Assignment # 3 – Object Oriented Programming

Assigned: October 23, 2014
Due: November 5, 2014 (11:59 pm)

1. An important technique in *Design Patterns* is the use of `delegation' to simulate `inheritance'. A systematic way for doing this transformation was sketched in Lecture 14. Extend this technique so that the inheritance of attributes and methods, and also method over-riding are handled. Apply the resulting technique to the program in Delegation.java which defines four classes, C1, C11, C111, and C112, where the classes C111 and C112 extend C11 which in turn extends C1.

The result of your transformation should be four classes called D1, D11, D111, and D112 which correspond to C1, C11, C111, and C112 respectively, but do not make use of class inheritance. Make the classes D1, D11, D111, and D112 implement interfaces I1, I11, I111 and I112 respectively, where the interfaces I111 and I112 should inherit from I11 which in turn should inherit from I1. Define these interfaces suitably.

Refer to the file Delegation.java on Piazza→Resources→Homeworks. This file contains the 'C' classes and also a driver class called Delegation. It also contains the outline of the 'D' classes and the interfaces. Complete the 'D' classes and the interfaces in the same file. Run the program through JIVE and save the sequence diagram in Delegation.png. Submit online the files Delegation.java and Delegation.png.

Important: For full credit, the transformation should be done in a systematic way, so that it can be applied to a bigger inheritance hierarchy.

2. Define in Java an external iterator for inorder traversal of a generic binary search tree (BST), and use this iterator to define the equality of two BSTs.

The external iterator should be generic and should support two public methods: a boolean function done(), which tests whether the iterator has any more values to yield, and a function next(), which returns the next value if the iterator is not yet done. It may contain other helper methods, which should be declared private.

The equality test should repeatedly obtain one value from each tree, in alternating fashion, and compare these values for equality. It should return false if the values are not equal (mismatch). If there are no mismatches and the iterators are done at the same time, the equality test should return true.

Refer to the file GenericTreeDriver.java on Piazza→Resources→Homeworks. This file contains the complete definition of a generic BST, the main driver, and the outline of the iterator and equality test. Complete these definitions, run the program through JIVE and save the sequence diagram in Iterator.png. Submit online the files GenericTreeDriver.java and Iterator.png.

Further guidance will be provided through Piazza.

Note: This asst. may be done by a team of one or two students. The submitted file should be named Asst3_<UBIT Id1>_<UBIT Id1>.zip if done by two students; otherwise, Asst3_<UBIT Id>.zip.

End of Assignment 3