Solutions to Quick Check Questions

13

Inheritance and Polymorphism

13.1 A Simple Example

1. Define the Reptile class as a subclass of the Pet class. The speak method returns an empty string.

   Answer:

```
class Reptile extends Pet {

   public String speak( ) {
      return "";
   }
}
```

2. Which one of the following statements is valid?

```
Pet p = new Cat();  <--- Valid

Cat c = new Pet();  <--- Invalid
```

3. Is the following code valid?

```
Pet p = new Dog();
System.out.println(p.fetch());
```

Answer:

No. You must typecast p to Dog as ((Dog)p).fetch().

13.2 Defining Classes with Inheritance

1. Which is the subclass and which is the superclass in the following declaration?

```
class X extends Y { ... }
```

X is the subclass, and Y is the superclass.

2. Which visibility modifier allows the data members of a superclass to be accessible to the instances of subclasses?

The purpose of the access modifier protected *is to allow the access to the data members of a superclass from the instances of the subclasses, but disallow the access from the outside classes. The access modifier* public*, of course, will allow access to all other classes including the subclasses, but the purpose of the* public *modifier is not allowing access to the subclasses.*

13.3 Using Classes Effectively with Polymorphism

1. Suppose Truck and Motorcycle are subclasses of Vehicle. Which of the following declarations are invalid?

```
Truck       t = new Vehicle();
Vehicle     v = new Truck();
Motorcycle m1 = new Vehicle();
Motorcycle m2 = new Truck();
```

The declarations for t, m1, and m2 are invalid.

2. What is the purpose of the instanceof operator?

To determine whether an object is an instance of a given class or not.

13.4 Inheritance and Member Accessibility

1. If X is a private member of the Super class, is X accessible from a subclass of Super?

   No. A private member of a class is only accessible within the class.

2. If X is a protected member of the Super class, is X of one instance accessible from another instance of Super? What about from the instances of a subclass of Super?

   Yes. Data members of one instance are fully accessible from any other instances of the same class.

   Yes. Protected (and public) members of a superclass are accessible from the instances of a subclass.

13.5 Inheritance and Constructors

1. How do you call the superclass's constructor from its subclass?

   By using the reserved super *and passing the correct number and types for the arguments. Also, the call to* super *must be the first statement in the subclass constructor.*

2. What statement will be added to a constructor of a subclass if it is not included in the constructor explicitly by the programmer?

   A call to the superclass's constructor using the reserved word super *with no arguments.*

3. Modify the definition of GraduateStudent and UndergraduateStudent in Section 13.1 so we can create their instances in the following way:

```
student1 = new UndergraduateStudent();
student2 = new UndergraduateStudent("Mr. Espresso");
student3 = new GraduateStudent();
student4 = new GraduateStudent("Ms. Latte");
```

   Add the following constructors:

```
class UndergraduateStudent extends Student {
   public UndergraduateStudent( ) {
      super( );
```

```
    }

    public UndergraduateStudent( String name ) {
        super( name );
    }
    . . .
}

class GraduateStudent( ) {
    public GraduateStudent( ) {
        super( );
    }

    public GraduateStudent( String name ) {
        super( name );
    }
    . . .
}
```

13.6 Abstract Superclasses and Abstract Methods

1. Can you create an instance of an abstract class?

   No.

2. Must an abstract class include an abstract method?

   No. The class could inherit abstract methods from its superclass.

3. What is wrong with the following declaration?

```
class Vehicle {
    abstract public getVIN();
    ...
}
```

   The Vehicle *class must be declared as* abstract *because it contains an abstract member.*

   The getVIN *method does not include the return type. Moreover, the* public *modifier is redundant because an abstract method is implicitly a public method.*

13.7 Inheritance versus Interface

No Quick Check Questions.

13.8 Sample Development: Computing Course Grades

No Quick Check Questions.