# Ejercicio 4.1 Herencia

## CUENTA Cuenta

public class Cuenta { protected float saldo; protected int numeroConsignaciones = 0; protected int numeroRetiros = 0; protected float tasaAnual; protected float comisionMensual = 0;

public Cuenta(float saldo, float tasaAnual) { this.saldo = saldo; this.tasaAnual = tasaAnual;

}

public void consignar(float cantidad) { saldo = saldo + cantidad; numeroConsignaciones = numeroConsignaciones + 1;

}

public void retirar(float cantidad) { float nuevoSaldo = saldo - cantidad;

if (nuevoSaldo >= 0) { saldo -= cantidad;

numeroRetiros = numeroRetiros + 1;

} else {

System.out.println("La cantidad a retirar excede el saldo actual. ");

}

}

public void calcularInteres() { float tasaMensual = tasaAnual / 12; float interesMensual = saldo \* tasaMensual; saldo += interesMensual;

}

public void extractoMensual() { saldo -= comisionMensual; calcularInteres();

}

}

## CuentaAhorros

public class CuentaAhorros extends Cuenta {

private boolean activa;

public CuentaAhorros(float saldo, float tasa) { super(saldo,tasa); if (saldo < 10000) activa = false; else activa = true;

}

public void retirar(float cantidad) {

if (activa) super.retirar(cantidad);

}

public void consignar (float cantidad) {

if (activa)

super.consignar(cantidad);

}

public void extractoMensual() { if (numeroRetiros > 4) { comisionMensual += (numeroRetiros - 4) \* 1000;

}

super.extractoMensual(); if (saldo < 10000) activa = false;

}

public void imprimir() {

System.out.println("Saldo = $ " + saldo);

System.out.println("Comisión mensual = $ " + comisionMensual);

System.out.println("Número de transaciones = " + (numeroConsignaciones + numeroRetiros)); System.out.println();

}

}

## CuentaCorriente

public class CuentaCorriente extends Cuenta {

float sobregiro;

public CuentaCorriente(float saldo, float tasa) { super(saldo, tasa); sobregiro = 0;

}

public void retirar(float cantidad) { float resultado = saldo - cantidad; if (resultado < 0) { sobregiro = sobregiro - resultado; saldo = 0; } else { super.retirar(cantidad);

}

}

public void consignar (float cantidad) { float residuo = sobregiro - cantidad; if (sobregiro > 0) { if (residuo > 0) { sobregiro = 0; saldo = residuo; } else { sobregiro = -residuo;

saldo = 0;

} } else {

super.consignar(cantidad);

}

}

public void extractoMensual() { super.extractoMensual();

}

public void imprimir() {

System.out.println("Saldo = $ " + saldo);

System.out.println("Cargo mensual = $ " + comisionMensual);

System.out.println("Número de transaciones = " + (numeroConsignaciones + numeroRetiros));

System.out.println("Valor de sobregiro = $" + (numeroConsignaciones + numeroRetiros)); System.out.println();

}

}

## PruebaCuenta

import java.util.\*;

public class PruebaCuenta {

public static void main(String args[]) {

Scanner input = new Scanner(System.in);

System.out.println("Cuenta de ahorros"); System.out.println("Ingrese saldo inicial= $"); float saldoInicialAhorros = input.nextFloat(); System.out.print("Ingrese tasa de interés= "); float tasaAhorros = input.nextFloat();

CuentaAhorros cuenta1 = new

CuentaAhorros(saldoInicialAhorros, tasaAhorros);

System.out.print("Ingresar cantidad a consignar: $");

float cantidadDepositar = input.nextFloat(); cuenta1.consignar(cantidadDepositar);

System.out.print("Ingresar cantidad a retirar: $"); float cantidadRetirar = input.nextFloat(); cuenta1.retirar(cantidadRetirar); cuenta1.extractoMensual(); cuenta1.imprimir();

}

}

# Ejercicio 4.2 Paquetes y métodos de acceso

## INMUEBLES ApartaEstudio package Inmuebles;

public class ApartaEstudio extends Apartamento {

protected static double valorArea = 1500000;

public ApartaEstudio(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños) {

super(identificadorInmobiliario, area, direccion, 1, 1);

}

void imprimir() { super.imprimir();

System.out.println();

}

}

**Apartamento** package Inmuebles;

public class Apartamento extends InmuebleVivienda {

public Apartamento(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños) { super(identificadorInmobiliario, area, direccion, numeroHabitaciones, numeroBaños);

}

void imprimir() { super.imprimir();

}

}

## ApartamentoFamiliar package Inmuebles;

public class ApartamentoFamiliar extends Apartamento { protected static double valorArea = 2000000; protected int valorAdministracion;

public ApartamentoFamiliar(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños, int valorAdministracion) { super(identificadorInmobiliario, area, direccion, numeroHabitaciones, numeroBaños); this.valorAdministracion = valorAdministracion;

}

void imprimir() { super.imprimir(); System.out.println("Valor de la administración = $" + valorAdministracion); System.out.println();

}

}

**Casa** package Inmuebles;

public class Casa extends InmuebleVivienda {

protected int numeroPisos;

public Casa(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños, int numeroPisos) {

// Invoca al constructor de la clase padre super(identificadorInmobiliario, area, direccion, numeroHabitaciones, numeroBaños); this.numeroPisos = numeroPisos;

}

void imprimir() { super.imprimir(); // Invoca al método imprimir de la clase padre System.out.println("Número de pisos = " + numeroPisos);

}

}

## CasaConjuntoCerrado package Inmuebles;

public class CasaConjuntoCerrado extends CasaUrbana { protected static double valorArea = 2500000; protected int valorAdministracion; protected boolean tienePiscina;

protected boolean tieneCamposDeportivos;

public CasaConjuntoCerrado(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños, int numeroPisos, int valorAdministracion, boolean tienePiscina, boolean tieneCamposDeportivos) { super(identificadorInmobiliario, area, direccion, numeroHabitaciones, numeroBaños, numeroPisos); this.valorAdministracion = valorAdministracion; this.tienePiscina = tienePiscina;

this.tieneCamposDeportivos = tieneCamposDeportivos;

}

void imprimir() { super.imprimir();

System.out.println("Valor de la administración = " + valorAdministracion);

System.out.println("Tiene piscina? = " + tienePiscina);

System.out.println("Tiene campos deportivos? = " + tieneCamposDeportivos); System.out.println();

}

}

**CasaIndependiente** package Inmuebles;

public class CasaIndependiente extends CasaUrbana { protected static double valorArea = 3000000;

public CasaIndependiente(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños, int numeroPisos) { super(identificadorInmobiliario, area, direccion, numeroHabitaciones, numeroBaños, numeroPisos);

}

void imprimir() { super.imprimir();

System.out.println();

}

}

**CasaRural** package Inmuebles;

public class CasaRural extends Casa {

protected static double valorArea = 1500000; protected int distanciaCabera; protected int altitud;

public CasaRural(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños, int numeroPisos, int distanciaCabera, int altitud) {

super(identificadorInmobiliario, area, direccion,numeroHabitaciones, numeroBaños, numeroPisos); this.distanciaCabera = distanciaCabera; this.altitud = altitud;

}

void imprimir() { super.imprimir(); // Invoca al método imprimir de la clase padre

System.out.println("Distancia la cabecera municipal = " + numeroHabitaciones + " km.");

System.out.println("Altitud sobre el nivel del mar = " + altitud + " metros"); System.out.println();

}

}

**CasaUrbana** package Inmuebles;

public class CasaUrbana extends Casa {

public CasaUrbana(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños, int numeroPisos) {

super(identificadorInmobiliario, area, direccion, numeroHabitaciones, numeroBaños, numeroPisos);

}

void imprimir() { super.imprimir();

}

}

**Inmueble** package Inmuebles;

public class Inmueble { protected int identificadorInmobiliario; protected int area; protected String direccion; protected double precioVenta;

Inmueble(int identificadorInmobiliario, int area, String direccion) { this.identificadorInmobiliario = identificadorInmobiliario; this.area = area; this.direccion = direccion;

}

double calcularPrecioVenta(double valorArea) { precioVenta = area \* valorArea; return precioVenta;

}

void imprimir() {

System.out.println("Identificador inmobiliario = " + identificadorInmobiliario); System.out.println("Area = " + area);

System.out.println("Dirección = " + direccion);

System.out.println("Precio de venta = $" + precioVenta);

}

}

**InmuebleVivienda** package Inmuebles;

public class InmuebleVivienda extends Inmueble { protected int numeroHabitaciones; protected int numeroBaños;

public InmuebleVivienda(int identificadorInmobiliario, int area, String direccion, int numeroHabitaciones, int numeroBaños) {

super(identificadorInmobiliario, area, direccion); this.numeroHabitaciones = numeroHabitaciones; this.numeroBaños = numeroBaños;

}

void imprimir() { super.imprimir(); // Invoca al método imprimir de la clase padre

System.out.println("Número de habitaciones = " + numeroHabitaciones); System.out.println("Número de baños = " + numeroBaños);

}

}

**Local** package Inmuebles;

public class Local extends Inmueble { enum tipo {INTERNO,CALLE }; protected tipo tipoLocal;

public Local(int identificadorInmobiliario, int area, String direccion, tipo tipoLocal) { super(identificadorInmobiliario, area, direccion); this.tipoLocal = tipoLocal;

}

void imprimir() { super.imprimir();

System.out.println("Tipo de local = " + tipoLocal);

}

}

**LocalComercial** package Inmuebles;

public class LocalComercial extends Local { protected static double valorArea = 3000000; protected String centroComercial;

public LocalComercial(int identificadorInmobiliario, int area, String direccion, tipo tipoLocal, String centroComercial) { super(identificadorInmobiliario, area, direccion, tipoLocal); this.centroComercial = centroComercial;

}

void imprimir() { super.imprimir();

System.out.println("Centro comercial = " + centroComercial); System.out.println();

}

}

**Oficina** package Inmuebles;

public class Oficina extends Local { protected static double valorArea = 3500000; protected boolean esGobierno; public Oficina(int identificadorInmobiliario, int area, String direccion, tipo tipoLocal, boolean esGobierno) { super(identificadorInmobiliario, area, direccion, tipoLocal); this.esGobierno = esGobierno;

}

void imprimir() { super.imprimir();

System.out.println("Es oficina gubernamental = " + esGobierno); System.out.println();

}

}

**Prueba** package Inmuebles;

public class Prueba { public static void main(String args[]) {

ApartamentoFamiliar apto1 = new ApartamentoFamiliar(103067,120,"Avenida Santander 45-45",3,2,200000); System.out.println("Datos apartamento"); apto1.calcularPrecioVenta(ApartamentoFamiliar.valorArea);

apto1.imprimir();

System.out.println("Datos apartamento");

ApartaEstudio aptestudio1 = new ApartaEstudio(12354, 50 ,"Avenida Caracas 30-15", 1 , 1); aptestudio1.calcularPrecioVenta(ApartaEstudio.valorArea); aptestudio1.imprimir();

}

}

# Ejercicio 4.7 Clases abstractas

**ANIMALS Animal** package Animals;

public abstract class Animal {

protected String sonido; protected String alimentos; protected String hábitat; protected String nombreCientifico; public abstract String getNombreCientífico(); public abstract String getSonido(); public abstract String getAlimentos(); public abstract String getHábitat();

}

**Canido** package Animals;

public abstract class Canido extends Animal {

}

**Felino** package Animals; public abstract class Felino extends Animal {

}

## Gato

package Animals;

public class Gato extends Felino {

public String getSonido() { return "Maullido";

}

public String getAlimentos() { return "Ratones";

}

public String getHábitat() { return "Domestico";

}

public String getNombreCientífico() { return "Felis silvestris catus";

}

}

**Leon** package Animals; public class Leon extends Felino {

public String getSonido() { return "Rugido";

}

public String getAlimentos() { return "Carnivoro";

}

public String getHábitat() { return "Praderas";

}

public String getNombreCientífico() { return "Panthera leo";

}

}

**Lobo** package Animals;

public class Lobo extends Canido{

public String getSonido() { return "Aullido";

}

public String getAlimentos() { return "Carnivoro";

}

public String getHábitat() { return "Bosque";

}

public String getNombreCientífico() { return "Canis lupus";

}

}

**Perro** package Animals;

public class Perro extends Canido{

public String getSonido() { return "Ladrido";

}

public String getAlimentos() { return "Carnivoro";

}

public String getHábitat() { return "Domestico";

}

public String getNombreCientífico() { return "Canis lupus familiaris";

}

}

**Prueba** package Animals; public class Prueba { public static void main(String[] args) { Animal[] animales = new Animal[4]; animales[0] = new Gato(); animales[1] = new Perro(); animales[2] = new Lobo(); animales[3] = new Leon();

for (int i = 0; i < animales.length; i++) {

System.out.println(animales[i].getNombreCientífico()); System.out.println("Sonido: " + animales[i].getSonido());

System.out.println("Alimentos: " + animales[i]. getAlimentos());

System.out.println("Habitat: " + animales[i].getHábitat());

System.out.println();

}

}

}

# Ejercicio 4.8 Métodos abstractos

## CARRERACICLISTICA

**Ciclista** package CarreraCiclistica;

public abstract class Ciclista { private int identificador; private String nombre; private int tiempoAcumulado = 0;

public Ciclista(int identificador, String nombre) { this.identificador = identificador; this.nombre = nombre;

}

protected String getNombre() { return nombre;

}

protected void setNombre(String nombre) { this.nombre = nombre;

}

protected int getPosiciónGeneral(int posiciónGeneral) { return posiciónGeneral;

}

protected void setPosiciónGeneral(int posiciónGeneral) { posiciónGeneral = posiciónGeneral;

}

protected int getTiempoAcumulado() { return tiempoAcumulado;

}

protected void setTiempoAcumulado(int tiempoAcumulado) { this.tiempoAcumulado = tiempoAcumulado;

}

protected void imprimir() {

System.out.println("Identificador = " + identificador);

System.out.println("Nombre = " + nombre);

System.out.println("Tiempo Acumulado = " + tiempoAcumulado);

}

}

**Contrarrelojista** package CarreraCiclistica;

public class Contrarrelojista extends Ciclista { private double velocidadMáxima;

public Contrarrelojista(int identificador, String nombre, double velocidadMáxima) { super(identificador, nombre); this.velocidadMáxima = velocidadMáxima;

}

protected double getVelocidadMáxima() { return velocidadMáxima;

}

protected void setVelocidadMáxima(double velocidadMáxima) { this.velocidadMáxima = velocidadMáxima;

}

protected void imprimir() { super.imprimir();

System.out.println("Aceleración promedio = " + velocidadMáxima);

}

protected String imprimirTipo() { return "Es un constrarrelojista";

}

}

**Equipo** package CarreraCiclistica; import java.util.\*;

public class Equipo { private String nombre; private static double totalTiempo; private String país;

Vector listaCiclistas;

public Equipo(String nombre, String país) { this.nombre = nombre; this.país = país; totalTiempo = 0; listaCiclistas = new Vector();

}

public String getNombre() { return nombre;

}

public void setNombre(String nombre) { this.nombre = nombre;

}

private String getPaís() { return país;

}

private void setPaís(String país) {

this.país = país;

}

void añadirCiclista(Ciclista ciclista) { listaCiclistas.add(ciclista); // Se agrega el ciclista al vector de ciclistas

}

void imprimir() {

System.out.println("Nombre del equipo = " + nombre);

System.out.println("País = " + país);

System.out.println("Total tiempo del equipo = " + totalTiempo);

}

void listarEquipo() { for (int i = 0; i < listaCiclistas.size(); i++) {

Ciclista c = (Ciclista) listaCiclistas.elementAt(i);

System.out.println(c.getNombre());

}

}

void calcularTotalTiempo() { for (int i = 0; i < listaCiclistas.size(); i++) { Ciclista c = (Ciclista) listaCiclistas.elementAt(i); totalTiempo = totalTiempo + c.getTiempoAcumulado();

}

}

}

**Escalador** package CarreraCiclistica;

public class Escalador extends Ciclista {

private double aceleraciónPromedio; private double gradoRampa; public Escalador(int identificador, String nombre, double aceleraciónPromedio, double gradoRampa) { super(identificador, nombre); this.aceleraciónPromedio = aceleraciónPromedio; this.gradoRampa = gradoRampa;

}

protected double getAceleraciónPromedio() { return aceleraciónPromedio;

}

protected void setAceleraciónPromedio(double aceleraciónPromedio) { this.aceleraciónPromedio = aceleraciónPromedio;

}

protected double getGradoRampa() { return gradoRampa;

}

protected void setGradoRampa(double gradoRampa) { this.gradoRampa = gradoRampa;

}

protected void imprimir() {

super.imprimir();

System.out.println("Aceleración promedio = " + aceleraciónPromedio); System.out.println("Grado de rampa = " + gradoRampa);

}

protected String imprimirTipo() { return "Es un escalador";

}

}

**Velocista** package CarreraCiclistica;

public class Velocista extends Ciclista { private double potenciaPromedio; private double velocidadPromedio; public Velocista(int identificador, String nombre, double potenciaPromedio, double velocidadPromedio) { super(identificador, nombre); potenciaPromedio = potenciaPromedio; this.velocidadPromedio = velocidadPromedio;

}

protected double getPotenciaPromedio() { return potenciaPromedio;

}

protected void setPotenciaPromedio(double potenciaPromedio) { this.potenciaPromedio = potenciaPromedio;

}

protected double getvelocidadPromedio() { return velocidadPromedio;

}

protected void setVelocidadPromedio(double velocidadPromedio) {

this.velocidadPromedio = velocidadPromedio;

}

protected void imprimir() { super.imprimir(); // Invoca al método imprimir de la clase padre

System.out.println("Potencia promedio = " + potenciaPromedio);

System.out.println("Velocidad promedio = " + velocidadPromedio);

}

protected String imprimirTipo() { return "Es un velocista";

}

}

**Prueba** package CarreraCiclistica;

public class Prueba {

public static void main(String args[]) {

Equipo equipo1 = new Equipo("Sky","Estados Unidos");

Velocista velocista1 = new Velocista(123979, "Geraint Thomas", 320, 25);

Escalador escalador1 = new Escalador(123980, "Egan Bernal", 25, 10);

Contrarrelojista contrarrelojista1 = new Contrarrelojista(123981, "Jonathan Castroviejo", 120); equipo1.añadirCiclista(velocista1); equipo1.añadirCiclista(escalador1); equipo1.añadirCiclista(contrarrelojista1); velocista1.setTiempoAcumulado(365); escalador1.setTiempoAcumulado(385); contrarrelojista1.setTiempoAcumulado(370); equipo1.calcularTotalTiempo(); equipo1.imprimir(); equipo1.listarEquipo();

}

}

# Ejercicio 8.1

**PERSONAS**

## MainWindow

package Personas;

import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import javax.swing.\*;

public class MainWindow extends JFrame implements ActionListener{

private PeopleList lista; // El objeto ListaPersonas de la aplicación private Container contenedor; /\* Un contenedor de elementos

gráficos \*/

// Etiquetas estáticas para los nombres de los atributos private JLabel nombre, apellidos, teléfono, dirección;

// Campos de ingreso de texto private JTextField campoNombre, campoApellidos, campoTeléfono, campoDirección; private JButton añadir, eliminar, borrarLista; // Botones private JList listaNombres; // Lista de personas private DefaultListModel modelo; // Objeto que modela la lista private JScrollPane scrollLista; // Barra de desplazamiento vertical

public MainWindow(){ lista = new PeopleList(); // Crea la lista de personas

inicio();

setTitle("Personas"); // Establece el título de la ventana setSize(270,350); // Establece el tamaño de la ventana setLocationRelativeTo(null); /\* La ventana se posiciona en el centro de la pantalla \*/

// Establece que el botón de cerrar permitirá salir de la aplicación setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setResizable(false); /\* Establece que el tamaño de la ventana no se puede cambiar \*/

}

private void inicio() { contenedor = getContentPane(); /\* Obtiene el panel de contenidos de la ventana \*/ contenedor.setLayout(null); /\* Establece que el panel no tiene asociado ningún layout \*/

// Establece la etiqueta y el campo nombre nombre = new JLabel(); nombre.setText("Nombre:"); nombre.setBounds(20, 20, 135, 23); /\* Establece la posición de la etiqueta nombre \*/ campoNombre = new JTextField();

// Establece la posición del campo de texto nombre campoNombre.setBounds(105, 20, 135, 23); // Establece la etiqueta y el campo apellidos apellidos = new JLabel();

apellidos.setText("Apellidos:"); /\* Establece la posición de la etiqueta apellidos \*/ apellidos.setBounds(20, 50, 135, 23); campoApellidos = new JTextField();

// Establece la posición del campo de texto apellidos campoApellidos.setBounds(105, 50, 135, 23);

// Establece la etiqueta y el campo teléfono teléfono = new JLabel(); teléfono.setText("Telefono:"); teléfono.setBounds(20, 80, 135, 23); /\* Establece la posición de la etiqueta teléfono \*/ campoTeléfono = new JTextField();

// Establece la posición del campo de texto teléfono campoTeléfono.setBounds(105, 80, 135, 23); // Establece la etiqueta y el campo dirección dirección = new JLabel(); dirección.setText("Direccion:"); dirección.setBounds(20, 110, 135, 23); /\* Establece la posición de la etiqueta dirección \*/ campoDirección = new JTextField();

// Establece la posición del campo de texto dirección campoDirección.setBounds(105, 110, 135, 23); // Establece el botón Añadir persona añadir = new JButton(); añadir.setText("«Añadir»"); añadir.setBounds(105, 150, 80, 23); /\* Establece la posición del botón Añadir persona \*/

/\* Agrega al botón un ActionListener para que gestione eventos

del botón \*/ añadir.addActionListener(this);

// Establece el botón Eliminar persona eliminar= new JButton(); eliminar.setText("Eliminar"); eliminar.setBounds(20, 280, 80, 23); /\* Establece la posición del botón Eliminar persona \*/

/\* Agrega al botón un ActionListener para que gestione eventos

del botón \*/ eliminar.addActionListener(this); // Establece el botón Borrar lista borrarLista= new JButton(); borrarLista.setText("Borrar Lista"); borrarLista.setBounds(120, 280, 120, 23); /\* Establece la posición del botón Borrar lista \*/

/\* Agrega al botón un ActionListener para que gestione eventos del botón \*/ borrarLista.addActionListener(this); // Establece la lista gráfica de personas listaNombres = new JList();

/\* Establece que se pueda seleccionar solamente un elemento de

la lista \*/

listaNombres.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION); modelo = new DefaultListModel();

// Establece una barra de desplazamiento vertical scrollLista = new JScrollPane();

// Establece la posición de la barra de desplazamiento vertical scrollLista.setBounds(20, 190 ,220, 80);

// Asocia la barra de desplazamiento vertical a la lista de personas scrollLista.setViewportView(listaNombres);

// Se añade cada componente gráfico al contenedor de la ventana contenedor.add(nombre);

contenedor.add(campoNombre); contenedor.add(apellidos); contenedor.add(campoApellidos); contenedor.add(teléfono); contenedor.add(campoTeléfono); contenedor.add(dirección); contenedor.add(campoDirección); contenedor.add(añadir); contenedor.add(eliminar); contenedor.add(borrarLista); contenedor.add(scrollLista);

}

public void actionPerformed(ActionEvent evento) { if (evento.getSource() == añadir) { // Si se pulsa el botón añadir anadirPersona(); // Se invoca añadir persona

}

if (evento.getSource() == eliminar) { /\* Si se pulsa el botón eliminar \*/

/\* Se invoca el método eliminarNombre que elimina el elemento seleccionado \*/ eliminarNombre(listaNombres.getSelectedIndex() );

}

if (evento.getSource() == borrarLista) { /\* Si se pulsa el botón borrar lista \*/ lista.borrarLista(); // Se invoca borrar lista

}

}

private void anadirPersona(){

Person p = new Person(campoNombre.getText(),

campoApellidos.getText(), campoTeléfono.getText(), campoDirección.getText()); lista.anadirPersona(p);

String elemento = campoNombre.getText() + "-" +

campoApellidos.getText() + "-" + campoTeléfono.getText() + "-" + campoDirección.getText(); modelo.addElement(elemento); listaNombres.setModel(modelo);

// Se colocan todos los campos de texto nulos campoNombre.setText(""); campoApellidos.setText(""); campoTeléfono.setText(""); campoDirección.setText("");

}

private void eliminarNombre(int indice) { if (indice >= 0) { // Si la posición existe modelo.removeElementAt(indice); /\* Se elimina la persona seleccionada de la lista gráfica \*/

}

}

private void borrarLista() {

lista.borrarLista(); // Se eliminan todas las personas del vector

modelo.clear(); // Limpia el JList, la lista gráfica de personas

}

}

**PeopleList** package Personas;

import java.util.\*; public class PeopleList {

Vector listaPersonas; // Atributo que identifica un vector de personas public PeopleList() { listaPersonas = new Vector(); // Crea el vector de personas

}

public void anadirPersona(Person p) { listaPersonas.add(p);

}

public void eliminarPersona(int i) { listaPersonas.removeElementAt(i);

}

public void borrarLista() {

listaPersonas.removeAllElements();

}

}

**Person** package Personas;

public class Person {

String nombre; // Atributo que identifica el nombre de una persona String apellidos; /\* Atributo que identifica los apellidos de una persona \*/

String teléfono; // Atributo que identifica el teléfono de una persona String dirección; /\* Atributo que identifica la dirección de una persona \*/ public Person(String nombre, String apellidos, String teléfono, String dirección) { this.nombre = nombre; this.apellidos = apellidos; this.teléfono = teléfono; this.dirección = dirección;

}

}

**Principal** package Personas;

public class Principal {

public static void main(String[] args) {

MainWindow miVentanaPrincipal; /\* Define la ventana

principal \*/

miVentanaPrincipal = new MainWindow(); /\* Crea la ventana principal \*/ miVentanaPrincipal.setVisible(true); /\* Establece la ventana como visible \*/

}

}

Link video youtube: <https://youtu.be/gM6vyLa7WyU>

DIAGRAM PACKAGE CUENTABANCO

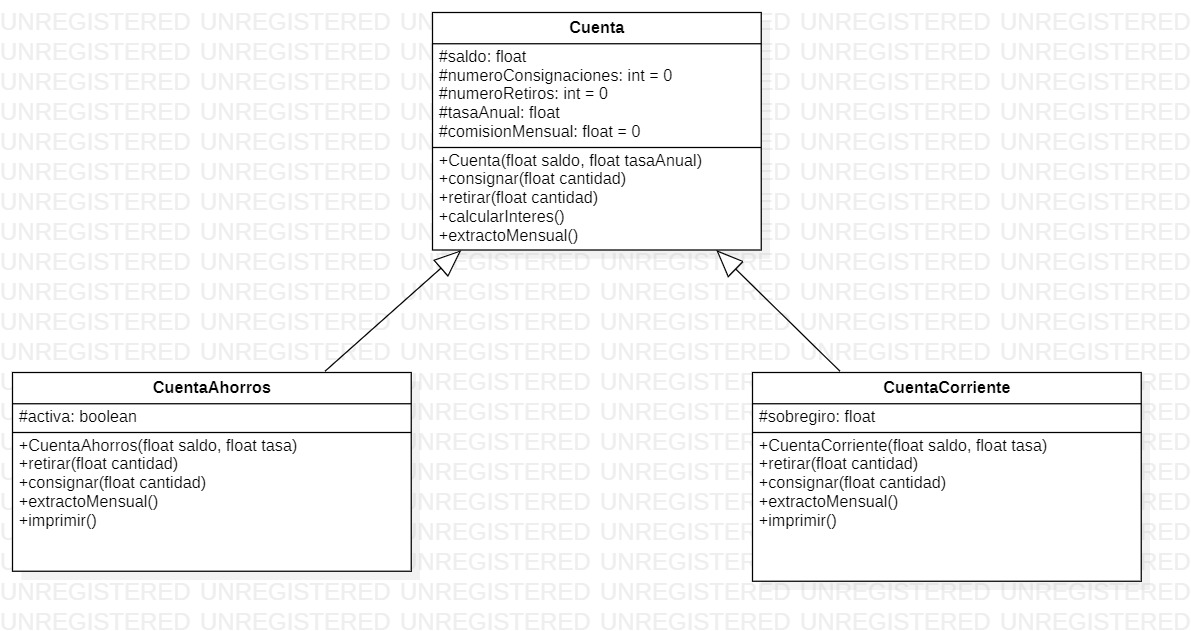


DIAGRAM PACKAGE INMUEBLES

Diagrama

Descripción generada automáticamente

DIAGRAM PACKAGE ANIMALS

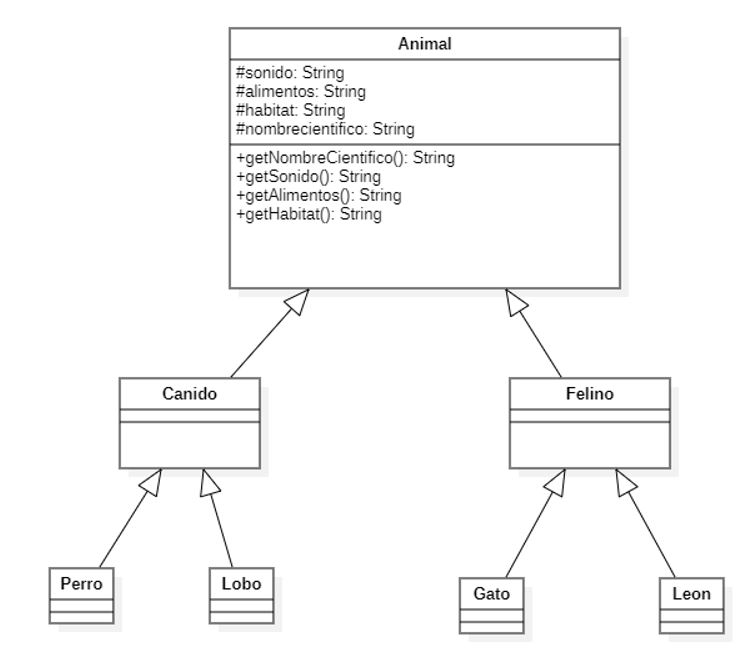


DIAGRAM PACKAGE CARRERACICLISTA

Diagrama

Descripción generada automáticamente

DIAGRAM PACKAGE PERSONAS

Diagrama

Descripción generada automáticamente