

Assignment 1 – Part2

(may be done by a team of at most two students)

Assigned: Wednesday, September 11, 2019

Due: Monday, September 23, 2019 1 (11:59 pm)

Part 2: Transform Inheritance in terms of Delegation

An important technique in the study of Object-Oriented Design is the use of *delegation* to replace *class inheritance*. A systematic approach for this transformation was presented in Lectures 4 and 5. Apply this approach to the program [Resources → Assignments → Delegation.java](#). This program defines classes [A](#), [B](#), [C](#), [D](#), [E](#), and [F](#). The result of your transformation should be definition of classes called [A2](#), [B2](#), [C2](#), [D2](#), [E2](#), and [F2](#) which correspond to classes [A](#), [B](#), [C](#), [D](#), [E](#), and [F](#) respectively, but do not make use of class inheritance. The original program and the transformed program, when executed, should produce the same results for the given test cases.

A systematic approach involves the following steps:

1. Define an interface hierarchy with interfaces [IA](#), [IB](#), [IC](#), [ID](#), [IE](#), and [IF](#) based upon classes [A](#), [B](#), [C](#), [D](#), [E](#), and [F](#). (Optimize the interfaces to avoid redundancy.)
2. Define new classes [A2](#), [B2](#), [C2](#), [D2](#), [E2](#), and [F2](#) which implement [IA](#), [IB](#), [IC](#), [ID](#), [IE](#), and [IF](#) respectively.
3. Set up the delegation hierarchy, define delegation methods, and provide a translation for every protected abstract method.
4. Translate the pseudo-variables [this](#) and [super](#) as [this2](#) and [super2](#) respectively. Access superclass attributes by ‘[super2](#)-chaining’, e.g., [super2.super2](#). ...
5. In the translated program, you will also need to define two constructors for every non-leaf non-abstract class of the original class hierarchy, i.e., for classes [B2](#) and [D2](#).

The file [Delegation.java](#) contains the definitions of classes [A](#) ... [F](#) and also contains two tester classes called [Delegation](#) and [Delegation2](#). It also contains the outline of the classes [A2](#) ... [F2](#). Define the interfaces [IA](#) ... [IF](#) and the classes [A2](#) ... [F2](#) in the same file.

Run both test cases through JIVE and save the object and sequence diagrams in files named [A1_Delegation_obj.png](#) and [A1_Delegation_seq.png](#) (for the first test case) and [A1_Delegation2_obj.png](#) and [A1_Delegation2_seq.png](#) (for the second test case). For object diagrams, choose ‘Objects with Tables’. For sequence diagrams, choose ‘Expand Lifelines’.

Important: *Credit will be given only if the transformation is done in a systematic way. An ad hoc translation that produces correct results is not an acceptable solution for this assignment.*

What to Submit. Prepare a top-level directory named [A1_Part2_UBITId1_UBITId2](#) if the assignment is done by a team of two students; otherwise, name it as [A1_Part2_UBITId](#) if it is done solo. (Order the [UBITId](#)s in alphabetic order, in the former case.) Place [Delegation.java](#) and all diagrams in this directory. Compress the directory and submit the resulting compressed file using the [submit_cse522](#) command. Only one submission per team is required.

End of Assignment 1 Part2