Diseño y Programación Orientados a Objetos

1er. Semestre 2025

Certamen I ELO-329 Diseño y Programación Orientada a Objetos (1s2025)

En este certamen usted no podrá hacer preguntas. Si algo no está claro, indíquelo en su respuesta, haga una suposición razonable y resuelva conforme a ella.

Segunda parte, **con apuntes**.

Segunda pregunta (35 pts.)

Laboratorio de Clonación

El laboratorio **BioTek ExoLabs** estudia especies experimentales recolectadas en planetas lejanos. En su más reciente expedición, los investigadores capturaron un espécimen original de dos especies alienígenas: **EspecieAlpha** y **EspecieBeta**.

La información de cada espécimen recolectado se entregará a través de un archivo de texto que contiene los datos relevantes de cada uno (nombre, peso, altura, especie y fecha de nacimiento).

Su objetivo es clonar cada espécimen, aplicarles experimentos aleatorios y generar un informe de los resultados obtenidos, comparando los clones con su original. Usted ha sido encargado de desarrollar un sistema en Java que modele este proceso, respetando principios de **herencia, encapsulamiento y copia profunda**.

3.1 - Modelado de la entidad base (10 pts.)

Cree una **clase abstracta** `EntidadBiologica`, que represente cualquier ser vivo del laboratorio. Esta clase debe contener los siguientes atributos:

- `String nombre`
- `double peso`
- `double altura`
- `Date fechaNacimientoOriginal` (fecha real del primer espécimen, obtenida del archivo de texto)
- `Date fechaNacimiento` (fecha de nacimiento del individuo actual; si es un clon, corresponde a la fecha de hoy)
- `String estado` (`"neutro"`, `"enfermo"` o `"mejorado"`)

También debe **declarar** los siguientes métodos:

- `public Object clone()` que realice una **copia profunda** del objeto.
- `public abstract void aplicarExperimento(double resultado)` que simula la aplicación de un experimento y modifica atributos según la especie.

- `public String toString()` que describe al individuo, incluyendo todos los atributos y si se trata de un clon. Este debe entregar:
- Nombre, especie
- Peso y altura
- Estado actual (`neutro`, `enfermo`, `mejorado`)
- Fecha de nacimiento original
- Fecha de nacimiento del individuo (distinta en los clones)

Ejemplo esperado:

```
Especie: EspecieAlpha
Nombre: Zarglax
Altura: 2.38
Peso: 80.5
Estado: mejorado
Fecha nacimiento original: 2015-03-14
Fecha nacimiento individuo: 2015-03-14
```

Todos los individuos deben iniciar con estado `"neutro"`. Recuerde implementar `Cloneable` y sobrescribir correctamente el método `clone()`.

R: Distribución general de puntos

- (2pts) Creación de clase solicitada
- (2pts) Definición e implementación de métodos
- (2pts) Definición de atributos
- (2pt) Implementación de interfaz
- (2pts) Funcionamiento general

```
import java.util.Date;

public abstract class EntidadBiologica implements Cloneable {
    private String nombre;
    private double peso;
    private double altura;
    private Date fechaNacimientoOriginal;
    private Date fechaNacimiento;
    private String estado;

    public EntidadBiologica(String nombre, double peso, double altura, Date
fechaNacimientoOriginal) {
        this.nombre = nombre;
        this.peso = peso;
        this.altura = altura;
        this.fechaNacimientoOriginal = new
```

```
Date(fechaNacimientoOriginal.getTime());
        this.fechaNacimiento = new Date(fechaNacimientoOriginal.getTime()); //
por omisión, igual a original
        this.estado = "neutro";
    }

    // Métodos getters y setters
    public String getNombre() {
        return nombre;
    }

    public double getPeso() {
        return peso;
    }

    public double getAltura() {
        return altura;
    }

    public Date getFechaNacimientoOriginal() {
        return new Date(fechaNacimientoOriginal.getTime());
    }

    public Date getFechaNacimiento() {
        return new Date(fechaNacimiento.getTime());
    }

    public String getEstado() {
        return estado;
    }

    // Setters protegidos (para uso por subclases)
    protected void setPeso(double peso) {
        this.peso = peso;
    }

    protected void setAltura(double altura) {
        this.altura = altura;
    }

    protected void setEstado(String estado) {
        this.estado = estado;
    }

    public void setFechaNacimiento(Date fechaNacimiento) {
        this.fechaNacimiento = new Date(fechaNacimiento.getTime());
    }

    // Método clone con copia profunda
    public Object clone() {
        try{
            EntidadBiologica copia = (EntidadBiologica) super.clone();
            copia.fechaNacimientoOriginal = new
Date(this.fechaNacimientoOriginal.getTime());
```

```
            copia.fechaNacimiento = new Date(); // la fecha del clon es hoy
            return copia;
        }catch(CloneNotSupportedException e){return null;}
    }

    // Método abstracto a implementar por subclases
    public abstract void aplicarExperimento(double resultado);


    // toString muestra información de la especie
    public String toString() {
        String nombreMostrar = nombre;

        // Si fechas son iguales, no es clon, en caso contrario, es clon.
        if (!fechaNacimiento.equals(fechaNacimientoOriginal)) {
            nombreMostrar += " Clon";
        }

        return "Especie: " + this.getClass().getName() + "\n" +
               "Nombre: " + nombreMostrar + "\n" +
               "Altura: " + altura + "\n" +
               "Peso: " + peso + "\n" +
               "Estado: " + estado + "\n" +
               "Fecha nacimiento original: " + String.format("%tF",
fechaNacimientoOriginal) + "\n" +
               "Fecha nacimiento individuo: " + String.format("%tF",
fechaNacimiento);
    }

}
```

3.2 - Definición de subclases por especie (10 pts.)

Implemente dos subclases de `EntidadBiologica`, las cuales deben sobrescribir el método `aplicarExperimento(double resultadoAleatorio)` y evaluar el valor entregado de la siguiente forma:

- `< 0.33`: `"enfermo"`
- `0.33-0.66`: `"neutro"`
- `> 0.66`: `"mejorado"`

La generación del número aleatorio será responsabilidad del laboratorio.

Las clases a generar son las siguientes:

a) **`EspecieAlpha`**

- Si al realizar el experimento el individuo resulta `"enfermo"`: la **altura** se multiplica por 0.8.
- Si al realizar el experimento el individuo resulta `"mejorado"`: la **altura** se divide por 0.8.

b) **`EspecieBeta`**

- Si al realizar el experimento el individuo resulta `"enfermo"`: el **peso** se multiplica por 1.2.
- Si al realizar el experimento el individuo resulta `"mejorado"`: el **peso** se divide por 1.2.

R: Distribución general de puntos

- (2pts) Creación clase EspecieAlpha
- (2pts) Creación clase EspecieBeta
- (2pts) Herencia con clase EntidadBiologica
- (2pts) Implementación de método abstracto
- (2pts) Verificación de experimentos

```
import java.util.Date;
// Código clase EspecieAlpha
public class EspecieAlpha extends EntidadBiologica {

    public EspecieAlpha(String nombre, double peso, double altura, Date
fechaNacimientoOriginal) {
        super(nombre, peso, altura, fechaNacimientoOriginal);
    }

    public void aplicarExperimento(double resultado) {
        if (resultado < 0.33) {
            setEstado("enfermo");
            setAltura(getAltura() * 0.8);
        } else if (resultado <= 0.66) {
            setEstado("neutro");
        } else {
            setEstado("mejorado");
            setAltura(getAltura() / 0.8);
        }
    }
}
```

```
// Código EspecieBeta
import java.util.Date;
// Código EspecieBeta
public class EspecieBeta extends EntidadBiologica {

    public EspecieBeta(String nombre, double peso, double altura, Date
fechaNacimientoOriginal) {
        super(nombre, peso, altura, fechaNacimientoOriginal);
    }

    public void aplicarExperimento(double resultado) {
        if (resultado < 0.33) {
            setEstado("enfermo");
            setPeso(getPeso() * 1.2);
        } else if (resultado <= 0.66) {
            setEstado("neutro");
        } else {
```

```
            setEstado("mejorado");
            setPeso(getPeso() / 1.2);
        }
    }
}
```

3.3 - Lectura desde archivo y clonación (5 pts.)

Cree un programa el cual debe leer un archivo `individuos.txt` (creado por ud.) con una línea por especie:

```
nombre,peso,altura,especie,fechaNacimientoOriginal
```

Ejemplo:

```
Zarglax 80.5 1.9 EspecieAlpha 2015-03-14
Dranox 42.0 1.3 EspecieBeta 2018-11-02
```

Por cada línea del archivo, el método `main` debe:

- Crear el objeto correspondiente (`EspecieAlpha` o `EspecieBeta`).
- Generar **un clon** utilizando el método `clone()` y modificar la fecha de nacimiento (`fechaNacimiento`) del clon con la fecha actual (`new Date()`).
- Guardar los dos objetos en una estructura de tipo `ArrayList<EntidadBiologica>` creada en el laboratorio.

R: (solo se muestra la parte solicitada en esta pregunta del método `main`) Distribución general de puntos

- (1pt) Lectura archivo
- (1pt) Parsear por espacios
- (1pt) Completado de atributos
- (1pt) Creación de individuos
- (1pt) Creación de clones

```
.
.
.
Scanner scanner = new Scanner(new File("individuos.txt"));

while (scanner.hasNextLine()) {
    // trim para borrar espacios en blanco al inicio al final
    String linea = scanner.nextLine().trim();
    // Split por espacios
    String[] partes = linea.split("\\s+");

    String nombre = partes[0];
```

```
        double peso = Double.parseDouble(partes[1]);
        double altura = Double.parseDouble(partes[2]);
        String especie = partes[3];

        // Parseo manual de fecha yyyy-MM-dd con solo java.util.Date
        String[] fechaPartes = partes[4].split("-");
        int anio = Integer.parseInt(fechaPartes[0]) - 1900; // año desde 1900
        int mes = Integer.parseInt(fechaPartes[1]) - 1;     // meses desde 0
        int dia = Integer.parseInt(fechaPartes[2]);
        Date fechaNacimientoOriginal = new Date(anio, mes, dia);

        EntidadBiologica individuo = null;

        if (especie.equals("EspecieAlpha")) {
            individuo = new EspecieAlpha(nombre, peso, altura,
    fechaNacimientoOriginal);
        } else if (especie.equals("EspecieBeta")) {
            individuo = new EspecieBeta(nombre, peso, altura,
    fechaNacimientoOriginal);
        }

        EntidadBiologica clon = (EntidadBiologica) individuo.clone();
        clon.setFechaNacimiento(new Date());
    .
    .
    .
    }
```

3.4 - Clase Laboratorio (5 pts.)

Implemente una clase `Laboratorio`, encargada de gestionar los experimentos y mostrar los resultados. Esta clase debe:

- Contener un atributo `ArrayList<EntidadBiologica>` con todos los individuos.
- Generar una instancia de `Random` con semilla fija (`new Random(42)`).
- Tener un método `public void realizarExperimentos()` que, para cada individuo, genere un número aleatorio `double` por cada uno usando `nextDouble()`, y llame a `aplicarExperimento(resultadoAleatorio)` entregándole dicho valor.
- Tener un método `public void imprimirInforme()` que recorra la lista e imprima por consola el resultado usando `toString()` de cada objeto.
- Utilice la clase `Laboratorio` para gestionar los experimentos dentro del `main` de la pregunta anterior (3.3).

Esta clase representa el punto central del sistema: gestiona los individuos, aplica los experimentos y entrega los resultados.

R: Distribución general de puntos

- (1pt) Creación de clase Laboratorio

- (1pt) Definición de atributos
- (1pt) Definición de métodos
- (1pt) Implementación funcionalidad aleatoria y realización de experimentos
- (1pt) Funcionamiento general correcto

```
import java.util.ArrayList;
import java.util.Random;

public class Laboratorio {
    private ArrayList<EntidadBiologica> individuos;
    private Random random;

    public Laboratorio() {
        this.individuos = new ArrayList<>();
        this.random = new Random(42); // Semilla fija
    }

    public void agregarIndividuo(EntidadBiologica e) {
        individuos.add(e);
    }

    public void realizarExperimentos() {
        for (EntidadBiologica individuo : individuos) {
            double resultado = random.nextDouble();
            individuo.aplicarExperimento(resultado);
        }
    }

    public void imprimirInforme() {
        for (EntidadBiologica individuo : individuos) {
            System.out.println(individuo);
            System.out.println();
        }
    }
}
```

3.5 - Informe de resultados (5 pts.)

Después de aplicar el experimento a cada individuo (originales y clones), dentro del `main` del programa se debe imprimir por consola un **informe resumen** con el resultado de cada uno.

Utilice `imprimirInforme()` de la clase `Laboratorio` para mostrar la información.

Ejemplo esperado:

```
Especie: EspecieAlpha
Nombre: Zarglax
Altura: 2.38
Peso: 80.5
```

```
Estado: mejorado
Fecha nacimiento original: 2015-03-14
Fecha nacimiento individuo: 2015-03-14

Especie: EspecieAlpha
Nombre: Zarglax Clon 1
Altura: 1.52
Peso: 80.5
Estado: enfermo
Fecha nacimiento original: 2015-03-14
Fecha nacimiento individuo: 2024-05-04
.
.
.
```

R: (código del método `main` completo) Distribución general de puntos

- (1pt) Agregación del laboratorio
- (1pt) Agregación de individuos al laboratorio
- (1pt) Realización de experimentos
- (1pt) Creación de informe
- (1pt) Funciomaiento general correcto

```java
import java.io.File;
import java.util.*;

public class P2 {
    public static void main(String[] args) {
        try {
            Laboratorio laboratorio = new Laboratorio();

            Scanner scanner = new Scanner(new File("individuos.txt"));

            while (scanner.hasNextLine()) {
                // trim para borrar espacios en blanco al inicio al final
                String linea = scanner.nextLine().trim();
                // Split por espacios
                String[] partes = linea.split("\\s+");

                String nombre = partes[0];
                double peso = Double.parseDouble(partes[1]);
                double altura = Double.parseDouble(partes[2]);
                String especie = partes[3];

                // Parseo manual de fecha yyyy-MM-dd con solo java.util.Date
                String[] fechaPartes = partes[4].split("-");
                int anio = Integer.parseInt(fechaPartes[0]) - 1900; // año
desde 1900
                int mes = Integer.parseInt(fechaPartes[1]) - 1;     // meses
desde 0
                int dia = Integer.parseInt(fechaPartes[2]);
```

```
                Date fechaNacimientoOriginal = new Date(anio, mes, dia);

                EntidadBiologica individuo = null;

                if (especie.equals("EspecieAlpha")) {
                    individuo = new EspecieAlpha(nombre, peso, altura,
fechaNacimientoOriginal);
                } else if (especie.equals("EspecieBeta")) {
                    individuo = new EspecieBeta(nombre, peso, altura,
fechaNacimientoOriginal);
                }

                EntidadBiologica clon = (EntidadBiologica) individuo.clone();
                clon.setFechaNacimiento(new Date());

                laboratorio.agregarIndividuo(individuo);
                laboratorio.agregarIndividuo(clon);
            }

            laboratorio.realizarExperimentos();
            laboratorio.imprimirInforme();

            scanner.close();
        } catch (Exception e) {}
    }
}
```

Restricciones y observaciones

- El programa **no debe utilizar interfaces gráficas**.
- Use estructuras de datos vistas en clases.
- No es necesario manejar errores de entrada. Se asume que el archivo viene correctamente formateado.
- **¿Qué subir a AULA?** Cree un archivo `P2.tar` (`.zip` o `.rar`) con todas sus clases y súbalo.

R: Código completo disponible en el siguiente enlace