**Ingeniería en Tecnologías de Información**
**Ingeniería Civil en Computación e Informática**
**Ingeniería Civil Industrial**

# Prueba Sumativa 1

## Programación Orientada a Objetos – 03 de octubre 2024

| | | | |
|---|---|---|---|
| **Nombre** | | **RUT** | |
| **Paralelo** | ( ) APaolini ( ) MMoraga | | |

## Antecedentes generales:

| | | | |
|---|---|---|---|
| Puntaje total de la prueba/Puntos para nota aprobatoria (4.0) | 100 puntos<br>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.<br>2. Aplicar técnicas de programación orientada al objeto en la resolución de problemas.<br>3. Crear tipos de datos abstractos con bajo acoplamiento entre la implementación y su comportamiento que permitan la resolución problemas.<br>4. Analizar las relaciones causa efecto de los procesos en estudio.<br>6. Seleccionar los procesos, técnicas y herramientas adecuados de acuerdo a los requerimientos. | | |
| Fecha de entrega de resultados | 17 octubre 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: teléfonos móviles, 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 como relojes inteligentes, tables, audífonos, 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 y 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 una palta 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: _____

**Ingeniería en Tecnologías de Información**
**Ingeniería Civil en Computación e Informática**
**Ingeniería Civil Industrial**

## Problema 1. Ruteo (30 puntos)

Rutee el siguiente código e indique la salida esperada. Dibuje la **situación final** de los objetos involucrados al terminar la ejecución. No es necesario indicar todo el paso a paso.

```java
public class Main {

    public static void main(String[] args) {
        Wallet w = new Wallet();
        Card c1 = new Card(1);
        Card c2 = new Card(2);
        Card c3 = new Card(3);
        Card c4 = new Card(4);
        Card c5 = new Card(5);
        w.add(c1, 3);
        w.add(c2);
        w.add(c3, 5);
        w.add(c4);
        w.add(c5);
        Card.setCreditLimit(1000);
        Card ca = new Card(99);
        w.add(ca);
        w.print();
        w.purchase(c5, 100, 3);
        w.purchase(c2, 25, 7);
        w.purchase(c5, 200, 4);
        w.purchase(c5, 100, 3);
        w.purchase(c3, 100, 3);
        w.purchase(c1, 3000, 20);
        w.purchase(c1, 30, 20);
        w.purchase(c5, 100, 3);
        w.purchase(ca, 350, 2);
        Card.setCreditLimit(800);
        w.print();
    }

}
```

**Ejemplo:**

Línea 3 → vc1

| VagonCarga |
| --- |
| codigo = V1 |
| peso = 2500 |

```java
public class Wallet {
    private Card[] cards = new Card[6];

    public void add(Card card) {
        for (int i = 0; i < cards.length; i++) {
            if (cards[i] == null) {
                cards[i] = card;
                break;
            }
        }
    }

    public void add(Card card, int position) {
        cards[position] = card;
    }

    public void purchase(Card card, int price, int quantity) {
        for (int i = 0; i < cards.length; i++) {
            if (cards[i] == card) {
                card.pay(price * quantity);
                if (i > 0) {
                    Card c = cards[i];
                    for (int j = i; j > 0; j--) {
                        cards[j] = cards[j - 1];
                    }
                    cards[0] = c;
                }
                break;
            }
        }
    }

    public void print() {
        for (int i = 0; i < cards.length; i++) {
            System.out.println(cards[i]);
        }
    }
}
```

**Ingeniería en Tecnologías de Información**
**Ingeniería Civil en Computación e Informática**
**Ingeniería Civil Industrial**

```java
public class Card {
    private int number;
    private static int creditLimit;
    private int credit;

    public Card(int number) {
        this.number = number;
        this.credit = 0;
    }

    public int getNumber() {
        return number;
    }

    public void pay(int amount) {
        if (creditLimit - credit >= amount) {
            credit += amount;
            System.out.println("Compra Aprobada: " + amount);
        } else {
            System.out.println("Compra Rechazada");
        }

    }

    @Override
    public String toString() {
        if (credit > creditLimit) {
            return "Card [#" + number + ", utilizado=" + credit + ", Cupo sobrepasado]";
        }
        return "Card [#" + number + ", utilizado=" + credit + "]";
    }

    public static void setCreditLimit(int creditLimit) {
        Card.creditLimit = creditLimit;
    }

}
```

## Problema 2. Casino UCN (70 puntos)

En el casino de la **Universidad Católica del Norte**, han detectado un problema relacionado con la planificación del uso de los ingredientes. El casino no está seguro de si los ingredientes que reciben semanalmente serán suficientes para cubrir los menús de toda la semana, ya que los distribuidores no siempre envían la cantidad exacta requerida para los 5 días. El transporte de alimentos llega al inicio de cada semana con una cantidad determinada de ingredientes para preparar los menús diarios, pero no garantiza que alcancen para toda la semana.

### Estructura de los Archivos

Cuando se inicia el programa se debe cargar el archivo de inventario de la bodega, ya que al inicio de la semana, los ingredientes recibidos están almacenados en el archivo `bodega.txt` y este código se ejecutaría todos los lunes. Cada línea del archivo sigue la estructura:

| Archivo | Explicación |
|---|---|
| `001,Porotos,1,50`<br>`002,Lentejas,1,100`<br>`003,Tallarines 05,1,60`<br>.<br>. | **Código de producto**: Identificador único del ingrediente [String].<br>**Nombre de producto**: Nombre del ingrediente [String].<br>**Estado**: Indica si el producto está en buen estado (`1`) o vencido (`0`) [entero].<br>**Cantidad del producto**: Cantidad en kilogramos (kg) disponible en bodega [entero]. |

**Nota**: Si el mismo ingrediente aparece más de una vez en el archivo, con el mismo **código de producto**, las cantidades deben sumarse.

**Ingeniería en Tecnologías de Información**
**Ingeniería Civil en Computación e Informática**
**Ingeniería Civil Industrial**

Los menús que se pueden preparar para cada día están descritos en el archivo `recetas.txt`. La estructura del archivo es la siguiente:

| Archivo | Explicación |
|---|---|
| `Lentejas con Arroz,Fondo,4`<br>`Lentejas,10`<br>`Arroz,5`<br>`Zapallo,2`<br>`carne molida,2`<br>`Puré,Fondo,4`<br>`leche,10`<br>`papas,20`<br>`mantequilla,2`<br>`huevos,2`<br>`Ensalada Lechuga,Ensalada,1`<br>`Lechuga,100` | Nombre del menú, tipo Menú, Cantidad de ingredientes N<br>Ingrediente 1, Cantidad del ingrediente (en kg)<br>Ingrediente 2, Cantidad del ingrediente (en kg)<br>...<br>Ingrediente N, Cantidad del ingrediente (en kg)<br><br>**Nombre del menú**: Nombre del plato a preparar.<br>**Cantidad de ingredientes N**: Número de ingredientes necesarios para el menú.<br>**Ingredientes**: Lista de ingredientes con la cantidad requerida para preparar 100 platos. |

**Requisitos**

El casino necesita preparar un almuerzo diario para 100 personas de lunes a viernes. Cada menú consta de:

- Entrada: Ensalada, con una cantidad específica de ingredientes (incluida en el recetario).
- Plato de fondo: Preparado con una mezcla de ingredientes según lo indicado en el recetario.

Se requiere que diseñe e implemente una solución utilizando Programación Orientada a Objetos (POO) que permita:

Leer el inventario de productos de la bodega desde el archivo bodega.txt y organizar los datos para poder consultarlos.

Revisar el recetario en recetas.txt y determinar si es posible preparar los menús para los 5 días de la semana (de lunes a viernes), verificando si hay suficientes ingredientes para cubrir 100 platos por día.

Si no es posible preparar algún menú, debe indicar y explicar por qué (falta algún ingrediente).

Al final de la semana, mostrar si se pudo completar el menú de todos los días y qué ingredientes sobraron en la bodega.

Al final del proceso, se debe mostrar un reporte de los ingredientes restantes en la bodega (con su cantidad actualizada).

**Debe entregar:**

- Diagrama de clases (25%)
- Código Java (75%)

**Consideraciones:**

- Debe usar orientación a objetos.
- No se deben utilizar ciclos dentro de ciclos. Use funciones para hacerse la vida más fácil.
- El código fuente debe exportarlo en un solo archivo .zip y subirlo a Campus Virtual.
- Los 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.
- para saber que las leyó todas
- El recetario incluirá un máximo de 15 menús diferentes para la semana.
- La bodega no contendrá más de 20 productos en total
- Cada menú tendrá un número de ingredientes variable, pero no superior a 5 ingredientes.
- Si un ingrediente aparece repetido en el archivo con el mismo código de producto, deben sumar sus cantidades en base a dicho código.
- Los ingredientes en mal estado no deben ser utilizados en el programa.
- Debe usar referencias
- Si lee esto quiere decir que leyó las consideraciones, dibuje un tenedor al final de la página.

## Ejemplos de ejecución:

Revise el archivo `ejemplos.txt`