Control 3 Paradigmas de Programación (2022-1)

Junio

Duración: 45 minutos

NOMBRE:_____

RUT: _____

Profesor: _____

(30 pts) Pregunta I (Procure responder todos los ítems de esta pregunta en esta misma hoja (puede usar ambos lados).

De acuerdo al siguiente código implementado en Java (asuma que funciona, aún si encuentra algún error léxico o sintáctico), responda las siguientes preguntas:

1. **(10 pts)** Indique en el código que conceptos de la Programación Orientada a Objetos son utilizados.
2. **(10 pts)** Explique el contexto del código (no más de 5 líneas).
3. **(10 pts)** Elabore un diagrama de clase que exprese todos los aspectos estructurales. (entidades, atributos, comportamientos y relaciones) contenidos en él. Procure señalar con comentarios en el diagrama los tipos de relaciones empleadas.

```
public abstract class Usuario {   //Clase Abstracta
  String rut; //Atributos
  String nombre;
  String contrasena;
  public abstract void login(String rut, String contrasena);   //Método abstracto
}

public class Especialista extends Usuario {   //Clase y herencia
  //Atributos
  String especialidad;
  ArrayList<Paciente> listaPacientes;           //Agregación
//Métodos
  public void atenderPaciente(Paciente p) { /* */ }
  public void login(String rut, String contrasena)  { /* */ }
}

public class Paciente extends Usuario {      //Clase y herencia
//Atributos
  String diagnostico;
  ArrayList<Medicamento> prescripcion;   //Agregación
  ArrayList<String> fichaMedica;
//Métodos
  public String toString() { /* */ }   //Sobreescritura
  public void login(String rut, String contrasena)  { /* */ }
}

public class Medicamento {  //Clase
//Atributos
  String nombre;
  float dosis;
}
```

```
public class Farmacia {   //clase
//Atributos
  ArrayList<Medicamento> stock;  //Agregación
//Métodos
  public void entregarMedicamento (Paciente p) { /* */ }  //Dependencia
  public void renovarStock() { /* */ } //sobrecarga
  public void renovarStock(Medicamento m) { /* */ }
}
```

a)
Identifica en al menos una de las clases:

Concepto	**Puntaje** **Evaluar con** **0: No Cumple** **1: Cumple** **(binario)**	**Ponderación**
Clase		5%
Clase Abstracta		5%
Método Abstracto		5%
Atributos		5%
Métodos		5%
Agregación		20%
Dependencia		20%
Sobreescritura		15%
Sobrecarga		15%
Herencia		5%

Ptje Item: 10*SumatoriaPonderada(Puntaje,Ponderación)

b) Para la pregunta solo se pide describir el contexto, no una narrativa completa. A modo general, el contexto es un sistema que permite a una farmacia entregar medicamentos a pacientes en base a la prescripción otorgada por un especialista. (con esto es suficiente)

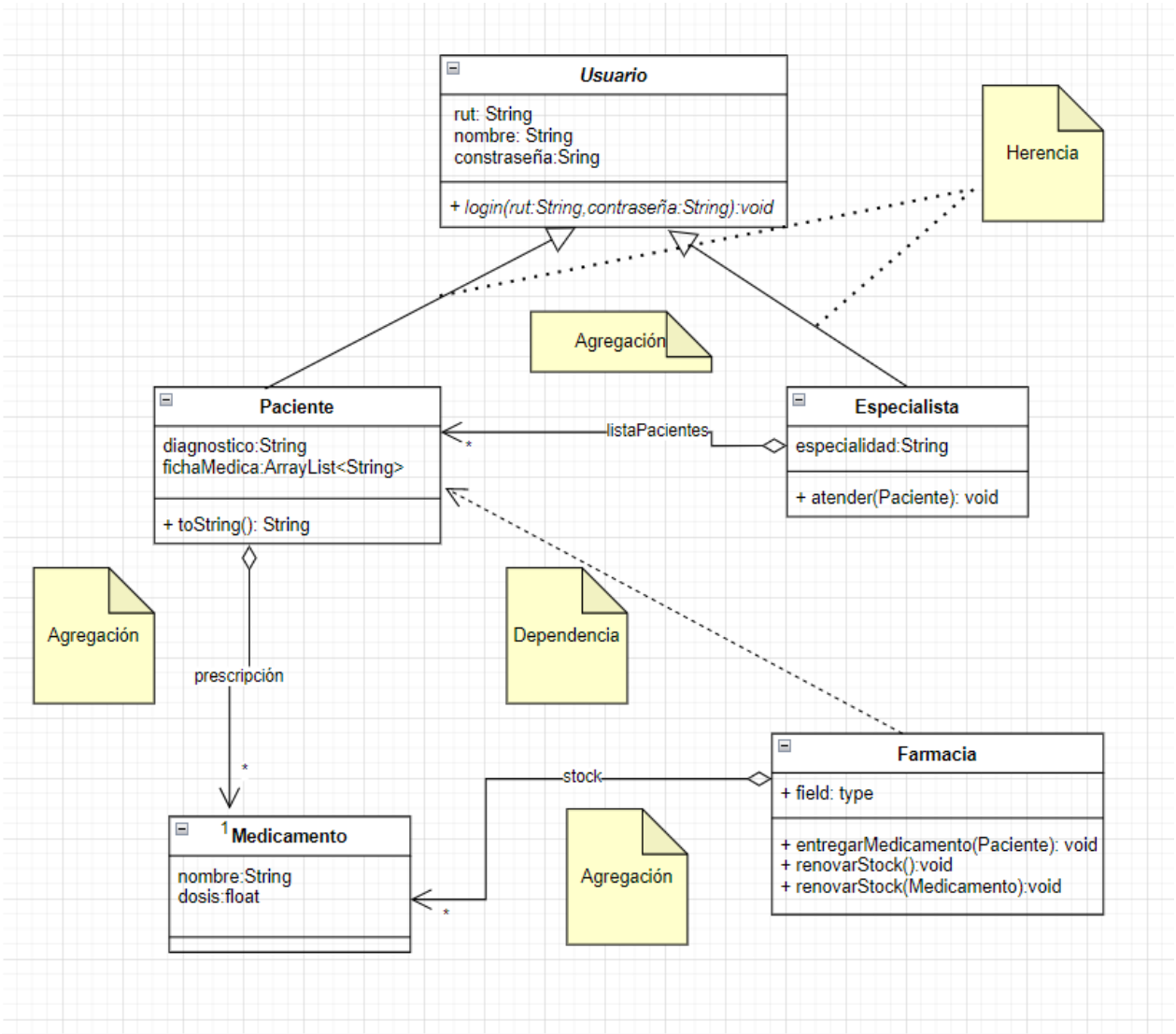
Si alguién escribe la narrativa, también es aceptable y sería algo como:
Un sistema en el que existen dos tipos de usuarios: Pacientes y Especialistas. Los usuarios quedan identificados por un rut, nombre y contraseña. Ambos tipos de usuarios inician sesión con el rut y contraseña, pero lo hacen de forma diferente. Los Especialistas tienen una especialidad y tienen varios pacientes vinculados a los que además atienden. Los Pacientes, tiene un diagnóstico, una lista de medicamentos prescritos y una ficha médica. Los medicamentos quedan identificados por un nombre y dosis. Por último, la farmacia maneja un stock de medicamentos, entrega los medicamentos y renueva su stock.

Para evaluar este item considere:

0: No Cumple
0.5: Incompleto (cubre parcialmente el contexto o narrativa asociada al código).
1: Cumple

Ptje Item: 10*PrjeAsignado

c)



Elemento a evaluar	Puntaje Evaluar con 0: No Cumple 0.5: incompleto 1: Cumple	
Expresa 5 clases		10%
Clase Abstracta y método login abstracto (cursiva) o bien usa estereotipo <<abstract>>		10%
Agregación Paciente con 0 o muchos Medicamento (*). Relación se llama prescripción y no redunda poniendo el atributo prescripción en la clase.		10%

Agregación Especialista con 0 o muchos Paciente (*). La relación se llama listadoPaciente sy no redunda poniendo el atributo listadoPaciente en la clase.		10%
Lista atributos en cada clase		10%
Lista métodos en cada clase		10%
Expresa relación herencia entre Paciente y Usuario		10%
Expresa relación herencia entre Especialista y Usuario		10%
Dependencia entre Farmacia y Paciente		10%
Agregación entre Farmacia y Medicamento llamada “stock”. No redunda indicando atributo stock dentro de la clase Farmacia		10%

Ptje Item: 10*SumatoriaPonderada(Puntaje,Ponderación)

Control 3 Paradigmas de Programación (2022-1)

Junio

Duración: 45 minutos

NOMBRE:_____

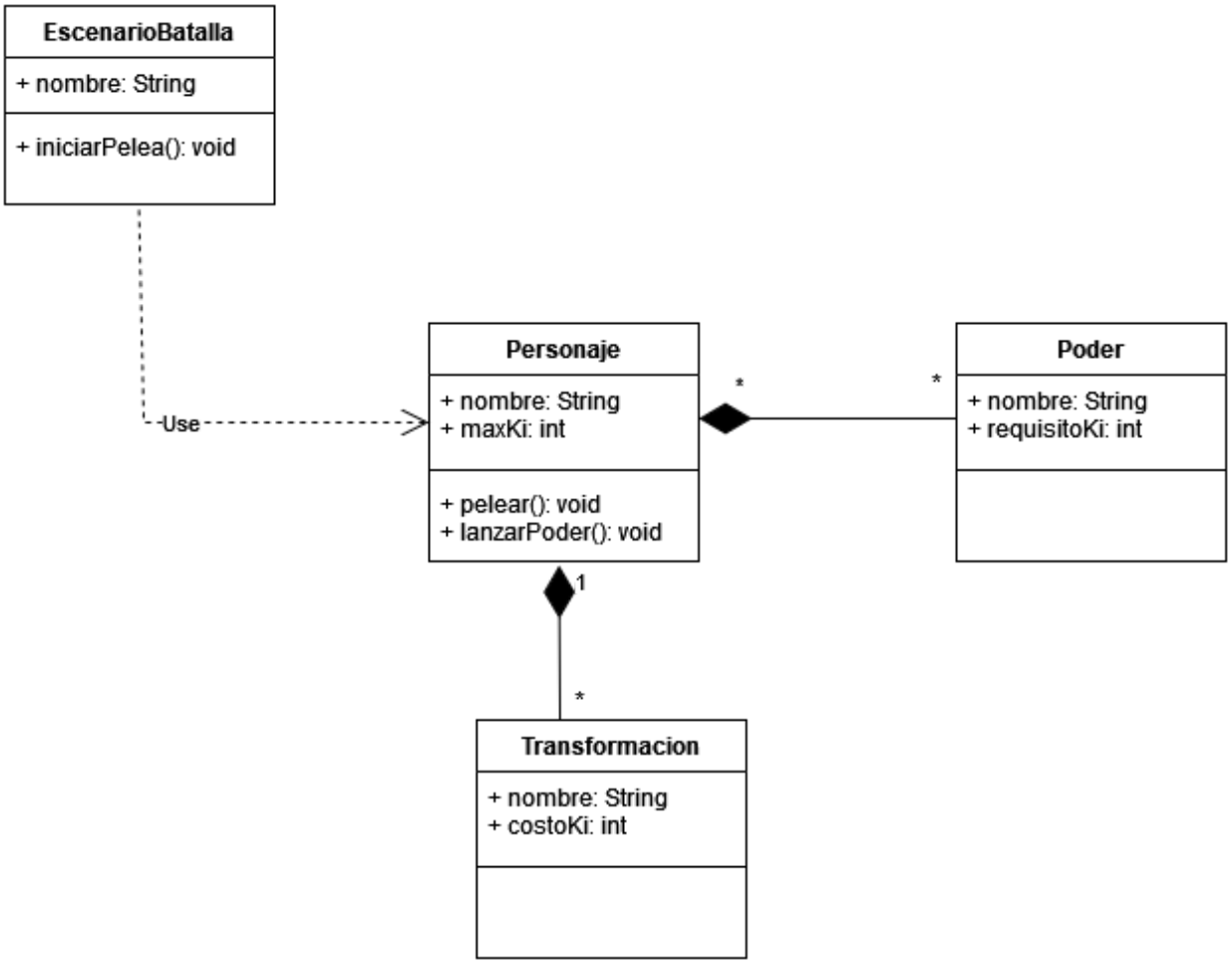
RUT: _____

Profesor: _____

(30 pts) Pregunta II (Procure responder todos los ítems de esta pregunta en esta misma hoja (puede usar ambos lados).

De acuerdo al siguiente diagrama de clase UML, responda las siguientes preguntas:

1. **(10 pts)** Indique en el diagrama que conceptos de la Programación Orientada a Objetos son utilizados.
2. **(20 pts)** Construya un esqueleto de código en pseudo Java basado en el diagrama de clase. Procure expresar las clases, atributos, métodos y relaciones. Para los métodos solo declare la firma (y no la lógica de éstos), dejando el cuerpo sólo como { /***/ }.



a) Identifica:

Concepto	**Puntaje** **Evaluar con** **0: No Cumple** **0.5: incompleto** **1: Cumple**	**Ponderación**
Clases		20%
Atributos		10%
Métodos		10%
Dependencia entre EscenarioBatalla y Personaje		20%
Composición entre Personaje y Poder		20%
Composición entre Personaje y Transformación.		20%

Ptje Item: 10*SumatoriaPonderada(Puntaje,Ponderación)

b) Código

```
class EscenarioBatalla
{
    public String nombre;
    public void iniciarPelea()
    { …   //[opcional]  aquí podría mostrar uso de Personaje, para ilustrar dependencia
      }
}

class Personaje
{
    public String nombre;
    public int maxKi;
    public ArrayList<Poder> poderes = new ArrayList<Poder>();   //Esto no es composición, ya que lo que instancia es el ArrayList y no el Poder
    public ArrayList<Transformacion> transformaciones = new ArrayList<Transformacion>(); //Esto no es composición, ya que lo que instancia es el ArrayList y no la Transformación
    Personaje()
    {
             poderes.add(new Poder());  //solo para ilustrar composición
             transformaciones.add(new Transformacion());  //solo para ilustrar composición
    }

    public void pelear() {...}
    public void lanzarPoder() {...}
}

class Transformacion
{
   public String nombre;
```

```
   public int costoKi;
}
```

Cubre:

Elemento a evaluar	**Puntaje** **Evaluar con** **0: No Cumple** **1: Cumple** **(binario)**	**Ponderación**
Implementa 4 Clases		20%
Expresa los Atributos en cada clase		20%
Expresa los Métodos en cada clase		20%
Expresa la composición desde Personaje a Poder		20%
Expresa la composición desde Personaje a Transformación		20%

Ptje Item: 20*SumatoriaPonderada(Puntaje,Ponderación)