Unidad 4. Clases y Objetos.

Ejercicios:

E1. Test.

T1. Una caja negra es un componente del software que se utiliza sin conocer su estructura interna. La interfaz de una caja negra es la forma en que interactúa con el resto del software, mientras que la implementación es la forma en que se construye la caja negra.

T2. Las subrutinas son un contrato porque establecen un acuerdo entre el programador y el compilador. El aspecto sintáctico del contrato se refiere a la forma que se llama a la subrutina y a los parámetros que se pasan a la misma. El aspecto semántico se refiere a lo que hace la subrutina y a lo que devuelve.

T3. Las subrutinas ayudan a diseñar programas de forma top-down porque permiten dividir un programa en partes más pequeñas y manejables. Esto hace que el programa sea más fácil de entender y depurar. Además, las subrutinas pueden ser reutilizadas en diferentes partes del programa, lo que ahorra tiempo y reduce la cantidad de código que se necesita escribir.
Diseñar top-down permite abordar primero la estructura general del programa, definiendo subrutinas y sus interfaces sin preocuparse por los detalles internos. Luego, puedes trabajar en la implementación de cada subrutina de manera aislada.

T4. Los parámetros de una subrutina se utilizan para pasar información a la subrutina.
Los parámetros formales son los nombres de los parámetros que se especifican en la definición de la subrutina, mientras que los parámetros actuales son los valores que se pasan a la subrutina cuando se llama. La diferencia entre los parámetros formales y los parámetros actuales es que los primeros son simplemente nombre, mientras que los egundos son los valores reales que se pasan a la subrutina.

T5. Una API es un conjunto de reglas y protocolos que se utilizan para desarrollar software.
En Java, una API es un conjunto de clases y métodos que se utilizan para interactuar con el lenguaje de programación. Por ejemplo, la API de Java proporciona clases y métodos para trabajar con archivos, bases de datos, gráficos, redes, etc.

```
public class EjemploAPI {

public static void main ( String [ ] args) {
//Uso de la API para imprimir en consola
System.out.println(“Hola, esto es un mensaje de la API de Java”);
```

“System” es una clase de la Api de Java que proporciona acceso a varios recursos del sistema, incluída la consola.
“out” es un objeto dentro de la clase “system” que representa la salida estándar.
“println” es un método de la clase “out” que permite imprimir una línea de texto en la consola.

EJ2.

```
package ClasesyObjetos;

public class Empleado {
    String nombre;
    int edad;

    Empleado(String nombre, int edad){//construyo el constructor
      this.nombre = nombre;
      if(edad >= 18) {
            this.edad = edad;
      }else {
            this.edad = 18;
      }
    }
    public class PruebaEmpleado{


      public static void main(String[] args) {
            // TODO Auto-generated method stub
      Empleado e1 = new Empleado("Paco", 14);
      System.out.println(e1.edad);
```

EJ3.

Un método estático es un método que pertenece a una clase en lugar de a una instancia de la clase. Esto significa que puedes llamar a un método estático sin tener que crear un objeto de la clase.
En el caso del método getNombre() propuesto, el método es estático porque devuelve el nombre del empleado. Sin embargo, el método propuesto no es correcto porque no tiene acceso al nombre del empleado. El nombre es una variable de instancia, lo que significa que solo se puede acceder a ella a través de una instancia de la clase Empleado.
Para corregirlo podríamos hacer que el campo 'nombre' sea estático:

public class Empleado{
static String nombre;
int edad;
public static String getNombre(){
return Empleado.nombre;

o hacer que el método getNombre() no sea estático.

public class Empleado {
String nombre;
int edad;
public String getNombre(){
return nombre;

EJ4.

```
package ClasesyObjetos;

public class EmpleadoEj4 {
      String nombre;
      int edad;
      double sueldo;
      String password = "1234";
```

```
	EmpleadoEj4(String nombre, int edad, double sueldo){//Constructor del objeto EmpleadoEj4
		this.nombre= nombre;
		this.edad= edad;
		this.sueldo= sueldo;


		}
	public String getNombre()  {
		return nombre;

	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public int getEdad() {
		return edad;
	}
	public void setEdad(int edad) {
		if (edad >= 18 ) {
			this.edad = edad;
		}else {
			System.out.println("La edad debe ser mayor o igual a 18.");
		}
	}
			public double getSueldo(String password) {
		if (password.equals(password)) {
			return sueldo;
		}else {
			System.out.println("Contraseña incorrecta.");
			return 0;
		}
	}
	public void setSueldo(double sueldo, String password) {
		if(password.equals(password)) {

		}else {
			System.out.println("Contraseña incorrecta.");
		}


	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
	    Empleado empleado1 = new Empleado("Pepe", 35, 27000);

	    System.out.println("Nombre: " + empleado1.getNombre());
	    System.out.println("Edad: " + empleado1.getEdad());
	    System.out.println("Sueldo " + empleado1.getSueldo());

	    empleado1.setSueldo(30000, "password");
	    System.out.println("Nuevo sueldo: " + empleado1.getSueldo());

	    empleado1.setEdad(40);
	    System.out.println("Nueva edad: " + empleado1.getEdad());
	}

}
```

EJ5.

Observo que la primera palabra Public está en mayúscula y debe ser en minúscula (public).
El nombre de la clase se define según el comentario como Punto2D y la clase está el nombre de (empleado) esto es incorrecto.
El constructor usa la palabra (void) que es un tipo de retorno y void no devuelve ningún valor.
Los valores x e y están asignados a sí mismos en lugar de asignarlos a las variables de instancia.
tendrían que ser this.x = x; this.y = y;
Por último en el método main el objeto p2 no está definido y el método println debería usar (p1.x + ” , ” + p1.y);

```
public class Punto2D{
public double x, y;

public Punto2D (double x, souble y) {
    this.x = x;
    this.y = y;

public class PruebaPuntos {
public static void main(String[] args) {
Punto2D p1 = new Punto2D(3, 2.5);
System.out.println(p1.x + ” , ” + p1.y);
```

Ej6.

a) No hacemos nada.
Esta opción no es buena porque aunque Java crea un constructor predeterminado automáticamente, no garantiza que las coordenadas se establezcan en (0,0). Dependiendo de la implementación de Java, podrían ser otros valores predeterminados.

b) Esta opción no es válida porque un constructor no debe devolver nada.

c) Aquí se está usando un constructor dentro del constructor, esto daría error.

d) La forma correcta de hacerlo es llamar al propio constructor desde otro constructor utilizando (this).

```
public class Punto2D {
public double x, y;

public Punto2D() {
this(0, 0); //llamamos al constructor y le damos los valores

public Punto2D( double x, double y) { //constructor con parámetros
this.x = x; // establecemos las coordenadas según los valores proporcionados
this.y = y;
```

Ej7.

```
package ClasesyObjetos;

public class Punto2D {

      double x, y;

      public Punto2D(double x, double y) {
```

```
		this.x = x;
		this.y = y;
	}
	public double getX() {
		return x;
	}
	public void setX(double x) {
		this.x = x;
	}
	public double getY() {
		return y;
	}
	public void setY(double y) {
		this.y = y;
	}
	public Punto2D suma(Punto2D otroPunto) {
		double nuevaX = this.x + otroPunto.getX();
		double nuevaY = this.y + otroPunto.getY();
		return new Punto2D(nuevaX, nuevaY);

	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Punto2D p1 = new Punto2D(3,  2.5);
		Punto2D p2 = new Punto2D(1, -1);
		Punto2D p3 = p1.suma(p2);

		System.out.println("Coordenadas de p3: (" + p3.getX() + "," + p3.ge-
tY() + ")");
	}
```

EJ8.

```
	}
	@Override
	public String toString() {
		return "(" + x + ", " + y + ")";
	}
	public Punto2D suma(Punto2D otroPunto) {
		double nuevaX = this.x + otroPunto.getX();
		double nuevaY = this.y + otroPunto.getY();
		return new Punto2D(nuevaX, nuevaY);

	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Punto2D p1 = new Punto2D(3,  2.5);
		Punto2D p2 = new Punto2D(1, -1);
		Punto2D p3 = p1.suma(p2);

		System.out.println(p1 + ", " + p2+ "= " + p3);
	}
```

EJ9.

```
@Override
	public boolean equals (Object o) {// comparamos las coodenadas x e y
		if(this == o)
			return true;// son lo mismo, por lo tanto son iguales

		if (o == null ||getClass() != o.getClass())
			return false;
		Punto2D punto2D = (Punto2D) o;//convertimos el objeto a Punto2D para
compara las coodenadas
		return Double.compare(punto2D.x, x) == 0 && Double.compare(punto-
2D.y, y) == 0;

	}
	public Punto2D suma(Punto2D otroPunto) {
		double nuevaX = this.x + otroPunto.getX();
		double nuevaY = this.y + otroPunto.getY();
		return new Punto2D(nuevaX, nuevaY);

	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Punto2D p1 = new Punto2D(3,  2.5);
		Punto2D p2 = new Punto2D(3, 2.5);

		p2.setX(p1.getX());
		p2.setY(p1.getY());


		System.out.println(p1 + "es igual " + p2+ "?" + p1.equals(p2));
	}

}
```

EJ16.

```
package ClasesyObjetos;

public class Redondeo {

	public Redondeo() {//constructor de la clase

	}

	public double redondearADecimas(double x){//método para redondear a las
décimas
		return Math.floor(x * 10 + 0.5) / 10;
	}
   public double redondearACentesimas(double x) {//Método para redondear a la
centésimas
		return Math.floor(x * 100 + 0.5) / 100;
		}
 double redondearAMilesimas(double x) { //Método para redondear a las milesimas
		return Math.floor(x * 1000 + 0.5) /1000;
   }
   public double redondearADiezmilesimas(double x) {
			return Math.floor(x * 10000 + 0.5) /10000;
   }


	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Redondeo redondeo = new Redondeo();
```

```
            double numero = 123.456789;

            System.out.println("Número original:" + numero);
            System.out.println("Redondeado a décimas: " + redondeo.redondearADe-
cimas(numero));
            System.out.println("Redondeado a centésimas: " + redondeo.redondea-
rACentesimas(numero));
            System.out.println("Redondeado a milésimas: " + redondeo.redondearA-
Milesimas(numero));
            System.out.println("Redondeado a diezmilésimas: " + redondeo.redon-
dearADiezmilesimas(numero));
      }
```

EJ21 (Hola Profesor, este ejercicio lo he planteado como los exámenes que tuvimos en la primera evaluación, muchas gracias, por colgar los vídeos ya que necesitaba ver como piensas y como planteas los ejercicios, los he analizado cada uno y apuntado el código para estudiarlos y pensarlos , es muy largo pero la parte del menú es parecida a los exámenes, tengo una duda importante en como esclarecer el método mayordeEdad, ya que creo que no lo tengo bien porque hay muchos factores como el formato, si en el año actual ha cumplido años.... y creo que me he complicado la vida, bueno esperaré a los resultados, muchas gracias)

```
package ClasesyObjetos;

import java.time.LocalDate;

import java.util.Scanner;
import java.io.File;
import java.io.PrintWriter;
public class Cliente { //Mostramos los atributos

      private enum Sexo { HOMBRE, MUJER}
      private String nombre;
      private String fechaNacimiento;
      private int numPaciente;
      private char sexo;
      private double peso;
      private double altura;

      private static int contadorPacientes = 0;//Variable estática para generar
números únicos de paciente

      //Constructor con nombre, fecha de nacimiento y sexo.
      public Cliente(String nombre, String fechaNacimiento, char sexo ) {
            this.nombre= nombre;
            this.fechaNacimiento = fechaNacimiento;
            this.sexo = sexo;
            this.peso = 0.0; //valor por defecto para tipo primitivo (double)
            this.altura = 0.0;//valor por defecto para tipo primitivo (double)
            this.numPaciente = ++contadorPacientes;
      }
      //Constructor con todos los atributos
      public Cliente(String nombre, String fechaNacimiento, char sexo, double
peso, double altura) {
            this.nombre = nombre;
            this.fechaNacimiento = fechaNacimiento;
            this.sexo = sexo;
            this.peso = peso;
            this.altura = altura;
            this.numPaciente = generaNumPaciente();// LLamada al método gnera-
NumPaciente() para obtener un número único para cliente.
      }
      //Método para calcular el IMC.
      public int calcularIMC() {
```

```
            if (altura == 0) {//Creo esta excepción para manejar posibles erro-
res.
                  throw new IllegalStateException("La altura no pude ser cero
pra calcular el IMC");
            }
            double imc = peso / (altura * altura);

            if (imc < 20) {
                  return -1;
            }else if (imc <= 25) {
                  return 0;
            }else if (imc > 25) {

            }return 1;
      }
      //Método MayorDeEdad.
      public boolean mayorDeEdad() {
            LocalDate fechaNac = LocalDate.parse(this.fechaNacimiento);//con-
vierto la fecha de nacimiento a LocalDate
            LocalDate ahora = LocalDate.now();//obtengo fecha actual
            int edad = ahora.getYear() - fechaNac.getYear();//calculo la edad
            //Ajustar la edad si el cumpleaños de este año aún no ha pasado
            if(fechaNac.getMonthValue() > ahora.getMonthValue() || (fechaNac.ge-
tMonthValue() == ahora.getMonthValue() &&
                        fechaNac.getDayOfMonth() > ahora.getDayOfMonth())) {
                  edad--;
            }
            return edad >= 18;
      }
      //Método comprobarSexo(char sexo).
      private void comprobarSexo(char sexo) {
            if(sexo == 'H' || sexo == 'M') {
                  this.sexo = sexo;
            }else {
                  this.sexo = 'H'; //Sexo por defecto si no es correcto.

            }
      }
      //Método toSring
      @Override
      public String toString() {
            return "Cliente{" + "nombre = " + nombre + '\'' + ", fechaNacimiento
= ' " + fechaNacimiento + '\'' +
                        ", numPaciente = " + numPaciente + ", sexo = " + sexo +
", peso = " + peso + ", altura = " + altura +'}';
      }
      //Método privado generarNumPaciente.
      private int generaNumPaciente() {
            return ++contadorPacientes;
      }
      //Métodos Getter y Setter de cada dato.
      public String getNombre() {
            return nombre;
      }
      public void setNombre(String nombre) {
            this.nombre = nombre;
      }
      public String getfechaNacimiento() {
            return fechaNacimiento;
      }
      public void setfechaNacimiento(String fechaNacimiento) {
            this.fechaNacimiento = fechaNacimiento;
      }
      public char getSexo() {
```

```
            return sexo;
      }
      public void setSexo( char sexo) {
            this.sexo = sexo;
      }
      public double getPeso() {
            return peso;
      }
      public void setPeso(double peso) {
            this.peso = peso;
      }
      public double getAltura() {
            return altura;
      }
      public void setAltura(double altura) {
            this.altura = altura;
      }
      // Añade el getter gerEdad que devuelva un entero.
      public int getEdad() {
            LocalDate fechaNac = LocalDate.parse(this.fechaNacimiento);//convie-
ro fecha de nacimiento a LocaDate
            LocalDate ahora = LocalDate.now();//Obtengo la fecha actual
            int edad = ahora.getYear() - fechaNac.getYear();
            return edad;

      }

      public static void main(String[] args) {
            // TODO Auto-generated method stub
          String [] pacientes = null;

          Scanner teclas = new Scanner (System.in);
          //Intento abrir el archivo de Pacientes y lo protejo.
          try(Scanner scf = new Scanner(new File("pacientes.txt"))){

          }catch(Exception e) {
              System.out.println("Archivo 'pacientes.txt' no encontrado. Se ini-
ciará con una lista vacía");
          }
          String opcion = "";
          do {//Mostramos menú.
              System.out.println("1. Nuevo paciente.");
              System.out.println("2. Ver/Modificar Paciente");
              System.out.println("3. Salir. ");
              System.out.println(" Escoja opción: ");
              opcion = teclas.nextLine();

              switch (opcion) {
              case "1":
                    System.out.print("Nombre: ");//Pido nombre y valido.
                    String nombre = teclas.nextLine();
                    if (nombre.isEmpty()) {
                          System.out.println("Error: El nombre no puede estar
vacío.");
                          return;
                    }
                    System.out.print("Fecha de Nacimiento (yyyy-mm-dd)");//Pido
Fecha de nacimiento y valido
                    try {
                          LocalDate.parse(teclas.nextLine());
                    }catch (Exception e) {
                          System.out.println("Error: Formato de fecha inváli-
do.");
                          return;
```

```
                    }
                    System.out.print("Sexo (H/M): ");//Pido sexo y valido
                    String sexo = teclas.nextLine().toUpperCase();
                    if(!sexo.equals("H") && !sexo.equals("M")) {
                          System.out.println("Error: Sexo debe ser 'H' O 'M'.");
                          return;
                    }
                    System.out.print("Altura (m): ");//Pido altura y valido
                    try {
                          double altura = Double.parseDouble(teclas.nextLine());
                          if(altura < 0.5 || altura > 2.5) {
                                throw new NumberFormatException();
                          }
                    }catch (NumberFormatException e) {
                          System.out.println("Error: Altura inválida.");
                          return;
                    }
                    System.out.print("Peso (kg): ");//Pido peso y valido
                    try {
                          double peso = Double.parseDouble(teclas.nextLine());
                          if(peso < 20 || peso > 300) {
                                throw new NumberFormatException();
                          }
                    }catch(NumberFormatException e) {
                          System.out.println("Error: Peso inválido.");
                          return;
                    }
                    for (int i = 0; i < pacientes.length; i++) {
                          if(pacientes[i] == null) {
                    String nuevoPaciente= nombre;
                                pacientes [i]= nuevoPaciente;
                                System.out.println("Paciente añadido con
éxito.");
                          }
                       }
                    break;
              case "2":
                    System.out.print("Ingrese el número o nombre del paciente:
");
                    String busqueda = teclas.nextLine();
                    int indicePaciente = buscar(busqueda, pacientes);
                    if(indicePaciente >= 0) {
                          System.out.println("Paciente encontrado: " + pacien-
tes[indicePaciente]);
                          System.out.print("¿Desea modificar los datos de este
paciente? (Sí/No): ");
                          String respuesta = teclas.nextLine();

                       if(respuesta.equalsIgnoreCase("Sí")) {
                             String[] datosPaciente = pacientes[indicePacien-
te].split(",");
                             System.out.print("Nombre (" +datosPaciente[0] +
"): ");
                             String nombre1 = teclas.nextLine();

                             if (!nombre1.isEmpty()) {
                                   datosPaciente[0]= nombre1;
                             }
                             pacientes[indicePaciente] = String.join(",", da-
tosPaciente);
                             System.out.println("Datos del paciente actualiza-
dos.");
                       }else {
                             System.out.println("Paciente no encontrado.");
```

```
                        }

                    }


                    break;
              case "3": //Salir y guardar cambios en el fichero
                    guardarDatos(pacientes);
                    break;
                    default:
                          System.out.println("Opción no válida. Intente de nue-
vo.");
                          break;
              }
           }while (!opcion.equals("3"));

      }

        public static int buscar(String e, String [] a) {
             if(a == null) return -1;
             for (int i = 0; i < a.length; i++) {
                   if (a[i] != null && a[i].contains(e)) {
                         return i;
                   }

             }
              return -1;
       }
         private static void guardarDatos(String [] pacientes) {
             try(PrintWriter pw = new PrintWriter(new File("pacientes.txt"))){
                  for (String paciente : pacientes) {
                        if (paciente != null) {
                              pw.println(paciente);
                        }
                  }
             }catch(Exception e) {
                   System.out.println("Error al guardar los datos: " + e.getMe-
ssage());
             }
          }
}
```

EJ22.

```
ackage ClasesyObjetos;

public class ConjuntoEnteros {

    int [] elementos;
      int contador;
      int capacidad = 10;

      public ConjuntoEnteros() {
            elementos = new int[capacidad];
            contador = 0;
      }
      public int cardinal() {
            return contador;
      }
      public void insertarElemento(int elemento) {
            if(!pertenece(elemento) && contador < capacidad ) {
                  elementos[contador++] = elemento;
```

```
        }
  }
public void borrarElemento(int elemento) {
  for (int i = 0; i < contador; i++) {
              if(elementos[i] == elemento) {
                    for (int j = 0; j < contador -1; j++) {

                    }
                    contador--;
                    return;
              }
        }
}
public boolean pertenece(int elemento) {
  for (int i = 0; i < contador; i++) {
        if(elementos[i] == elemento) {
              return true;
        }

        }
        return false;
}


  public static void main(String[] args) {
        // TODO Auto-generated method stub

  }
```