

## Java Fundamentals

### 7-4: Inheritance

### Practice Activities

#### Lesson Objectives:

- Demonstrate and explain UML (Unified Modeling Language) class diagrams
- Use the extends keyword to inherit a class
- Compare and contrast superclasses and subclasses
- Describe how inheritance affects member access
- Use super to call a superclass constructor
- Use super to access superclass members
- Create a multilevel class hierarchy
- Recognize when constructors are called in a class hierarchy
- Demonstrate understanding of inheritance through the use of applets
- Recognize correct parameter changes in an existing applet

#### Vocabulary:

Identify the vocabulary word for each definition below.

|  |  |
|--|--|
|  | When there is no access modifier. Same access as public, except not visible to other packages.   |
|  | The keywords used to declare a class, method, or variable as public, private, or protected. Default is when there is no access modifier.   |
|  | Classes that are more specific subsets of other classes and that inherit methods and fields from more general classes.   |
|  | A keyword in Java that allows you to explicitly declare the superclass of the current class.   |
|  | A programming philosophy that promotes protecting data and hiding implementation in order to preserve the integrity of data and methods.   |
|  | Visible only to the class where it is declared.  |
|  | A structure that categorizes and organizes relationships among ideas, concepts of things with the most general or all-encompassing component at the top and the more specific, or component with the narrowest scope, at the bottom. |
|  | Visible to all classes.  |
|  | Classes that pass down their methods to more specialized classes.  |
|  | The concept in object-oriented programming that allows classes to gain methods and data by extending another classes fields and methods.   |
|  | Visible to the package where it is declared and to subclasses in other packages.   |
|  | A standardized language for modeling systems and structures in programming.  |
|  | A keyword that allows subclasses to access methods, data, and constructors from their parent class.  |
|  | A helpful term used to conceptualize the relationships among nodes or leaves in an inheritance hierarchy.  |

## Try It/Solve It:

1. Modify the existing applet to change all the colors to black, white, and gray using the code below:

```
import java.awt.*;
import java.applet.*;

public class DrawShapes extends Applet {
    Font font;
    Color redColor;
    Color blueColor;
    Color backgroundColor;
    public void init() {
        //The Font is Arial size, 18 and is italicized
        font = new Font("Arial",Font.ITALIC,18);

        //Some colors are predefined in the Color class
        redColor = Color.red;
        backgroundColor = Color.orange;

        //Colors can be created using Red, Green, Blue values
        blueColor = new Color(0,0,122);

        //Set the background Color of the applet
        setBackground(backgroundColor);
    }
    public void stop() {
    }
    /**
     * This method paints the shapes to the screen
     */
    public void paint(Graphics graph) {
        //Set font
        graph.setFont(font);
        //Create a title
        graph.drawString("Draw Shapes",90,20);

        //Set the color for the first shape
        graph.setColor(blueColor);

        // Draw a rectangle using drawRect(int x, int y, int width, int height)
        graph.drawRect(120,120,120,120);

        // This will fill a rectangle
        graph.fillRect(115,115,90,90);

        //Set the color for the next shape
        graph.setColor(redColor);

        //Draw a circle using drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
        graph.fillArc(110,110,50,50,0,360);

        //Set color for next shape
        graph.setColor(Color.CYAN);

        //Draw another rectangle
        graph.drawRect(50,50,50,50);

        // This will fill a rectangle
        graph.fillRect(50,50,60,60);
    }
}
```

2. Draw simple UML Diagrams with the following classes:

- Tree, Hardwood, Softwood, Birch, Ash, Cedar, Pine, Red Pine, where Hardwood and Softwood are types of trees, Birch and Ash are types of Hardwoods and Cedar and Pine are types of Softwoods. Red Pine is a type of Pine.
- A, B, C, D, E, F, where B and C are a type of A, D is a type of B, E is a type of C, and F is a type of E.

3. Create a class hierarchy representing Students in a university. You know that Students are a type of Person, but Students have more specific characteristics like having a grade point average (GPA), student identification number (ID) and a discipline of study, or major (Mathematics, Computer Science, Literature, Psychology, etc.). Create a subclass of Person called Student using the code for Person below and the specifications listed below:

- Students have personal data that helps identify them to university administrators. Create variables for a Student ID number, a GPA, which gives the average grade points for the student, a major, degree that he or she is earning (Bachelor's, Masters, Ph.D), and his or her expected graduation year. Carefully consider whether these should be public or private data.
- For methods you declare private, you may want to provide access methods for other classes to retrieve this data.
- Create a detailed UML diagram listing all of the data and methods of each class (for both Person and Student).
- Create a method that allows administrators to change a student's major
- Create a method that calculates a student's GPA by taking in an array of his or her grades. Take the average of all of the student's grades using the following breakdown.

| Grade | Grade Point Equivalent |
|-------|------------------------|
| A     | 4                      |
| A-    | 3.67                   |
| B+    | 3.33                   |
| B     | 3                      |
| B-    | 2.67                   |
| C+    | 2.33                   |
| C     | 2                      |
| D     | 1                      |
| F     | 0                      |

Code for Person class:

```
import java.util.Date;
```

```
public class Person {
    private String firstName;
    private String middleName;
    private String lastName;
    private Date dateOfBirth;

    public Person(String firstName, String middleName,
                  String lastName, Date dateOfBirth){
        this.firstName = firstName;
        this.middleName = middleName;
        this.lastName = lastName;
        this.dateOfBirth = dateOfBirth;
    }
}
```

```
/**
 * Returns a String of the firstName
 * @return firstName
 */
public String getFirstName(){
```

```

        return firstName;
    }

    /**
     * Returns a string of the middleName
     * @return middleName
     */
    public String getMiddleName(){
        return middleName;
    }

    /**
     * Returns a String of the lastName
     * @return lastName
     */
    public String getLastName(){
        return lastName;
    }

    /**
     * Returns a concatenated string of the Person's name
     * @return the Person's first, middle, and last name.
     */
    public String getName(){
        return firstName + " " + middleName + " " + lastName;
    }

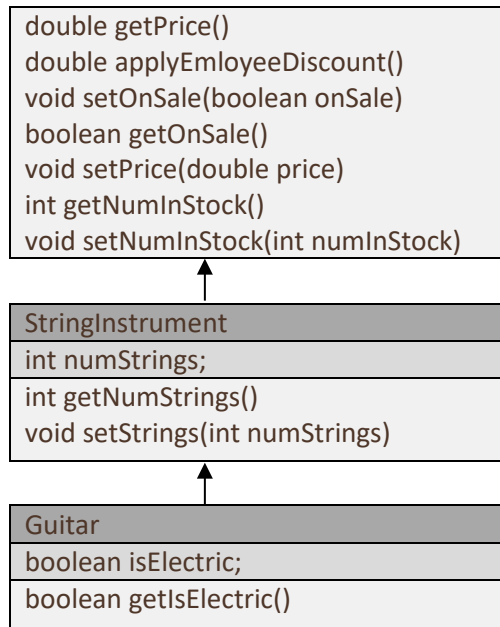
    /**
     * Returns the Person's date of birth as a date type
     * @return a Date type of the Person's date of birth.
     */
    public Date getDateOfBirth(){
        return dateOfBirth;
    }
}

```

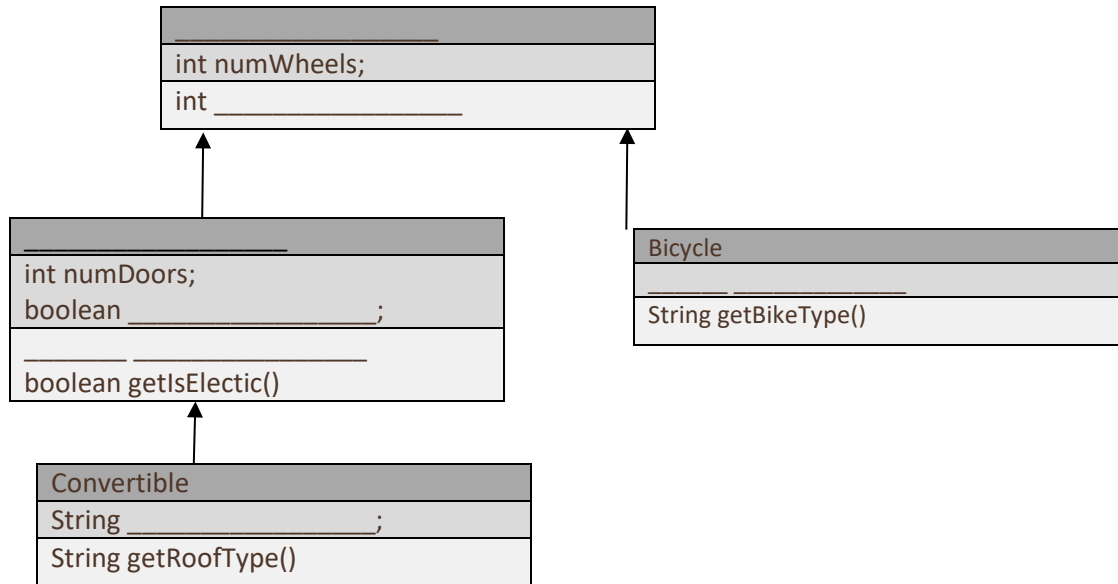
4. True/False – A subclass is able to access this code in the superclass: Why?
  - a. public String aString;
  - b. protected boolean aBoolean;
  - c. int anInt;
  - d. private double aDouble;
  - e. public String aMethod()
  - f. private class aNestedClass
  - g. public aClassConstructor()
5. Imagine you own a music shop that sells musical instruments. Create classes representing an inheritance hierarchy of musical instruments using the following UML diagram:

- Employee discount is 25% off the instrument (Hint: 75% of the initial cost).
- If an item is onSale, then the price returned for getPrice() should be 15% off.

| Instrument  |
|---|
| boolean onSale;<br>double price;<br>int numInStock; |



6. Using the code and UML diagram below, fill in the blanks with the correct keywords or class names. If a keyword is missing from the code, fill in the keyword that fits. If the class or data is missing from the UML, fill in the correct information.



```

public class Vehicle {
    _____ int numWheels;
    _____ Vehicle(int numWheels)
    {
        this.numWheels = numWheels;
    }

    public int getWheels() {
        return wheels;
    }
}

public class Car _____ Vehicle {
    _____ int numDoors;
    _____ boolean isElectric;

    public Car (int numWheels, int numDoors, boolean isElectric) {
        _____(numWheels);
        this.numDoors = numDoors;
        this.isElectric = isElectric;
    }

    public int getNumDoors() {
        return _____;
    }

    public boolean getIsElectric() {
        return isElectric;
    }
}

public class Bicycle _____ Vehicle {
    //Mountain bike, road bike, recumbent bike.. etc
    _____ String bikeType;

    public Bicycle(int numWheels, String bikeType) {
        super(numWheels);
        this.bikeType = _____;
    }

    public _____ getBikeType() {
        return bikeType;
    }
}

public class Convertible _____ Car {
    //Soft top or rag top, or hard top
    _____ String roofType;

    public Convertible(int numWheels, int numDoors, _____ isElectric, String roofType) {
        super(numWheels, _____, _____);
        this.roofType = roofType;
    }

    _____ String getRoofType() {
        return roofType;
    }
}

```

7. Given the classes Shape, Rectangle, Circle, Triangle, Square, and Object, write a UML diagram using the following relationships:

- A Square is a Rectangle
- A Rectangle, Triangle, and Circle are all Shapes

Use the simple way of diagramming a class.

- **Hint:** Start with the broadest, or most general class.

8. Given the classes Crab, Crustacean, Mammal, Dog, Hermit Crab, Animal, and Object, write a UML diagram using the following relationships:

- A Crab is a Crustacean.
- A Hermit Crab is a Crab.
- A Dog is a Mammal.
- Mammals and Crustaceans are types of Animals.

Use the simple way of diagramming a class.

- **Hint:** Start with the broadest or most general class.