Ingeniería Civil en Computación e Informática
Ingeniería en Tecnologías de Información

ESCUELA DE INGENIERÍA UCN COQUIMBO

Prueba Sumativa Nº 1

Programación Orientada a Objetos – 21 de Octubre 2023

Nombre		RUT	
Paralelo	( ) APaolini ( ) ERoss		

Antecedentes generales:

Puntaje total de la prueba/Puntos para nota aprobatoria(4.0)	100 puntos 60 puntos	**Puntaje Obtenido**	
Duración de la prueba	6 horas	**Nota final**	
Resultados de Aprendizaje a evaluar	1. Aplicar técnicas de ingeniería de software en la creación de software legible, mantenible y testeable. 2. Aplicar técnicas de programación orientada al objeto en la resolución de problemas. 3. Crear tipos de datos abstractos con bajo acoplamiento entre la implementación y su comportamiento que permitan la resolución problemas. 4. Analizar las relaciones causa efecto de los procesos en estudio. 6. Seleccionar los procesos, técnicas y herramientas adecuados de acuerdo a los requerimientos.		
Fecha de entrega de resultados	3 diciembre 2023		

Instrucciones:

1. Esta evaluación tiene 4 páginas (incluyendo la portada). Compruebe que dispone de todas las páginas.
2. Lea la prueba completamente DOS veces antes de hacer cualquier pregunta
3. Durante la prueba no se puede utilizar: teléfono móvil, calculadora, apuntes. Está prohibido intentar conectarse a internet de cualquier manera (excepto a CampusVirtual, y solo para subir su solución). Si es sorprendido obtendrá la calificación mínima. Tampoco puede utilizar dispositivos de almacenamiento externos o cualquier otro dispositivo relojes inteligentes, ábacos, etc.
4. Una prueba respondida correctamente en un 60%, de acuerdo a las ponderaciones asignadas, corresponde a una nota 4,0.
5. Solamente se pueden realizar preguntas durante los primeros 15 minutos de la prueba. Solo se responderán preguntas respecto a los enunciados a viva voz.
6. La prueba es individual, cualquier sospecha de copia será calificada con la nota mínima y el caso será remitido al comité de ética.
7. En su espacio personal no debe haber nada más que hojas de papel en blanco, lápiz, goma.
8. El resto de sus implementos debe guardarlos dentro de su mochila/bolso y ésta debe posicionarse al frente debajo de la pizarra. Si leyó hasta este punto, felicidades, para saber que lo hizo dibuje una estrella al final de esta página.

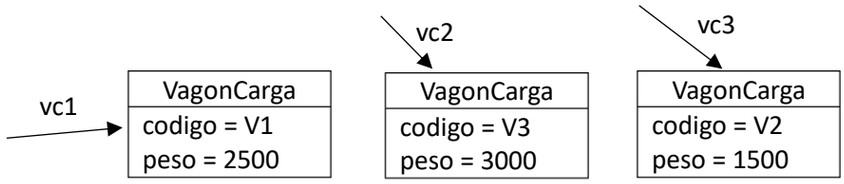
Acepto las condiciones firmando: _____

Problema 1. Ruteo (40 puntos)

Rutee el siguiente código, describiendo qué sucede desde la perspectiva de los objetos involucrados. Por cada instrucción del main que genere un cambio, indique cuáles líneas de código fueron las que lo generaron. Puede agrupar instrucciones.

Ejemplo:

Línea 3, 4 y 5:




```
1. public class Main {
2.     public static void main(String[] args) {
3.         VagonCarga vc1 = new VagonCarga("V1", 2500);
4.         VagonCarga vc2 = new VagonCarga("V3", 3000);
5.         VagonCarga vc3 = new VagonCarga("V2", 1500);
6.         Tren<VagonPasajeros> metro = new Tren<>();
7.         Tren<VagonCarga> cap = new Tren<>();
8.         for (int i = 1; i < 5; i++) {
9.             metro.agregarVagones(new VagonPasajeros("X" + i, 0));
10.            cap.agregarVagones(vc1);
11.            System.out.println(metro.getVagon(i - 1));
12.            System.out.println(cap.getVagon(i - 1));
13.        }
14.        int[] personas = { 30, 19, 15, 5 };
15.        VagonPasajeros.setCapacidadMax(20);
16.        for (int i = 0; i < metro.cantidadVagones(); i++) {
17.            metro.getVagon(i).subir(personas[i]);
18.        }
19.        metro.getVagon(2).bajar(5);
20.        cap.eliminarVagones(1);
21.        metro.getVagon(1).subir(8);
22.        metro.getVagon(0).bajar(3);
23.        metro.getVagon(3).bajar(6);
24.        cap.agregarVagones(vc2);
25.        metro.getVagon(2).bajar(3);
26.        cap.eliminarVagones(1);
27.        cap.agregarVagones(vc3);
28.        for (int i = 0; i < metro.cantidadVagones(); i++) {
29.            System.out.println(metro.getVagon(i));
30.        }
31.        for (int i = 0; i < cap.cantidadVagones(); i++) {
32.            System.out.println(cap.getVagon(i));
33.        }
34.    }
35. }
36.
37. import java.util.ArrayList;
38. import java.util.List;
39. public class Tren<T> {
40.     private List<T> vagones = new ArrayList<>();
41.     public T getVagon(int index) {
42.         return vagones.get(index);
43.     }
44.
45.     public void agregarVagones(T vagon) {
46.         this.vagones.add(vagon);
47.     }
48.     public void eliminarVagones(int index) {
49.         this.vagones.remove(index);
50.     }
51.     public int cantidadVagones() {
52.         return this.vagones.size();
53.     }
54. }
55. public class VagonPasajeros {
56.     public String codigo;
57.     public int cantidadP;
58.     public static int capacidadMax;
59.     public VagonPasajeros(String codigo, int cantidad) {
60.         this.codigo = codigo;
61.         this.cantidadP = cantidad;
62.     }
63.     @Override
64.     public String toString() {
65.         return "[Vagon: " + codigo + ", Pasajeros: " + cantidadP + "]";
66.     }
67.     public int getCantidad() {
68.         return cantidadP;
69.     }
70.     public void bajar(int c) {
```

```
71.         if (cantidadP >= c) {
72.             this.cantidadP -= c;
73.         } else {
74.             cantidadP = 0;
75.         }
76.     }
77.     public void subir(int c) {
78.         if (cantidadP + c <= capacidadMax) {
79.             this.cantidadP += c;
80.         } else {
81.             cantidadP = capacidadMax;
82.         }
83.     }
84.     public static void setCapacidadMax(int capacidad) {
85.         capacidadMax = capacidad;
86.     }
87. }
88. public class VagonCarga {
89.     public String codigo;
90.     public int peso;
91.     public VagonCarga(String codigo, int peso) {
92.         this.codigo = codigo;
93.         this.peso = peso;
94.     }
95.     @Override
96.     public String toString() {
97.         return "[Vagon: " + codigo + ", Carga: " + peso + " kg]";
98.     }
99. }
100.
```

Problema 2. (60 puntos)

En la UCN hay muchas personas, que cumplen diferentes roles: hay personas funcionarias, hay profesores, hay estudiantes. Pero, puede suceder que una persona cumpla más de un rol en forma simultanea. Además, cuando una persona tiene un "rol", lo tiene durante una cierta cantidad de tiempo, o sea, tienen una fecha de inicio y una fecha de término. Las fechas se identifican por su año, mes y día.

Cuando las personas cumplen un "rol" en la UCN, automáticamente quedan relacionadas con una unidad de la UCN. Por ejemplo, cuando se es "estudiante", se asocian a una carrera (considere que las carreras son unidades también). Cuando son funcionarios a una unidad (por ejemplo, "Servicio de Obras", "Escuela de Ingeniería"). Los profesores se asocian a una unidad también.

Qué debe hacer

1. Construya un programa en Java que presente un menú de opciones que permita realizar las siguientes acciones:
    a. Agregar una persona
    b. Agregar una unidad
    c. Asociar una persona a una unidad entre ciertas fechas
    d. Reportabilidad
        i. Dada una fecha (año, mes, día), indicar el rol que está cumpliendo cada persona registrada, indicando la unidad
        ii. Dada una fecha (año, mes, día), indicar la cantidad de personas en cada unidad.
        iii. Dado un RUT de una persona, indicar su "historia de roles", o sea, los roles que ha cumplido en la UCN, ordenados de menor a mayor por fecha.
        iv. Dada una unidad, mostrar el "historial de roles", o sea, los diferentes roles que han trabajado en esa unidad, especificando a la persona que cumplió ese rol.
        v. Estadísticas generales:
            1. La persona con más antigüedad en la UCN
            2. La persona que ha tenido más roles

Debe entregar:

- Modelo de Dominio (10%)
- Diagrama de Clases (20%)
- Código Java (70%)

Consideraciones:

- Debe usar orientación al objeto
- En el diagrama de clases debe especificar **TODO**.
- Tanto el modelo del dominio, como el diagrama de clases debe escribirlos en papel y entregarlos junto a la prueba.
- El código fuente debe comprimirlo en un solo archivo .zip y subirlo a Campus Virtual.
- Hojas sin nombre no se revisarán.