CSE505 – Spring 2018

Assignment 5 – Equivalences

Due Date: **May 10 (11:59 pm)**

You may work in pairs for this assignment.

Problem 1: Consider the following Python program defining a recursive generator called `mystery(x)` which is invoked by a list comprehension.

```
def mystery(x):
        if type(x) is int:
                yield x
        else:
                for e in x:
                        for y in mystery(e):
                                yield y

input = … enter suitable data for testing …

output = [x for x in mystery(input)]
print(output)
```

(a) Give a brief description of the function that `mystery` performs. You should state:
(i) what is the most general form of input that `mystery` can accept without error; and
(ii) what is the answer that it generates.

Include this description as a comment at the top of your answer file, called `mystery.py`.

(b) Write in Python an equivalent program which:
(i) replaces `mystery` by *a higher-order* definition called `mystery2`, which does ***not*** make use of any generator – not even the built-in list generator;
and
(ii) replaces the list comprehension by code that does ***not*** use any list comprehension.

Your program should have the same input-output behavior as the original program.

For full credit, a systematic methodology *must* be followed. Guidance for developing your solution: Lecture 23, slides 10-15. Be sure to apply the final optimization step which minimizes the number of extra functions used.

Problem 2: This problem explores the equivalence of inheritance and delegation. The program `AbsTree.java` posted at `Piazza→Homeworks` consists of three primary classes along with a driver class which contains the `main` method for testing.

An abstract class, `AbsTree`, defines the `insert` and `print` methods for a *binary search tree* of integers. `AbsTree` factors out the common details of its two subclasses, `Tree` and `DupTree`. The difference between these two classes is that the `insert` of `Tree` ignores duplicate insertions of the same value into the tree, whereas the `insert` of `DupTree` keeps track of duplicate insertions through a `count` field. Also, the `print` method for `Tree` prints only the `value` field whereas the `print` for `DupTree` prints the `value` as well as the `count` fields.

(a) Transform the given tree classes into an equivalent set of classes, called `AbsTree2`, `Tree2`, and `DupTree2` such that they provide the same functionality for `insert` and `print` as `AbsTree`, `Tree`, and `DupTree`.

For full credit, a systematic methodology *must* be followed. Guidance for developing your solution: Lecture 23, slides 16-17. Be sure to apply the final optimization step which minimizes the number of interfaces used.

Develop your solution in a file called `AbsTree2.java`.

(b) Check the correctness of your implementation by running `AbsTree.java` and `AbsTree2.java` under JIVE and generating the object diagrams, as indicated below.

(i) Choose 'Objects with Tables' from the inverted triangle menu (in the Object Diagram).
(ii) Run the program to completion.
(iii) Choose 'Focus on Call Paths' from the inverted triangle menu (in the Object Diagram).
(iv) From the top-level Search menu, choose `Search →JIVE Search → Variable Changed`. Enter the variable name as `value` in the text box, choose `==`, and enter `505` in the text box.
(v) Step forwards in the Object Diagram until all constructor fields are set.
(vi) Export the object diagram using the `SavesAs` button. Name the file as `inheritance.png` or `delegation.png` depending upon whether you ran `AbsTree.java` or `AbsTree2.java`.

Problem 3: An equivalence problem in concurrency and threads, to be posted early next week. Your program is to be written in a file called `Concurrency.java`.

WHAT TO SUBMIT (May 10):

Make a directory called `A5_UBITId` if working solo or make a directory called `A5_UBITId1_UBITId2` if working as a pair (give UBITId's in alphabetic order). Put in this directory the files `mystery.py`, `AbsTree.java AbsTree2.java`, `inheritance.png`, `delegation.png`, and `Concurrency.java`. Compress the directory, and submit it using the `submit_cse505` command.

End of Assignment 5