Prueba Resiliencia Sumativa Nº 1

Programación Orientada a Objetos – 18 de Noviembre 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	3 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	2 diciembre 2023		

Instrucciones:

1. Esta evaluación tiene 5 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.

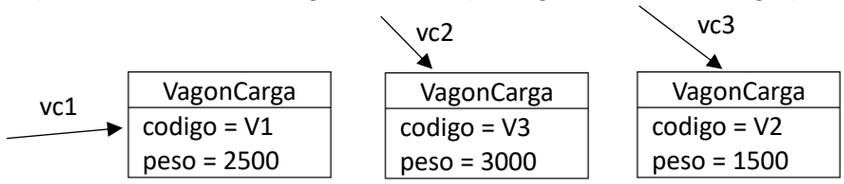
Acepto las condiciones firmando: _____

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;
```

```
        }
    }
    public void subir(int c) {
        if (cantidadP + c <= capacidadMax) {
            this.cantidadP += c;
        } else {
            cantidadP = capacidadMax;
        }
    }
    public static void setCapacidadMax(int capacidad) {
        capacidadMax = capacidad;
    }
}
public class VagonCarga {
    public String codigo;
    public int peso;
    public VagonCarga(String codigo, int peso) {
        this.codigo = codigo;
        this.peso = peso;
    }
    @Override
    public String toString() {
        return "[Vagon: " + codigo + ", Carga: " + peso + " kg]";
    }
}

```

Problema 2. (60 puntos)

En la UCN nuevamente nos pidieron ayuda para evaluar varias situaciones que se están dando. En particular, nos han entregado 3 archivos:

personas.txt	cursos.txt	estados_finales.txt
persona1,1 persona2,2 persona3,3 persona4,4	Curso1,AAA,20 Curso2,BBB,32 Curso3,CCC,22	1,BBB,4,10 1,AAA,2.4,3 2,AAA,6.6,20 3,BBB,7,30 3,AAA,4,7

personas.txt contiene el listado de personas que están matriculadas en cierta carrera. Los datos son: nombre de la persona, número de matrícula.

cursos.txt contiene el listado de cursos que se están dictando en esa carrera. Los datos son: nombre del curso, código del curso, cantidad de clases realizadas en el curso.

estados_finales.txt contiene el estado final de las personas en los cursos donde que tomaron. Los datos son: número de matrícula, código del curso, promedio final (de esa persona en el curso) y cantidad de clases asistidas (de esa persona en el curso).

Su misión es construir una aplicación que realice las siguientes acciones:

1. Cargar los 3 archivos que se entregan en estructuras de datos apropiadas.

2. Después de cargados los datos, presentar al usuario un menú que le permita realizar las siguientes acciones:

1. Estadísticas por cursos	Muestra estadísticas generales de todos los cursos cargados (vea el ejemplo para más detalles)
2. Estadísticas por personas	Muestra estadísticas generales de todas las personas cargadas (vea el ejemplo para más detalles)
3. Estadísticas por persona	A partir de un número de matrícula, muestra estadísticas de la persona (vea el ejemplo para más detalles)
4. Estadísticas por curso	A partir de un código de un curso, muestra estadísticas del curso (vea el ejemplo para más detalles)
5. Detectar anomalías	Escribe a la pantalla datos anómalos encontrados en los datos cargados. En particular, personas sin cursos, cursos sin personas, y personas con promedio general menor a 4 (vea el ejemplo para más detalles)
0. Salir	Termina la ejecución del programa

Para tomar en cuenta:

- Se asume que todo el análisis, y los datos entregados, corresponden a lo que sucede en **UN** semestre en particular.
- No es un error que existan personas sin cursos, ni que haya cursos sin personas.
- No se preocupe del control de errores en el ingreso de datos.

Debe entregar:

- Modelo de Dominio (10%)
- Diagrama de Clases (15%)
- Código Java (75%)

Consideraciones:

- Debe usar orientación al objeto.
- No se deben utilizar ciclos dentro de ciclos. Use funciones para hacerse la vida más fácil.
- En el diagrama de clases debe especificar TODO. No es buena práctica especificar todo público, ni tampoco crear getters o setters para todos los atributos.
- 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.
- Archivos .txt de ejemplo se encuentran en CampusVirtual

Ejemplo de ejecución:

```
Leídas 4 personas
Leídos 3 cursos
Menú:
1. Estadísticas por cursos
2. Estadísticas por personas
3. Estadísticas por persona
4. Estadísticas por curso
5. Detectar anomalías
0. Salir
> 1
Estadísticas cursos:
       AAA  : Curso1  : prom=4.333 prom%asist=50.0
       BBB  : Curso2  : prom=5.5  prom%asist=62.5
       CCC  : Curso3  : Sin estudiantes

Menú:
1. Estadísticas por cursos
2. Estadísticas por personas
3. Estadísticas por persona
4. Estadísticas por curso
5. Detectar anomalías
0. Salir
> 2
Estadísticas personas:
       1: persona1: prom=3.2 prom%asist=23.125
       2: persona2: prom=6.6 prom%asist=100.0
       3: persona3: prom=5.5 prom%asist=64.375
       4: persona4: Sin cursos

Menú:
1. Estadísticas por cursos
2. Estadísticas por personas
3. Estadísticas por persona
4. Estadísticas por curso
5. Detectar anomalías
0. Salir
> 3
Ingresa código de persona: 1
1: persona1
       Lista de cursos:
       Nombre              Nota Asist %Asist
       Curso2              4.0  10    31.25
       Curso1              2.4  3     15.0
       Promedio general : 3.2

Menú:
1. Estadísticas por cursos
2. Estadísticas por personas
3. Estadísticas por persona
4. Estadísticas por curso
5. Detectar anomalías
0. Salir
> 3
Ingresa código de persona: 4
4: persona4
       Persona no tuvo cursos

Menú:
1. Estadísticas por cursos
2. Estadísticas por personas
3. Estadísticas por persona
4. Estadísticas por curso
5. Detectar anomalías
0. Salir
> 4
Ingrese código de curso: BBB
BBB: Curso2
       Nómina de estudiantes:
       Nombre              Nota Asist %Asist
       persona1            4.0  10    31.25
       persona3            7.0  30    93.75
       Promedio del curso : 5.5

Menú:
1. Estadísticas por cursos
2. Estadísticas por personas
3. Estadísticas por persona
4. Estadísticas por curso
5. Detectar anomalías
0. Salir
> 4
Ingrese código de curso: CCC
CCC: Curso3
       El curso no tuvo estudiantes

Menú:
1. Estadísticas por cursos
2. Estadísticas por personas
3. Estadísticas por persona
4. Estadísticas por curso
5. Detectar anomalías
0. Salir
> 5
Detección de anomalías:
       Personas sin inscripción:
              4 : persona4
       Cursos sin inscripción:
              CCC : Curso3
       Personas con promedio menor a 4:
              1 : persona1
Menú:
1. Estadísticas por cursos
2. Estadísticas por personas
3. Estadísticas por persona
4. Estadísticas por curso
5. Detectar anomalías
0. Salir
> 0
```