|  |  |
| --- | --- |
| 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. |

* En eclipse llamado PuntoUno y en Main se ejecuta.

2.



3. En el eclipse Person y Student

4.

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.

5. En eclipse Instrument, StringInstrument y Guitar

6.



**Las palabras rojas son las que se completaban**

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;

}

}

7.



8.



Respuestas del Codigo

1.package ActividadDiez;

public class Guitar extends StringInstrument {

private boolean isElectric;

public boolean getIsElectric() {

return isElectric;

}

public void setIsElectric(boolean isElectric) {

this.isElectric = isElectric;

}

}

2.package ActividadDiez;

public class Instrument {

private boolean onSale;

private double price;

private int numInStock;

public double getPrice() {

if (onSale) {

return price \* 0.85; // precio con descuento del 15%

} else {

return price;

}

}

public double applyEmployeeDiscount() {

return price \* 0.75; // precio con descuento del 25%

}

public void setOnSale(boolean onSale) {

this.onSale = onSale;

}

public boolean getOnSale() {

return onSale;

}

public void setPrice(double price) {

this.price = price;

}

public int getNumInStock() {

return numInStock;

}

public void setNumInStock(int numInStock) {

this.numInStock = numInStock;

}

}

3.package ActividadDiez;

import javax.swing.JFrame;

import javax.swing.SwingUtilities;

public class Main {

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

JFrame frame = new JFrame();

PuntoUno applet = new PuntoUno();

frame.getContentPane().add(applet);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

applet.init();

applet.start();

}

});

}

}

4.package ActividadDiez;

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;

}

public String getFirstName(){

return firstName;

}

public String getMiddleName(){

return middleName;

}

public String getLastName(){

return lastName;

}

public String getName(){

return firstName + " " + middleName + " " + lastName;

}

public Date getDateOfBirth(){

return dateOfBirth;

}

}

5.package ActividadDiez;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

public class PuntoUno extends JApplet implements ActionListener {

private static final long serialVersionUID = 1L;

private Font font;

private Color redColor;

private Color blueColor;

private Color backgroundColor;

private JButton button;

private boolean isBlue = false;

public void init() {

font = new Font("Arial", Font.ITALIC, 18);

redColor = Color.red;

backgroundColor = Color.orange;

blueColor = new Color(0, 0, 122);

setBackground(backgroundColor);

button = new JButton("Cambiar color");

// Agregar el botón al applet

add(button);

button.addActionListener(this);

}

\* Este método dibuja las formas en la pantalla

public void paint(Graphics graph) {

super.paint(graph);

graph.setFont(font);

graph.drawString("Dibujar formas", 90, 20);

if (isBlue) {

graph.setColor(blueColor);

} else {

graph.setColor(redColor);

}

graph.drawRect(120, 120, 120, 120);

graph.fillRect(115, 115, 90, 90);

graph.setColor(Color.CYAN);

graph.drawRect(50, 50, 50, 50);

graph.fillRect(50, 50, 60, 60);

// Establecer el color para la siguiente forma

if (isBlue) {

graph.setColor(redColor);

} else {

graph.setColor(blueColor);

}

graph.fillArc(110, 110, 50, 50, 0, 360);

}

public void actionPerformed(ActionEvent e)

{ isBlue = !isBlue; repaint();

}

}

6.package ActividadDiez;

public class StringInstrument extends Instrument {

private int numStrings;

public int getNumStrings() {

return numStrings;

}

public void setNumStrings(int numStrings) {

this.numStrings = numStrings;

}

}

7.package ActividadDiez;

import java.util.Date;

public class Student extends Person {

private int id;

private String major;

private String degree;

private int gradYear;

private double gpa;

public Student(String firstName, String middleName, String lastName, Date dateOfBirth, int id, String major, String degree, int gradYear, double gpa) {

super(firstName, middleName, lastName, dateOfBirth);

this.id = id;

this.major = major;

this.degree = degree;

this.gradYear = gradYear;

this.gpa = gpa;

}

public int getId() {

return id;

}

public String getMajor() {

return major;

}

public String getDegree() {

return degree;

}

public int getGradYear() {

return gradYear;

}

public double getGpa() {

return gpa;

}

public void setMajor(String major) {

this.major = major;

}

public double calculateGPA(String[] grades) {

double totalPoints = 0;

int numCourses = grades.length;

for (int i = 0; i < numCourses; i++) {

switch (grades[i]) {

case "4":

totalPoints += 4.0;

break;

case "3.67":

totalPoints += 3.67;

break;

case "3.33":

totalPoints += 3.33;

break;

case "3":

totalPoints += 3.0;

break;

case "2.67":

totalPoints += 2.67;

break;

case "2.33":

totalPoints += 2.33;

break;

case "2":

totalPoints += 2.0;

break;

case "1":

totalPoints += 1.0;

break;

case "0":

totalPoints += 0.0;

break;

default:

System.out.println("Calificación inválida");

break;

}

}

return totalPoints / numCourses;

}

}