Conrad Horton
CSC203 Jun 2016
Inheritance and Polymorphism Lab
20160715

Part I

1. How does inheritance promote software reusability?

   You can use existing classes and methods as building blocks for subclasses.

2. Explain protected member access.

   Protected members are hidden from non class members, but are inherited by child classes.

3. Explain the difference between composition (i.e., the has-a relationship) and inheritance (i.e., the is-arelationship).

   Composition: Has data members of other class types.

   Inheritance: Has properties of the parent class, and can also be treated as the parent.

4. When an object of a subclass is created, the constructor for that subclass object is invoked to initialize the subclass object. Explain the complete details of initializing an object of class BasePlusCommissionEmployee with six arguments. Assume the CommissionEmployee-BasePlusCommissionEmployee hierarchy discussed in this chapter.

   Base constructor is called using the arguments that were passed into the child constructor. Java's Object constructor is called implicitly. The parent's data members are then initialized.

5. Explain how to invoke a superclass method from a subclass method for the case in which the subclass method overrides a superclass method and the case in which the subclass method does not override a superclass method.

   Overridden: super.doStuff();

   Not Overriden: doStuff();

6. Describe the concept of polymorphism.

   Being able to modify base functionality to extend upon the abilities of the parent class to fit the child's needs.

7. Define what it means to declare a method final and what it means to declare a class final.

   Final methods can't be overridden. Final classes cannot be extended.

8. What happens when a class specifies that it implements an interface, but does not provide declarations of all the methods in the interface?

   An error occurs.

9. Describe how to determine the class name of an object's class.

   Using **instanceof** clause.

10. Call method getClass on an object to obtain an object of type Class that represents the object's type. Then call

    ??????

11. Distinguish between an abstract class and a concrete class.

    Abstract classes are classes for which you only intend to subclass, because they cannot have instance objects created of them; these are used to create concrete classes from which instance methods can be created.