• Demonstrar y explicar los diagramas de clases de UML (Unified Modeling Language - Lenguaje de Modelado Unificado)

Una clase se puede dibujar como una caja que contiene el nombre de la clase, las variables de instancia y los métodos, las clases también se pueden dibujar como cajas simples, incluyendo solo el nombre de la clase, aunque puede ser útil incluir los métodos, no es necesario incluir cada atributo o aquellos atributos que representan conjuntos de datos (como las arreglas) Incluya únicamente los atributos más útiles.

|  |
| --- |
| Forma |
| String color |
| getColor() |

Los diagramas de clase se conectan usando líneas con flechas, la conexión de las líneas varía según la relación entre las clases, para la herencia, se utiliza una línea sólida y una flecha triangular para representar la relación “is-a”

|  |
| --- |
| Rectángulo |
| public int length  public int width |
| public int getLength()  public int getWidth() |

• Usar la palabra clave "extends" (extiende) para heredar una clase

En Java, puede elegir qué clases desea heredar al usar la palabra clave extends (extiende), la palabra clave extends le permite designar la superclase que posee los métodos que desea heredar o de quiénes son los métodos y datos que desea extender.

Por ejemplo, para heredar métodos de la clase Forma, use extends al crear la clase Rectángulo.

public class Rectangle extends Shape

{

//code

}

• Comparar y contrastar superclases y subclases

Superclase frente a Subclase

Las clases se pueden derivar de, o evolucionar desde, las clases de padres, o sea, que contienen los mismos métodos y campos que sus padres, pero se pueden considerar una forma más especializada de las clases de sus padres.

Las superclases contienen métodos y campos que se traspasan a todas sus subclases

Subclases:

−Heredan métodos y campos de sus superclases

−Pueden definir métodos o campos adicionales que la

superclase no tiene

La diferencia entre una subclase y una superclase es la siguiente:

|  |  |
| --- | --- |
| Superclase | Subclase |
| Es la clase más general desde la que otras clases derivan sus métodos y datos | La clase más específica que se deriva o hereda de otra clase (la superclase) |

• Describir cómo la herencia afecta el acceso a los miembros

Con la encapsulación, ni siquiera las subclases pueden acceder a los métodos y variables privados, los modificadores públicos y protegidos brindan acceso a los métodos y variables de las superclases.

|  |  |
| --- | --- |
| Modificador de acceso | Acceso autorizado al: |
| public (público) | Todas las clases |
| private (privado) | Solo la clase propiamente dicha |
| protected (protegido) | Todas las subclases y todas las clases en el mismo paquete |
| "default“ ("predeterminado“) | Si no se especifica una palabra clave, se puede acceder a las variables de los miembros mediante cualquier clase en el paquete |

• Usar "super" para invocar a un constructor de una superclase

public class Rectangle extends Shape

{

private int length;

private int width;

//Constructor

public Rectangle(String color, int length, int width)

{

super(color);

this.length = length;

this.width = width;

}//end constructor

}//end class Rectangle

• Usar "super" para acceder a los miembros de una superclase

public int setWidth(int width){

super.setLength(width);

super.setWidth(width);

}

• Crear una jerarquía de clases multinivel

public class Square extends Rectangle {

private boolean isFilled;

public Square(String color, int size, boolean isFilled){

super(color, size, size);

this.isFilled = isFilled;

}//end constructor

public void setLength(int length) {

super.setLength(length);

super.setWidth(length);

}//end method setLength

public void setWidth(int width) {

super.setWidth(width);

super.setLength(width);

}//end method setWidth

public boolean getIsFilled() {

return isFilled;

}//end method getIsFilled

}//end class Square

• Reconocer cuando los constructores son invocados en una jerarquía de clases

//Constructor

public Rectangle(String color, int length, int width)

{

super(color);

this.length = length;

this.width = width;

}//end constructor

• Aplicar referencias de superclases a objetos de subclases

Shape (superclass)

public String color

public String getColor()

Rectangle (subclass)

public String color

public String getColor()

//Rectangle-only data

public int length

public int width

public int getLength()

public int getWidth()

• Demonstrar comprensión del concepto de herencia a través del uso de applets

Una Applet de Java es un programa Java basado en la web que se puede incorporar a un navegador el Applet de clase se puede extender para crear Applets especiales usando alguno de los métodos principales en la clase del Applet.

• Reconocer los cambios de parámetros correctos en un applet existente

import java.applet.Applet;

import java.awt.Graphics;

import java.awt.Graphis2D;

import java.awt.Rectangle;

public class RectangleApplet extends Applet{

public void paint(Graphics g){

Graphics2D g2 = (Graphics2D)g;

Rectangle testRectangle = new Rectangle(5,10,20,30);

g2.draw(testRectangle);

}//end method paint

}//end class RectangleApplet

|  |  |
| --- | --- |
| "default“ ("predeterminado“) | Cuando no existe modificador de acceso. El mismo acceso que el público, salvo que no es visible para otros paquetes. |
| "default“ ("predeterminado“) | 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. |
| Subclases | Las clases que son subconjuntos de otras clases más específicas y que heredan métodos y campos de las clases más generales. |
| extends | Una palabra clave en Java que le permite declarar en forma explícita la superclase de la clase actual. |
| Encapsulamiento | 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. |
| private (privado) | Visible solo para la clase cuando está declarado. |
| Jerarquía | 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. |
| public (público) | Visible para todas las clases. |
| Superclase | Clases que traspasan sus métodos a clases más especializadas. |
| Herencia | 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. |
| protected (protegido) | Visible para el paquete cuando se lo declara y para las subclases en otros paquetes. |
| UML (Unified Modeling Language) | Un lenguaje estandarizado que se emplea en programación para modelar sistemas y estructuras. |
| super | Una palabra clave que les permite a las subclases acceder a los métodos, datos y constructores correspondientes a la clase principal o dominante. |
| Is A | Un término útil que sirve para conceptualizar las relaciones que existen entre los nodos u hojas en una jerarquía de herencia. |

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

•

TREE

SOFTWOOD

HARDWOOD

PINE

CEDAR

BIRCH

ASH

RED PINE

•

A

B

C

D

E

F

4. Verdadero o falso – Una subclase es capaz de acceder a este código en la superclase: ¿Por qué?

a. public String aString; Verdadero - Los miembros públicos de una superclase pueden ser accedidos por sus subclases.

b. protected boolean aBoolean; Verdadero - Los miembros protegidos de una superclase pueden ser accedidos por sus subclases.

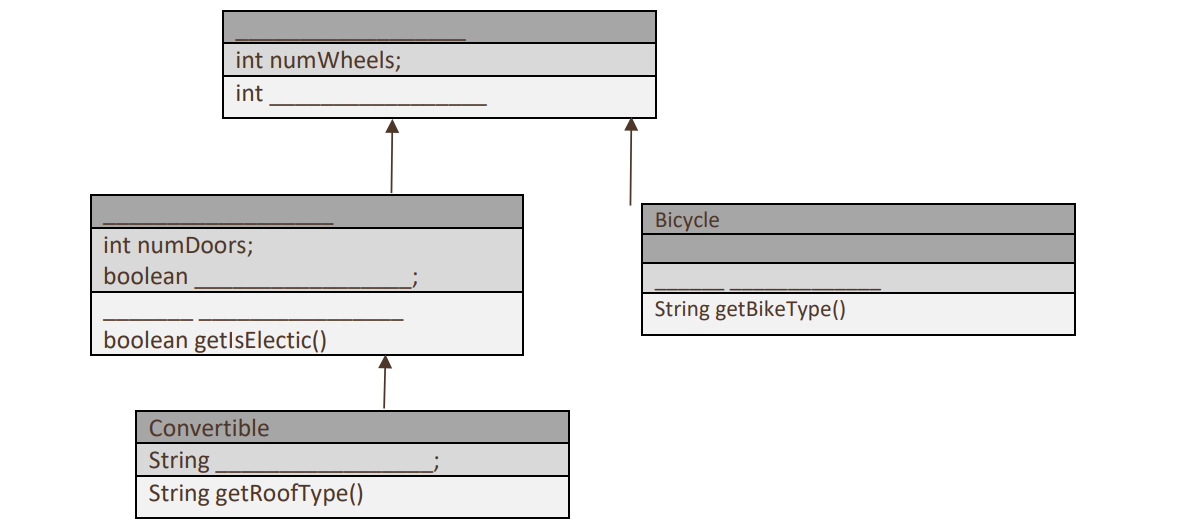
c. int anInt; Verdadero - Los miembros de paquete (sin modificador de acceso) de una superclase pueden ser accedidos por sus subclases que estén en el mismo paquete.

d. private double aDouble; Falso - Los miembros privados de una superclase no pueden ser accedidos por sus subclases.

e. public String aMethod() Verdadero - Los métodos públicos de una superclase pueden ser accedidos por sus subclases.

f. private class aNestedClass Falso - Las clases anidadas privadas de una superclase no pueden ser accedidas por sus subclases.

g. public aClassConstructor() Verdadero - Los constructores públicos de una superclase pueden ser llamados por sus subclases.

6. 

String BikeType;

getnumWheels()

RoofType

IsElectric

Int getnumDoors()

Car

Vehicle

public class Vehicle {

protected int numWheels;

public Vehicle(int numWheels)

{

this.numWheels = numWheels;

}

public int getnumWheels() {

return numWheels;

}

}

public class Car extends Vehicle {

private int numDoors;

private boolean isElectric;

public Car (int numWheels, int numDoors, boolean isElectric) {

super (numWheels);

this.numDoors = numDoors;

this.isElectric = isElectric;

}

public int getNumDoors() {

return numDoors;

}

public boolean getIsElectric() {

return isElectric;

}

}

public class Bicycle extends Vehicle {

//Mountain bike, road bike, recumbent bike.. etc

private String bikeType;

public Bicycle(int numWheels, String bikeType) {

super(numWheels);

this.bikeType = bikeType;

}

public String getBikeType() {

return bikeType;

}

}

public class Convertible extends Car {

//Soft top or rag top, or hard top

private String roofType;

public Convertible(int numWheels, int numDoors, boolean isElectric, String roofType) {

super(numWheels, numDoors, isElectric);

this.roofType = roofType;

}

public String getRoofType() {

return roofType;

}

}

7.

OBJECT

SHAPE

CIRCLE

SQUARE

TRIANGLE

RECTANGLE

8.

HERMIT CRAB

DOG

CRAB

CRUSTACEAN

MAMMAL

ANIMAL

OBJECT