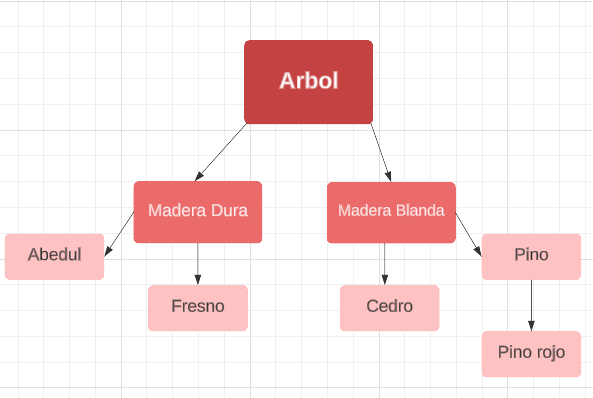
|  |  |
| --- | --- |
| **Default** | Cuando no existe modificador de acceso. El mismo acceso que el público, salvo que no es visible para otros paquetes. |
| **Modificadores de acceso** | Las palabras clave utilizadas para declarar una clase, método o variable como públicas, privadas o protegidas. Es 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. |
| **Encapsulación** | 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** | 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** | Visible para todas las clases. |
| **Superclass** | 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** | Visible para el paquete cuando se lo declara y para las subclases en otros paquetes. |
| **UML** | Un lenguaje estandarizado que se emplea en programación para modelar sistemas y estructuras. |
| **súper** | Una palabra clave que les permite a las subclases acceder a los métodos, datos y constructores correspondientes a la clase principal o dominante. |
| **relación “Is-a”** | Un término útil que sirve para conceptualizar las relaciones que existen entre los nodos u hojas en una jerarquía de herencia. |

Punto 2.

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

**a.** public String aString;

**R//** Verdadero, las subclases pueden acceder a los miembros públicos de la superclase.

**b.** protected boolean aBoolean;

**R//** Verdadero, las subclases pueden acceder a miembros protegidos de la superclase.

**c.** int anInt;

**R//** Verdadero, las subclases pueden acceder a los miembros con acceso al paquete de superclase default, si están en el mismo paquete.

**d.** private double aDouble;

**R//** Falso, las subclases no pueden acceder a los miembros privados de las superclases.

**e.** public String aMethod()

**R//** Verdadero, las subclases tienen acceso a los métodos públicos de la superclase.

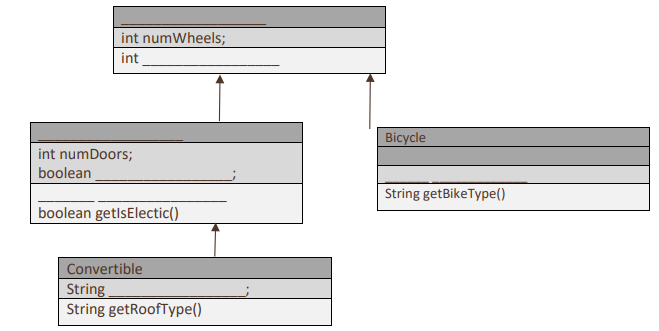
**f.** private class aNestedClass

**R//** Falso, las subclases no pueden acceder a las clases internas privadas de la superclase.

**g.** public aClassConstructor()

**R//** Falso, el constructor no es un miembro por lo tanto las subclases no pueden acceder a él.

6.



**getNumWheels()**

**roofType**

**bikeType**

**String**

**getNumDoors**()

**int**

**isElectric**

Car

**Vehicle**

* **Las palabras rojas son las que se tenían que completar.**

public class Vehicle {

**private** int numWheels;

**public** Vehicle(int numWheels) {

this.numWheels = numWheels;

}

public int getWheels() {

return wheels;

}

}

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 **extend** 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;

}

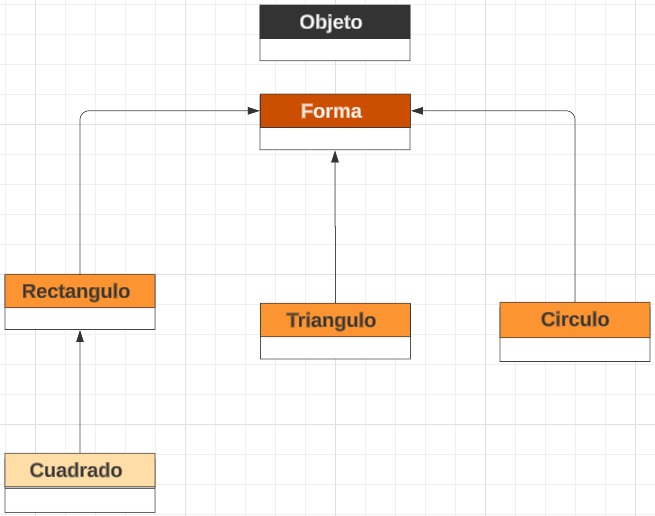
**public** String getRoofType() {

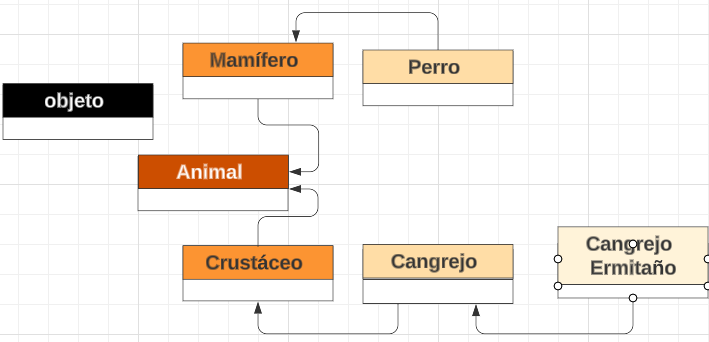
return roofType;

}

}

Punto 7



Punto 8.