Project 2: Strong Mobility for Java

CSC 7351, Fall 2012

Due: 27 November 2012

For this project, implement the basics of a translator from Java with strong mobility into Java with weak mobility. Follow the design of the mobility translator in Section 3 of the paper

```
http://www.csc.lsu.edu/~gb/Brew/Publications/MobilityDesign1.pdf
```

Translate all the code provided in the source file. That means, you do not need special treatment in semantic analysis for the interface `Mobile` and the classes `MobileObject` and `ContextInfo` described in Section 2.

Since testing mobile code is tedious and since the weak mobility library would take a long time to set up and get familiar with, simply implement the translation for methods and classes without migrating the objects. The translated code should behave the same as the untranslated code. For testing the "mobility," you could stop and restart the main tread on the same machine.

To keep it simpler, you don't need to provide support for inheritance and method dispatch (although it may not be too difficult to add that), `for` loops and `do-until` loops, exception handling, `switch` statements, Aglets message sends (simply use method calls), the Aglets `onCreation` method (simply use a constructor), the Aglets `dispatch` method, and synchronization. You do not need to generate locking calls for protecting the stack, but write your code generation, so that adding the locking calls would be easy.

Implement the translation for method bodies first while writing the code that would need to be generated for the class by hand. Try to design your translation such that the performance improvements described in Section 6 would be easy to add.

Submission

Implement your mobility translator in Polyglot. A new version of Polyglot, Version 2.5.1, recently became available. I recommend that you simply stick with Version 2.5, though, unless there are bug fixed in 2.5.1 that you really need.

To submit your projejct, upload the code to your `classes.csc.lsu.edu` account in a directory `prog2` and submit it using

```
~cs7351_bau/bin/p_copy 2
```

Please provide a README file (as text, PDF, or Word) as well as test cases with your submission. In the README file document and justify your language design choices, explain the architecture of your code, and explain what's working and what's not yet implemented. Write test cases to highlight any special features of your implementation.