Prueba Sumativa 1

Programación Orientada a Objetos – 23 de abril 2024

Nombre		RUT	
Paralelo	( ) APaolini ( ) MMoraga ( ) 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	7 mayo 2024		

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, apuntes. Está prohibido intentar conectarse a internet de cualquier manera (excepto a Campus Virtual, 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 con las ponderaciones asignadas, corresponde a una nota 4,0.
5. La prueba es individual, cualquier sospecha de copia será calificada con la nota mínima y el caso será remitido al comité de ética.
6. En su espacio personal no debe haber nada más que hojas de papel en blanco, lápiz, goma.
7. 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 un paralelepípedo al final de esta página.
8. Los estudiantes quienes se les compruebe falta de honestidad académica o cualquier otro acto contrario a las normas de permanencia universitaria o al espíritu universitario, serán sancionados, según sea la gravedad de la falta, con medidas desde la amonestación verbal hasta la suspensión o pérdida de la condición de estudiante, los estudiantes expulsados no podrán volver a ingresar a ninguna carrera, programa o curso de la institución. El estudiante que incurriere en falta de honestidad, durante la realización de un proceso evaluativo, será calificado con la nota mínima 1,0.

Acepto las condiciones firmando: _____

Problema 1. Ruteo (30 puntos)

Rutee el siguiente código, graficando 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 → vc1 → VagonCarga (codigo = V1, peso = 2500)

```java
public class App {

    public static void main(String[] args) {
        ClubDeYates yates = new ClubDeYates();
        Velero v1 = new Velero("Vagabundo 1", 502);
        Velero.asignarMembresia(20000);
        Velero v2 = new Velero("El Ucenino", 1281);
        Velero v3 = new Velero("Pirata Granate", 801);
        Velero v4 = new Velero("Vagabundo 2", 602);
        yates.atracar(v1, 12);
        yates.zarpar(v2);
        yates.zarpar(v2);
        yates.atracar(v3, 5);
        yates.zarpar(v3);
        yates.atracar(v4, 8);
        Velero.asignarMembresia(30000);
        yates.atracar(v4, 12);
        yates.atracar(v3, 15);
        yates.atracar(v2, 10);
        yates.zarpar(v2);
        yates.atracar(v2, 5);
        yates.zarpar(v1);
        yates.imprimir();
    }
}
```

```java
public class Velero {
    private String nombre;
    private int codigo;
    private int costoMantenimiento = 0;
    private static int costoMembresia;

    public Velero(String nombre, int codigo) {
        this.nombre = nombre;
        this.codigo = codigo;
    }

    public void calcularMembresia(int dias) {
        costoMantenimiento = getCostoMantenimiento() + costoMembresia * dias;
    }

    public static void asignarMembresia(int costo) {
        costoMembresia = costo;
    }

    public String getNombre() {
        return nombre;
    }

    public int getCostoMantenimiento() {
        return costoMantenimiento;
    }
}
```

```java
public class ClubDeYates {
    public Velero[] puerto = new Velero[5];
    public Velero[] bahia = new Velero[5];

    public void zarpar(Velero v) {
        for (int i = 0; i < bahia.length; i++) {
            if (bahia[i] == v) {
                break;
            } else if (bahia[i] == null) {
                bahia[i] = v;
                break;
            }
        }
        for (int i = 0; i < puerto.length; i++) {
            if (puerto[i] == v) {
                puerto[i] = null;
                break;
            }
        }
    }

    public void atracar(Velero v, int dias) {
        for (int i = 0; i < puerto.length; i++) {
            if (puerto[i] == v) {
                v.calcularMembresia(dias);
                break;
            } else if (puerto[i] == null) {
                puerto[i] = v;
                v.calcularMembresia(dias);
                break;
            }
        }
        for (int i = 0; i < puerto.length; i++) {
            if (bahia[i] == v) {
                bahia[i] = null;
                break;
            }
        }
    }

    public void imprimir() {
        System.out.println("Zarpe");
        for (int i = 0; i < bahia.length; i++) {
            if (bahia[i] == null) {
                System.out.print("null,");
            } else {
                System.out.print(bahia[i].getNombre() + ",");
            }
        }
        System.out.println();
        System.out.println("Atraque");
        for (int i = 0; i < puerto.length; i++) {
            if (puerto[i] == null) {
                System.out.print("null,");
            } else {
                System.out.print(puerto[i].getNombre() + ": " + puerto[i].getCostoMantenimiento() + ",");
            }
        }
        System.out.println();
    }
}
```

Problema 2. Mapaches celulares (70 puntos)

En un cierto universo paralelo, los animales que nosotros conocemos como mapaches se comportan de un modo diferente y especial. En primer lugar, su mundo es una cuadrícula, donde cada celda de la cuadrícula tiene una posición dada por una fila y columna, y en cada celda del mundo solamente puede existir un mapache.

Cuando el mundo se inicia, los mapaches se crean tomando en cuenta dos archivos: el archivo `mapaches.txt` lista todos los mapaches que existen en el mundo, de la siguiente forma:

Archivo	Explicación
6 2,PEDRO,1500 30,RENATA,2000 501,FRANCISCA,1100 6,CRISTINA,900	El primer número indica la cantidad de mapaches del mundo. Cada línea siguiente especifica a un mapache, indicando su RUT [entero], su nombre [string] y su masa (en gramos) [entero]

Además, el mundo se inicializa leyendo el archivo `mapaches_posiciones.txt`, que indica dónde vive cada mapache:

Archivo	Explicación
5 0,0,0,0,0 0,0,0,30,501 0,0,0,6,2 0,0,0,0,0 0,0,0,0,0	El primer número N indica la cantidad de filas y columnas del mundo. Después, hay N filas, cada una con N columnas, y cada número indica el RUT del mapache que vive en dicha celda. Si el número es CERO, significa que nadie vive ahí.

Lo que usted tiene que construir es un juego en que dos jugadores puedan simular lo que pasará al mover los mapaches por el mundo, y poder saber quién gana el juego. Las reglas de este juego son simples:

1. Existen solo 2 jugadores, que juegan por turnos, alternadamente. El primero jugador es el "`jugador 0`", y el segundo jugador es el "`jugador 1`".
2. En cada turno de un jugador, éste hace una acción. Las acciones posibles son:
   - Rendirse
   - Mover un mapache
   - Alimentar un mapache
   - Ver la información de un mapache

Después de que un jugador realiza una acción, su turno termina y le toca al otro jugador realizar su jugada. El juego continúa hasta que sucede alguna de estas situaciones:

- Un jugador se rinde. En ese caso, automática gana el otro jugador.
- Después de realizada una acción, solo queda un mapache en el mundo. En ese caso el jugador gana.

Cuando se realiza el movimiento de un mapache, el juego le debe preguntar al jugador por la dirección en la que quiere mover el mapache: Norte, Sur, Oeste o Este. Si la dirección de destino es válida (o sea, está dentro del mundo cuadriculado), el mapache se moverá. Pero, si la celda ya tenía un ocupante, los mapaches batallarán y solo habrá un ganador: el mapache con mayor masa (asuma que siempre habrá un ganador, y nunca empates).

Cuando un jugador alimenta a un mapache, éste gana un 10% de su masa actual.

Debe entregar:

- Diagrama del 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.
- El código fuente debe comprimirlo en un solo archivo .zip y subirlo a Campus Virtual.
- Archivos .txt de ejemplo se encuentran en CampusVirtual.
- Los diagramas de clases debe escribirlos en papel y entregarlos junto a la prueba.
- Hojas sin nombre no se revisarán.
- Debe usar referencias

Ejemplos de ejecución:

Revise el archivo `ejemplos.txt`