple inheritance. Using a diagram show how this could be implemented in Java? Be specific about the Java features that you are using.

6. One method of classifying iterators does so along two dimensions. The first indicates the location of control of the iteration (internal to the iterator or external client control), and the second which indicates the location of the definition of the iteration logic (embedded as part of the collection objects or in objects separate from the collections). Considering each dimension separately, what are the positive and/or negative aspects of iterators of each type?

7. What language constructs would you use to give iterators privileged access in Java and in C++? How will your answer depend on the classifications for iterators given above?

8. Iteration over a recursive data structure such as a binary tree can be tricky using an external iterator: Why? How can you accomplish this using an external iterator?

9. Draw sequence diagrams for the registration and update cycles of an Observer pattern implemented using appropriate Java classes. Be sure to label the object with the Java class and its role in the Observer pattern.

10. You have seen that it is very common to use the Observer pattern within GUI frameworks, such as Java's Swing framework. Why do you need to be concerned about how long it takes to process a notification by a GUI element? Should this be the application designer's concern or be dealt with by the framework itself? How can the concern be eliminated?

11. Argue both for and against including the functions to handle the Composite's children in the Component interface. What are the implications for implementation of the pattern and on the intention of the pattern?

12. How can a Builder enforce semantic constraints, i.e. it is only valid to add some certain parts to the product when it is within another part, certain parts must be added to the product before/after other parts? Suggest methods that a Builder can use to handle a violation of semantic constraints.

13. A new part is now able to be built into your product. This new part will be handled by all new ConcreteBuilders added to the system. What must change to accommodate the new parts? How can you let the current ConcreteBuilders ignore the part without rewriting any of those ConcreteBuilders? If an older ConcreteBuilder is in use and the Director requests for a new part type not known by the old builder to be put in the product what should happen?