**EJERCICIO BANCO**

**// ---- PROGRAMAR EL RESTO DE LOS MÉTODOS ----**

**import java.util.Scanner;**

**public class CuentaBancaria {**

**Scanner sc = new Scanner(System.*in*);**

**// Atributos de Clase**

**private int saldo; // Saldo actual de la cuenta**

**private int numMovimientos; // Número de movimientos realizados**

**private static int *numCuentas* = 0; // Número de cuentas creadas**

**// METODOS**

**// --METODOS ESTÁTICOS (CLASE)----------------------------**

**public static int totalCuentas(){**

**return *numCuentas*;**

**}**

**// --METODOS DE INSTANCIA (OBJETOS) ----------------------------**

**// Constructores**

**public CuentaBancaria(int saldo){**

**// Atributo de instancia (objeto)**

**this.saldo = saldo;**

**this.numMovimientos = 0;**

**// Atributo de clase**

**CuentaBancaria.*numCuentas*++;**

**public CuentaBancaria(){**

**this(0); // Llamo al primer constructor**

**}**

**// Resto de los MÉTODOS**

**//Ingreso, incrementa el saldo en una cantidad indicada como parámetro.**

**public void ingreso (int cantidad){**

**if (cantidad > 0) {**

**saldo = saldo + cantidad;**

**numMovimientos++;**

**}**

**}**

**// Abono, decremento el saldo en la cantidad indicada como parámetro.**

**public void abono (int cantidad){**

**// Verifica si la cantidad es positiva y menor o igual al saldo**

**if (cantidad > 0 && cantidad <= saldo) {**

**saldo = saldo - cantidad; // Decrementa el saldo**

**numMovimientos++;**

**}**

**}**

**// Anotar gastos decrementa el saldo en 20 euros si**

**// el saldo de la cuenta es menor 1000**

**public void anotarGastos(){**

**if (saldo < 1000) {**

**saldo = saldo - 20;**

**numMovimientos++;**

**}**

**}**

**// Anotar Intereses incrementa la cuenta según valor de interés indicado**

**// como parámetro en tanto por ciento.**

**public void anotarIntereses (int interes){**

**if (interes >= 0) {**

**saldo = saldo + (saldo \* interes) / 100;**

**}**

**}**

**//Realizar transferencia a cuenta, decrementa el saldo**

**// en la cantidad indicada**

**// como parámetro, realizando un ingreso en la cuenta destino.**

**public void transferencia ( int importe, CuentaBancaria destino){**

**if (importe > 0 && importe <= saldo ) {**

**this.abono(importe);**

**destino.ingreso(importe);**

**numMovimientos++;**

**}**

**}**

**// Consultar estado de la cuenta, mostrá el saldo actual y**

**// el número de operaciones realizadas**

**public String consultarEstado (){return " Saldo = "+ saldo + " Nº operaciones = "+ numMovimientos;}}**

**public class TestCuentaBancaria {**

**public static void main ( String argv[]){**

**CuentaBancaria c1 = new CuentaBancaria(100);**

**CuentaBancaria c2 = new CuentaBancaria(1900);**

**CuentaBancaria c3 = new CuentaBancaria();**

**System.*out*.println(" N.º de Cuentas="+CuentaBancaria.*totalCuentas*());**

**c1.abono(20);**

**c1.ingreso(10);**

**c1.anotarGastos();**

**System.*out*.println(" Cuenta c1 = "+c1.consultarEstado());**

**c2.ingreso(100);**

**c2.anotarGastos(); // No se aplican pues su saldo es mayor que 1000**

**c2.anotarIntereses(5); // 5% de interes**

**c2.transferencia(100,c3);**

**System.*out*.println(" Cuenta c2 = "+c2.consultarEstado());**

**c3.abono(75);**

**c3.abono(75);**

**System.*out*.println(" Cuenta c3 = "+c3.consultarEstado());**

**}**

**}**

**public class Actor extends Persona implements Actuacion**

**{**

**private String pelicula;**

**private String papel;**

**public Actor ( String nombre, String pelicula, String papel){**

**super(nombre);**

**this.pelicula = pelicula;**

**this.papel = papel;**

**}**

**public String decirAlgo() {**

**return papel;**

**}**

**public String getPelicula() {**

**return pelicula;**

**}**

**public String toString() {**

**return "Nombre: "+ super.toString() + ", Pelicula: " + pelicula + ", Papel: " + papel;**

**}**

**public static void main ( String arg [] ) {**

**Actor a = new Actor("Carmen", "El resplandor", "Pasame el hacha");**

**System.*out*.println(a);**

**}**

**}**

**public interface Actuacion**

**{**

**// Genera un String con el papel de una obra de teatro o película**

**public String decirAlgo();**

**}**

**public class Peliculas**

**{**

**public static void main (String arg []) {**

**Actor reparto [] = new Actor [4];**

**reparto[0] = new Actor("Eva","Supermar","Volando voy");**

**reparto[1] = new Actor("Daniel","Romeo y Julieta","Que bella");**

**reparto[2] = new Actor("Teresa","Romeo y Julieta","Tu eres feisima");**

**reparto[3] = new Actor("Juan","Supermar","Callate un rato");**

***mostrarReparto*(reparto, "Romeo y Julieta");**

**}**

**// Imprime los nombres de los actores que tienen asignada la película pasada como parámetro**

**// y el mensaje con papel que tiene asignado**

**public static void mostrarReparto(Actor lista[],String pelicula){**

**for (Actor a: lista) {**

**if (pelicula.equals(a.getPelicula()) ) {**

**System.*out*.println(a.toString());**

**System.*out*.println(a.decirAlgo());**

**}**

**}**

**}**

**}**

**public abstract class Persona**

**{**

**private String nombre;**

**public Persona( String nombre){**

**this.nombre = nombre;**

**}**

**public String toString() {**

**return nombre;**

**}**

**}**

**AHORCADO**

**import java.util.Scanner;**

**public class JuegoCht {**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.*in*);**

**// Pedir la palabra o frase a adivinar**

**System.*out*.print("Introduce una película: ");**

**String pelicula = scanner.nextLine().toUpperCase();**

**// Inicializar variables**

**String peliculaOculta = pelicula.replaceAll("[A-Z]", "-");**

**String ahorcado = "AHORCADO";**

**int fallos = 0;**

**final int MAX\_FALLOS = ahorcado.length();**

**System.*out*.println("Película a adivinar: " + peliculaOculta);**

**// Bucle principal del juego**

**while (fallos < MAX\_FALLOS && peliculaOculta.contains("-")) {**

**System.*out*.print("Introduce una letra: ");**

**String letra = scanner.nextLine().toUpperCase();**

**if (letra.length() != 1 || !letra.matches("[A-Z]")) {**

**System.*out*.println("Por favor, introduce una sola letra válida.");**

**continue;**

**}**

**if (pelicula.contains(letra)) {**

**// Reemplazar los guiones por la letra adivinada**

**StringBuilder nuevaPeliculaOculta = new StringBuilder(peliculaOculta);**

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

**if (pelicula.charAt(i) == letra.charAt(0)) {**

**nuevaPeliculaOculta.setCharAt(i, letra.charAt(0));**

**}**

**}**

**peliculaOculta = nuevaPeliculaOculta.toString();**

**} else {**

**// Incrementar los fallos**

**fallos++;**

**System.*out*.println("ERROR: " + ahorcado.substring(0, fallos));**

**}**

**System.*out*.println("Película a adivinar: " + peliculaOculta);**

**}**

**// Resultado final**

**if (peliculaOculta.equals(pelicula)) {**

**System.*out*.println("¡¡ENHORABUENA HAS ACERTADO!!");**

**} else {**

**System.*out*.println("Lo siento, has perdido. La película era: " + pelicula);**

**}**

**scanner.close();**

**}**

**}**

**(1.5 Puntos)**

**3.2 Crear la clase *TestCafeteras* que realice la siguientes operaciones:**

* **Cree un *array* que almacene cuatro cafeteras de distinta de capacidad máxima.**
* **Llenas todas la cafeteras de café**
* **Realizar varias operaciones de  *servirTaza y agregarCafe* con valores aleatorios de cantidad en todas las cafeteras de la tabla.**
* **Ordene el *array* por *capacidadActual***
* **Mostrar un informe según este formato:**

**---------------------------------------**

**Estado Actual de Cafeteras:**

**---------------------------------------**

**Nº Cantidad Actual Capacidad Máxima**

**1 1000 1500**

**2 850 2000**

**3 600 1000**

**4 200 2000**

|  |
| --- |
| **import java.util.Arrays;**  **import java.util.Random;**  **public class TestCafetera {**  **public static void main ( String [] arg) {**    **//Cree un array que almacene cuatro cafeteras de distinta de capacidad máxima.**    **Cafetera tcafeteras [] = new Cafetera [4];**    **tcafeteras[0] = new Cafetera();**  **tcafeteras[1] = new Cafetera(1500);**  **tcafeteras[2] = new Cafetera(500,250);**  **tcafeteras[3] = new Cafetera();**    **//Llenas todas la cafeteras de café**    **for (int i = 0; i < tcafeteras.length; i++) {**  **tcafeteras[i].llenarCafetera();**  **}**    **// Realizar varias operaciones de servirTaza y agregarCafe con valores aleatorios de cantidad en**  **//todas las cafeteras de la tabla.**    **Random rd = new Random();**    **for (int i = 0; i < tcafeteras.length; i++) {**  **tcafeteras[i].servirTaza( rd.nextInt(200));**  **tcafeteras[i].agregarCafe(rd.nextInt(200));**  **}**    **//Ordene el array por capacidadActual**    **Arrays.*sort*(tcafeteras);**    **//Mostrar un informe según este formato:**  **System.*out*.println("---------------------------------------");**  **System.*out*.println(" Estado Actual de Cafeteras: ");**  **System.*out*.println("---------------------------------------");**  **System.*out*.println("Nº Cantidad Actual Capacidad Máxima ");**  **for (int i = 0; i < tcafeteras.length; i++) {**  **System.*out*.printf("%2d %8d \t \t %8d \n", i+1,**  **tcafeteras[i].getCantidadActual(), tcafeteras[i].getCapacidadMaxima());**  **}**  **}**  **}** |

**public static int[] suprimenegativos(int tabla[])**

**Este método recibe como parámetro una tabla de enteros y devuelve otra tabla que solo contiene los valores positivos, con el cero incluido.**

**Ejemplo:**

**si int datos[] = {10,-2,4,-1, 0, 10 };**

**suprimenegativos(datos) → Devuelve {10,4,0,10}**

**public static int[] suprimenegativos(int tabla[]) {**

**// Contar la cantidad de números no negativos en la tabla**

**int contador = 0;**

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

**if (tabla[i] >= 0) {**

**contador++;**

**}**

**}**

**// Crear un nuevo vector con tamaño igual a la cantidad de números no negativos**

**int[] positivos = new int[contador];**

**// Copiar los valores no negativos del arreglo original al nuevo arreglo**

**int j = 0;**

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

**if (tabla[i] >= 0) {**

**positivos[j] = tabla[i];**

**j++;**

**}**

**}**

**// Devolver el nuevo arreglo**

**return positivos;**

**}**

**B) public static boolean sumafilarepe(int matriz[][])**

**Este método recibe una tabla bidimensional, y devuelve true si hay dos o más valores repetidos en el total de la suma de sus filas, si la suma de sus filas son todas distintas devuelve false;**

**int matriz1 [][] = {{2,5,6},**

**{6,6,1}};**

**int matriz2 [][] = {{2,1,6,9},**

**{6,6,1}};**

**int matriz3 [][] = {{2,5,6},**

**{1,6,1},**

**{5},**

**{5,3}};**

***sumafilarepe*(matriz1) → true Las sumas de la filas son ( 13 , 13 )**

***sumafilarepe*(matriz2) → false las sumas de las filas son ( 18, 13 )**

***sumafilarepe*(matriz3) → true las sumas de las filas son ( 13,8,5,8 )**

**public static boolean sumafilarepe(int matriz[][]) {**

**int filas = matriz.length;**

**int sumaFilas[] = new int[filas];**

**// Calcula la suma de cada fila**

**for (int i = 0; i < filas; i++) {**

**for (int j = 0; j < matriz[i].length; j++) {**

**sumaFilas[i] += matriz[i][j];**

**}**

**}**

**// Comprueba si hay valores repetidos en la suma de las filas**

**// Compara un elemento con los siguientes, el último no necesita ser comparado**

**for (int i = 0; i < filas - 1; i++) {**

**for (int j = i + 1; j < filas; j++) {**

**if (sumaFilas[i] == sumaFilas[j]) {**

**return true;**

**}**

**}**

**}**

**// Si no hay valores repetidos en la suma de las filas, devuelve falso**

**return false;**

**}**

**.**

**C) public static String ultimasletras(String cadena, int n)**

**Este método devuelve una nueva cadena formada las n últimas letras de la cadena, suponer que el valor de n esta entre 0 y el tamaño del string.**

***ultimasletras*("Extraordinario", 3) → "rio"**

***ultimasletras*("Extraordinario", 9) → "ordinario"**

***ultimasletras*("Extraordinario", 0) → ""**

**// Usando el método substring() de la clase String**

**public static String ultimasletras(String cadena, int n) {**

**return cadena.substring(cadena.length() - n);**

**}**

**// Construyendo manualmente la subcadena**

**public static String ultimasletras2(String cadena, int n) {**

**StringBuilder aux = new StringBuilder();**

**for (int i = cadena.length() - n; i < cadena.length(); i++) {**

**aux.append(cadena.charAt(i));**

**}**

**return aux.toString();**

**}**

**D)**

**Este método devuelve nueva cadena donde aparecen los valores de la tabla de String tcadenas separadas por el carácter -**

**unirCadenas({"Hola","Pepe","Luis"}) → "Hola-Pepe-Luis"**

**public static String unirCadenas(String[] tcadenas) {**

**StringBuilder sb = new StringBuilder();**

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

**sb.append(tcadenas[i]);**

**// La última palabra no lleva guión**

**if (i != tcadenas.length - 1) {**

**sb.append("-");**

**}**

**}**

**return sb.toString();**

**}**

**public class Bateria implements Comparable<Bateria> {**

**private int numSerie;**

**private int carga;**

**// Constructor: inicializa una nueva batería con un número de serie y carga 0**

**public Bateria(int numSerie) {**

**this.numSerie = numSerie;**

**this.carga = 0;**

**}**

**// Método para cargar la batería**

**public void cargar(int horas) {**

**int nuevaCarga = this.carga + horas \* 10; // Cada hora de carga añade 10% de carga**

**if (nuevaCarga > 100) {**

**this.carga = 100; // La carga máxima es 100%**

**} else {**

**this.carga = nuevaCarga;**

**}**

**}**

**// Método para descargar la batería**

**public void descargar(int horas) {**

**int nuevaCarga = this.carga - horas \* 10; // Cada hora de descarga reduce 10% de carga**

**if (nuevaCarga < 0) {**

**this.carga = 0; // La carga mínima es 0%**

**} else {**

**this.carga = nuevaCarga;**

**}**

**}**

**// Método para verificar si la batería está completamente cargada**

**public boolean estaCargada() {**

**return this.carga == 100;**

**}**

**// Método para verificar si la batería está completamente descargada**

**public boolean estaDescargada() {**

**return this.carga == 0;**

**}**

**// Método para representar la batería como una cadena de texto**

**public String toString() {**

**return "Bateria " + this.numSerie + ": " + this.carga + "%";**

**}**

**// Método para comparar baterías basado en su nivel de carga**

**public int compareTo(Bateria o) {**

**return this.carga - o.carga;**

**}**

**}**