Conceptos fundamentales de Java
7-4: Herencia
Actividades practices

Objetivos de la lección:

- Demonstrar y explicar los diagramas de clases de UML (Unified Modeling Language - Lenguaje de Modelado Unificado)
- Usar la palabra clave "extends" (extiende) para heredar una clase
- Comparar y contrastar superclases y subclases
- Describir cómo la herencia afecta el acceso a los miembros
- Usar "super" para invocar a un constructor de una superclase
- Usar "super" para acceder a los miembros de una superclase
- Crear una jerarquía de clases multinivel
- Reconocer cuando los constructores son invocados en una jerarquía de clases
- Aplicar referencias de superclases a objetos de subclases
- Demonstrar comprensión del concepto de herencia a través del uso de applets
- Reconocer los cambios de parámetros correctos en un applet existente

Vocabulario:

Identifique el término correspondiente a cada una de las siguientes definiciones.

	Cuando no existe modificador de acceso. El mismo acceso que el público, salvo que no es visible para otros paquetes.
	Las palabras clave utilizadas para declarar una clase, método o variable como públicas, privadas o protegidas. Es default (predeterminado) cuando no existe modificador de acceso.
	Las clases que son subconjuntos de otras clases más específicas y que heredan métodos y campos de las clases más generales.
	Una palabra clave en Java que le permite declarar en forma explícita la superclase de la clase actual.
	Filosofía de programación que promueve la protección de los datos y el ocultamiento de la implementación a efectos de preservar la integridad de los datos y métodos.
	Visible solo para la clase cuando está declarado.
	Una estructura que categoriza y organiza relaciones entre ideas, conceptos o cosas colocando los componentes más generales o incluyentes en la parte superior y los más específicos o con un alcance más limitado en la parte inferior.
	Visible para todas las clases.
	Clases que traspasan sus métodos a clases más especializadas.
	El concepto en una programación orientada por objetos que permite que las clases obtengan métodos y datos extendiendo los métodos y los campos de otras clases.
	Visible para el paquete cuando se lo declara y para las subclases en otros paquetes.
	Un lenguaje estandarizado que se emplea en programación para modelar sistemas y estructuras.
	Una palabra clave que les permite a las subclases acceder a los métodos, datos y constructores correspondientes a la clase principal o dominante.

	Un término útil que sirve para conceptualizar las relaciones que existen entre los nodos u hojas en una jerarquía de herencia.

Inténtelo/resuélvalo:

1. Modifique el applet existente para cambiar todos los colores a negro, blanco y gris usando el código que sigue a continuación:

```
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. Dibuje Diagramas de UML simples con las siguientes clases:

    - Árbol (Tree), Árbol de madera dura (Hardwood), Árbol de madera blanda (Softwood), Abedul (Birch), Fresno (Ash), Cedro (Cedar), Pino (Pine), Pino Rojo (Red Pine), donde Árbol de madera dura y Árbol de madera blanda son tipos de árboles, Abedul y Fresno son árboles de madera dura, y Cedro y Pino son tipos de árboles de madera blanda. Pino Rojo es un tipo de Pino.
    - A , B, C, D, E, F, donde B y C son tipos de A, D es un tipo de B, E es un tipo de C, y F es un tipo de E.

3. Crear una jerarquía de clase que represente a los Alumnos (Students) en una universidad. Usted sabe que Students (Alumnos) es un tipo de Persona (Person), pero Students tiene características más específicas como tener un promedio de calificaciones (GPA), un número de identificación de alumno (ID) y una disciplina de estudio o carrera (Matemática, Ciencias Informáticas, Literatura, Psicología, etc.). Crear una subclase de Persona llamada Alumno usando el código de Persona que figura más abajo y las especificaciones que se detallan a continuación:

    - Los Alumnos tienen datos personales que ayudan a identificarlos ante los administradores de la universidad. Crear variables para el número de ID de Alumno, un GPA, que indica el promedio de calificaciones del alumno, una carrera, el título que él o ella han obtenido (Licenciatura, Máster, Ph.D), y su año de graduación esperado. Considere cuidadosamente si estos datos deberían ser públicos o privados.
    - Para los métodos que usted declare como privados, usted podrá desear proveer métodos de acceso para que otras clases recuperen estos datos.
    - Crear un diagrama de UML detallado que liste todos los datos y métodos de cada clase (tanto para Persona como para Alumno).
    - Cree un método que les permita a los administradores cambiar la carrera de un alumno.
    - Cree un método que calcule el GPA (promedio) de un Alumno considerando un array (arreglo) de sus calificaciones. Tome el promedio de todas las calificaciones obtenidas por los alumnos usando el siguiente esquema.

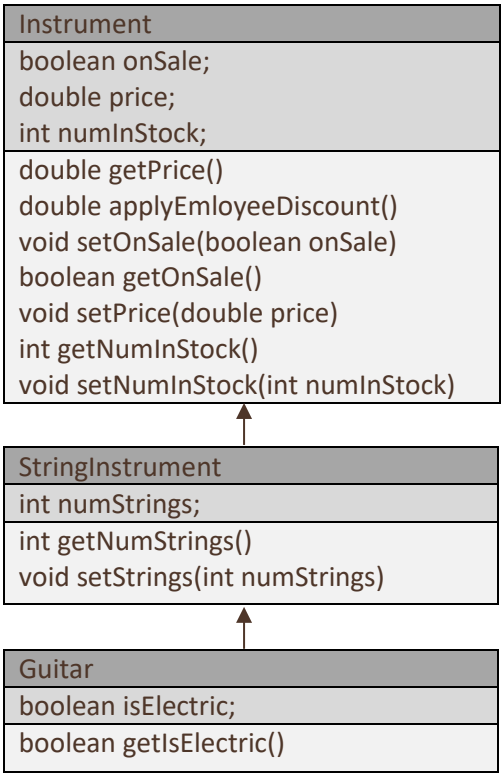
Calificación	Equivalente de la calificación en puntos
A	4
A-	3.67
B+	3.33
B	3
B-	2.67
C+	2.33
C	2
D	1
F	0

Code for Person class:

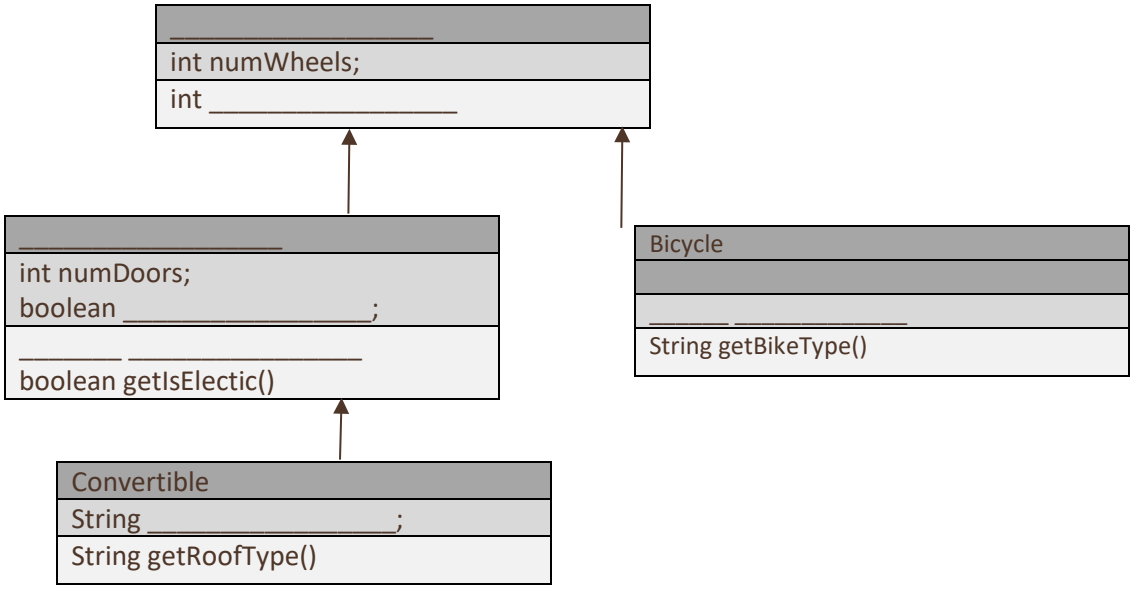
```
import java.util.Date;

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. Verdadero o falso – Una subclase es capaz de acceder a este código en la superclase: ¿Por qué?
    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 su propia tienda de música que vende instrumentos musicales. Crear clases que representen una jerarquía de herencia de instrumentos musicales usando el siguiente diagrama de UML:
    - El descuento para empleados es del 25% sobre el precio del instrumento (consejo: 75% del costo inicial).
    - Si un ítem está onSale (a la venta), entonces el precio devuelto para getPrice() debería ser de un 15% menos.



6. Usando el código y el diagrama de UML que figura más abajo, complete los espacios en blanco con las palabras clave o nombres de clases correctos. Si falta la palabra clave en el código, complete con la palabra clave que corresponda. Si faltan los datos o la clase en el diagrama UML, complete con la información correcta.




```
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. Dadas las clases Forma (Shape), Rectángulo (Rectangle), Círculo (Circle), Triángulo (Triangle), Cuadrado (Square) y Objeto (Object), escriba un diagrama de UML usando las siguientes relaciones:
    - Un Cuadrado es un Rectángulo
    - Un Rectángulo, un Triángulo y un Círculo son todas Formas

    Use la forma simple de diagramar una clase.
    - **Consejo:** Comience con la clase más amplia o general.

8. Dadas las clases Cangrejo (Crab), Crustáceo (Crustacean), Mamífero (Mammal), Perro (Dog), Cangrejo Ermitaño (Hermit Crab), Animal (Animal) y Objeto (Object), escriba un diagrama de UML usando las siguientes relaciones:
    - Un Cangrejo es un Crustáceo.
    - Un Cangrejo Ermitaño es un Cangrejo.
    - Un Perro es un Mamífero.
    - Mamíferos y Crustáceos son tipos de Animales.

    Use la forma simple de diagramar una clase.
    - **Consejo:** Comience con la clase más amplia o general.