Introducción a la Programación Orientada a Objetos

DCIC – UNS

28 de diciembre de 2022

Examen de promoción

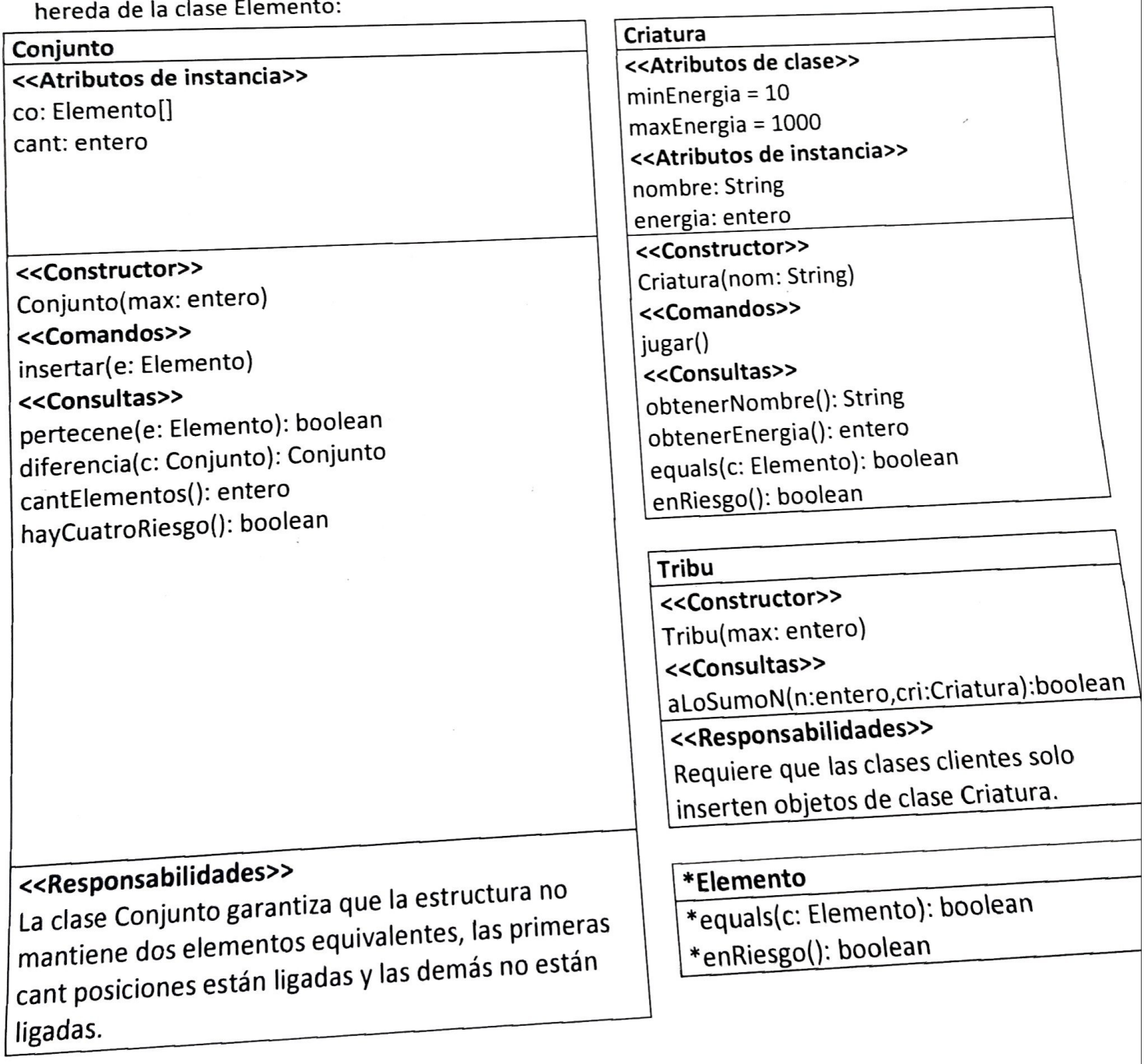
LA EVALUACIÓN CONSIDERA LA **CORRECTITUD, LEGIBILIDAD Y EFICIENCIA** DE LAS SOLUCIONES.

La interpretación del enunciado es parte del examen.

1. Defina el modelo computacional de la programación orientada a objetos y los conceptos de objeto, clase y tipo de dato abstracto.
2. Explique por qué la programación orientada a objetos favorece la productividad del software.
3. Explique el significado de las siguientes instrucciones que forman parte del constructor de una clase que extiende a JFrame:

```
1 JButton botonRojo = new JButton("Rojo");
2 OyenteBotonR ponerRojo = new OyenteBotonR();
3 botonRojo.addActionListener(ponerRojo);
```

4. Dado el siguiente diagrama de clase que modela una clase genérica Conjunto y la clase Criatura que hereda de la clase Elemento:



a) Implemente la clase Conjunto considerando las siguientes funcionalidades y responsabilidades:

- **Conjunto(max: entero)** Requiere max mayor a 0.
- **insertar(e: Elemento)** asigna *e* a la posición cant e incrementa el valor de esta variable.
- **pertenece(e: Elemento):boolean** retorna true si el conjunto mantiene un elemento equivalente a *e*.
- **diferencia(c: Conjunto): Conjunto** genera un nuevo conjunto que incluye a los elementos que resultan de la diferencia de conjuntos entre el conjunto que recibe el mensaje y el que viene por parámetro. Esto es, genera un conjunto que incluye los elementos que están en el conjunto que recibe el mensaje y no en el conjunto *c*, siempre comparando por estado interno.
- **hayCuatroRiesgo():boolean** retorna true si el conjunto que recibe el mensaje contienen una y solo una secuencia de exactamente cuatro elementos que ocupan posiciones consecutivas y verifican la propiedad enRiesgo().

b) Implemente la clase Elemento y la clase Criatura que extiende a Elemento.

- **Criatura(nom: String)** Requiere nom ligado, inicializa el atributo energía con el valor del atributo de clase maxEnergia.
- **jugar()** consume 1 unidad de energía, siempre y cuando este valor sea mayor a 0.
- **equals(c:Elemento) :boolean** compara en forma superficial. Requiere c ligada.
- **enRiesgo():boolean** retorna true si la energía de la criatura que recibe el mensaje es menor a minEnergia.

c) Implemente la clase Tribu que extiende a Conjunto.

- **Tribu(max: entero)** requiere max mayor a 0.
- **aLoSumoN(n:entero,cri:Criatura):boolean** retorne true si el conjunto contiene a lo sumo *n* criaturas con la energía mayor a la energía de *cri*.

d) Implemente una clase tester que valide los métodos *insertar* y *aLoSumoN*. Para mostrar el conjunto puede agregar en la clase Conjunto un método *obtenerElemento(p:entero):Elemento* que retorne el elemento que ocupa la posición *p*, asumiendo que las clases clientes de Conjunto garantizan que *p* es una posición válida.

e) Envíe el proyecto empaquetado a ipoo.dcic@gmail.com, asegúrese de que su nombre figura en el proyecto