ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
PROGRAMACIÓN ORIENTADA A OBJETOS
EXAMEN PRIMERA EVALUACIÓN 2025 - 2T

Nombre: _____ Paralelo: _____

1. Responda Verdadero o Falso según corresponda. [5 puntos]

Para invocar una variable de instancia llamada **valor** dentro de un método estático necesita usar this.valor.	
Al sobrescribir un método, este puede ser más restrictivo que el original	
Una variable del tipo de clase hija puede apuntar a un objeto de la clase padre.	
Se puede invocar a un constructor dentro de otro.	
El encapsulamiento se implementa usando correctamente los modificadores de acceso.	

2. ¿Qué pasaría si tratas de compilar y ejecutar el siguiente código Java? [4 puntos]

```
class Top {
        public Top(String s) {
            System.out.print("B");
        }
}

public class Bottom extends Top {
        public Bottom(String s) {
              System.out.print("D");
        }
        public static void main(String[] args) {
            Bottom obj = new Bottom("C");
             System.out.println(" ");
        }
}
```

a) Compilará e imprimirá BD

b) Compilará e imprimirá DB

c) Compilará e imprimirá BDC

d) No compilará

3. ¿Qué pasaría si tratas de compilar y ejecutar el siguiente código Java? [4 puntos]

```
class Cantante {
    public static String cantar() { return "la"; }
}

public class Tenor extends Cantante {
  public static String cantar () { return "fa"; }
    public static void main (String [] args ) {
      Tenor t = new Tenor ();
      Cantante s = new Tenor ();
      System.out.println (t.cantar () + " " + s.cantar() );
  }
}
```

a) No compilará

b) Compilará e imprimirá fa la

c) Compilará e imprimirá la fa

d) Compilará e imprimirá fa fa

4. Este código contiene 5 errores. Para cada error indica:

- **La línea donde se encuentra el error (aproximado).**
- **Una breve descripción del error.**

```
 1  package com.navidad;
 2
 3  public class Elfo {
 4    private String nombre;
 5    private int nivelMagia;
 6
 7    public Elfo(String nombre){
 8       nombre = n;
 9       nivelMagia = 50;
10    }
11    public void trabajar(){
12       System.out.println("El elfo " + nombre
13         + " está preparando regalos");
14    }
15  }
```

```
package com.navidad;

public class ElfoConstructor extends Elfo {
   String herramienta;
   public ElfoConstructor(String nombre, int nivel,
                          String herramienta){
      super();
      this.herramienta = herramienta;
   }

   @Override
   public String trabajar(){
      return "El elfo constructor usa " + herramienta;
   }
}
```

```
1  package com.taller;
2  import com.navidad.Elfo;
3  public class TallerNavidad {
4      public static void main(String[] args) {
5          Elfo e = new ElfoConstructor("Bobby", 80, "martillo mágico");
6          e.trabajar(10);
7      }
8  }
```

5. Diagrama de clases: Sistema de Gestión del Evento InnovaAcción [20 puntos]

La universidad celebra anualmente **InnovaAcción**, un evento diseñado para promover la innovación y la resolución colaborativa de problemas. Para cada edición de este evento, la universidad necesita un sistema que registre toda la información clave.

Cada **Edición del evento** se celebra una vez al año y está definida por su **año**, una **fecha de inicio**, una **fecha de fin**, y tres **retos** específicos. Al concluir cada edición, se selecciona un único **equipo ganador**.

Los **Retos** son los desafíos centrales del evento. Cada edición presenta exactamente tres retos, que abordan distintas problemáticas. Cada reto tiene un **título**, una **descripción** y una **pregunta** que plantea el problema a resolver.

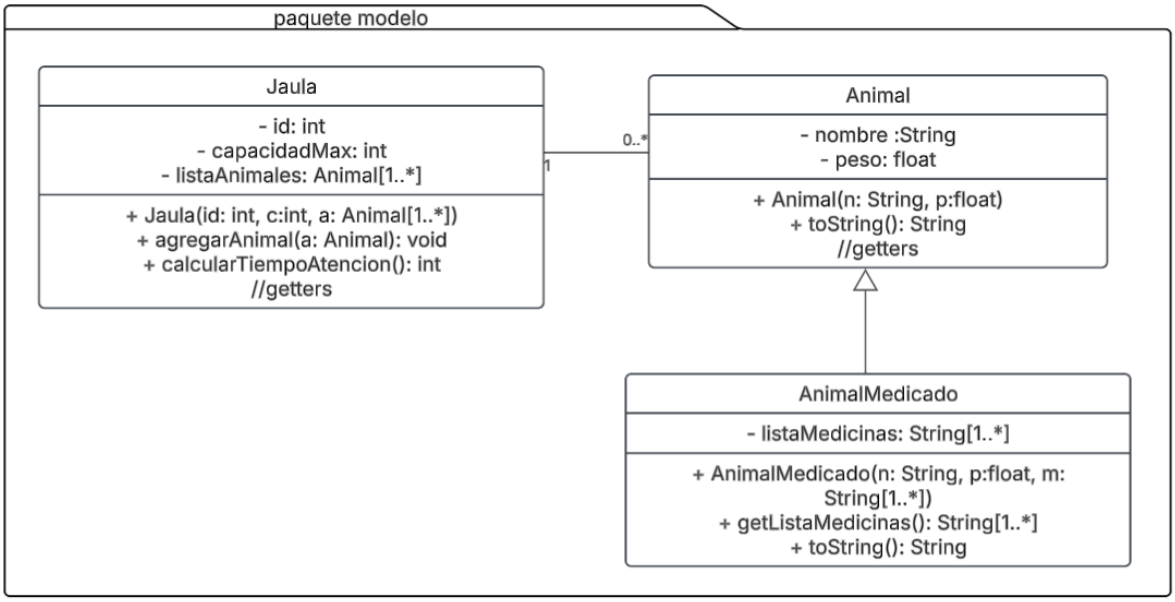
Los **Participantes** pueden ser tanto docentes como administrativos. Ambos comparten atributos como **nombre**, **correoInstitucional** y **departamento**. Además, los **Docentes** incluyen un **área académica**, mientras que los **Administrativos** tienen un **cargo**.

Los **Equipos** se forman entre dos hasta cuatro participantes (docentes o administrativos). Cada equipo propone un solo **proyecto** que debe estar asociado a uno de los retos de la edición en curso. Es importante destacar que un participante no puede formar parte de más de un equipo en la misma edición.

Finalmente, los **Proyectos** son las propuestas desarrolladas por los equipos. Cada proyecto se caracteriza por su **título**, una **descripción** y las **tecnologías utilizadas**. Cada proyecto debe estar vinculado a uno de los retos de la edición. Al finalizar el evento, se selecciona un **proyecto ganador**, el cual está directamente asociado al equipo que resultó ganador.

6. Tema de Desarrollo [57 puntos]

El refugio “Huellitas Felices” necesita un sistema para gestionar el cuidado de los animales que tienen. En el refugio existen 20 jaulas para alojar a los animales. Se necesita tener una funcionalidad que permita estimar el tiempo que necesita un veterinario para atender a todos los animales. Se le proporciona el siguiente diagrama de clases.



La clase Animal representa un animal en el refugio. Asuma que esta clase ya está totalmente implementada. El atributo peso representa un valor en Kg.

Desarrolle lo siguiente

1.- Clase AnimalMedicado

La clase **AnimalMedicado** representa un animal que requiere medicación constante. Implemente la clase de acuerdo con el diagrama.

A. Definición de paquete.
B. Importación de paquetes necesarios.
C. Definición de la clase.
D. Atributo listaMedicinas, un Arraylist de tipo String.
E. Constructor para inicializar las variables de instancia.
F. método getListaMedicinas() que devuelve la lista de medicinas.
G. Método toString que retorna lo mismo que el toString de Animal más la lista de medicinas.

2.- Clase Jaula

La clase **Jaula** representa un contenedor donde se pueden alojar varios animales.

A. En esta clase solamente implemente el método **calcularTiempoAtencion()**, el cual debe retornar el tiempo total de atención, en minutos, requerido por todas las mascotas que se encuentran en esa jaula. Para determinar dicho tiempo, considere que cada animal requiere 5 minutos de atención básica. Adicionalmente, si el **animal** es una instancia de **AnimalMedicado**, deberá sumar 2 minutos extra por cada medicina asignada.

Asuma que los demás métodos ya están implementados.

3.- Clase Main

A. Implemente el método estático **estimarAtencionTotal(Jaula[] arrJaulas)** que recibe un arreglo de tipo Jaula para calcular y mostrar el tiempo total de atención de todos los animales del refugio. Recuerde que la clase Jaula posee el método **calcularTiempoAtencion()**. Muestre el tiempo en formato **horas y minutos** (ej., si el total es 285 minutos, debe mostrarse como "4 horas y 45 minutos").
B. Complete el método main para:

   i. Crear un animal medicado con los siguientes datos:

      - Una lista con un solo elemento que tenga el texto “antibiótico”.
      - Nombre: Punchi
      - Peso: 3kg

   ii. Agregar este nuevo animal a la primera jaula.

```
public class Main {

    public static void main(String[] args) {

        Jaula[] arr = new Jaula[20];

        //se crean objetos jaula en todas las posiciones del arreglo

        //con 1 animal en cada jaula

        //escriba el código solicitado para este método









        //llamada a método para imprimir el tiempo requerido

        estimarAtencionTotal(arr);


    }

   //escribir método estimarAtencionTotal

}
```

Métodos ArrayList: add, clear, get, remove, set, size