Winston Wang

Mr. Kuszmaul

8 September 2014

AP Computer Science

Scope Homework

1. The code would print out the following:  
   3  
   3.14  
   3  
   7  
   7  
   88  
   2  
   2.7
2. The number 7 is printed twice because Myscope.*b* is a static variable, which means it is not associated with the particular instance of Myscope. Therefore, when the main method changes myInstance.*b*, it changes b for myOtherInstance as well. On the other hand, a is not static. Therefore, when myInstance.a is changed to 88, myOtherInstance.a does not change and remains 2.
3. It will throw the following exception:  
   “Exception in thread "main" java.lang.Error: Unresolved compilation problem: Cannot make a static reference to the non-static field MyScope.a”  
   The reason is MyScope.a does not exist. Because a is not static, it does not exist outside instances of MyScope. Therefore, a cannot be called directly from the class. It an instance of MyScope must be initialized first.
4. I would like to make it possible to declare a variable that is local everywhere in the class it is declared in, using a new special word LocalInClass. Therefore, if the following code were run:

public class MyScope   
{  
 public static void main(String[] args)   
 {  
 for (int LocalInClass i = 1; i<5; i++)  
 {  
 System.out.println(i);  
 }  
 System.out.println(i);  
 }  
}

This program, without the LocalInClass keyword, would result in an error because i would not be defined outside of the for loop. However, because of the LocalInClass keyword, the output of this program would be:

1

2

3

4

5

However, there are quite a few problems with this. Most important is the fact that this keyword may not be needed, especially in this scenario. It is extremely easy to declare i outside of the for loop, which would have the same effect as adding the LocalInClass keyword to the declaration.

1. Changing the scope with a conditional variable not known until runtime would be a very bad idea. This is because it could easily result in hard to fix errors due to the scope becoming smaller or larger. For example, if the programmer programed his code on the assumption that a certain object was defined everywhere, and then during runtime the scope of that object changed to be smaller, anywhere outside the new scope where that object is referenced would result in an error.