CS 61B Lab 5
February 24-25, 2004

Goal: This lab will give you practice writing code that uses inheritance and Java interfaces.

Please make sure you have a partner for this lab.

No code is provided with this lab. Write code to experimentally resolve the following questions about Java. You will need to show your code to the TA or lab assistant.

Part I: Assignments and Casting (1 point)

Let Y be a subclass of X, and let y and x be variables of classes Y and X respectively. From lecture, you know that the assignment "x = y" is valid, but the assignment "y = (Y) x" requires a cast to compile, and will cause a run-time error if x references an object that isn't a Y.

What about arrays of objects? Suppose xa is an array of X's, and ya is an array of Y's.

(a) At compile-time, can we assign xa to ya, and vice versa? When is a cast required?
(b) At run-time, if ya references an array of Y's, can we assign it to xa? Can we then assign it back from xa to ya?
(c) If xa references an array of X's (that are not Y's), can we assign it to ya? Can we then assign it back from ya to xa? Does it make a difference if the array of type X[] references objects that are all of class Y? Why do you think this is the case?

Part II: Conflicting Declarations (1 point)

Suppose a subclass inherits a method implementation from a superclass, and implements a Java interface (that's the "interface" keyword) that contains a method with the same name and prototype.

(a) Will Java compile the result?
(b) What if the method declaration in the interface has a different return type?
(c) What if the method declaration in the interface has the same return type, but a signature with a different parameter type?
(d) What if the method declaration in the interface has the same return type, and the same number of parameters and parameter types, but those parameters have different names?

Part III: More Conflicting Declarations (1 point)

Suppose a subclass inherits a "public static final" constant from a superclass, and implements a Java interface that contains a "public static final" constant with the same name.

(a) Will Java compile the result? Does it make any difference whether the constant in the superclass and the constant in the interface have the same value?
(b) Write a main() method in the subclass that accesses the constant using the same name used in the superclass and the Java interface. Will Java compile the result? Does it make any difference whether the constant in the superclass and the constant in the interface have the same value?
(c) Figure out how to modify your main() method so that it accesses and prints one of the two conflicting values. (Look to Lecture 9 for clues.) Make sure your code compiles and runs without errors.

Part IV: Method Overriding (1 point)

Consider a subclass that has a method that overrides a method with the same prototype in its superclass.

(a) Define a variable whose static type is the subclass and which references an object of the subclass. If we cast the variable to the superclass type before calling the overridden method

```
((Superclass) subclassvariable).method();
```

does Java call the superclass method or the subclass method?
(b) Define a variable whose static type is the superclass and which references an object of the superclass (but not the subclass). If we cast the variable to the subclass type before calling the method, does Java call the superclass method or the subclass method?
(c) Suppose you have an object whose class is the subclass. Can you figure out a way to call the superclass method on that object without having to go through the subclass method of the same name?

Check-off

Show your TA or Lab Assistant your code and, in some cases, its output.

1 point: Give your answers for Part I. Show the code with which you answered Part I (c).
1 point: Give your answers for Part II.
1 point: Give your answers for Part III. Show _and_run_ the code with which you answered Part III (c).
1 point: Give your answers for Part IV. Show the code with which you answered Part IV (c).