Objetivos:

- Datos, alcance y tiempo de vida.
- Sobre escritura de métodos.
- Ejecución de programas, Método Main().
- Numeros Grandes, LocalDateTime, Instant
- Diseño OO.
- Uso de patrones de diseño.

En esta práctica usaremos algunas clases definidas en la práctica 1. Es por ello que se recomienda finalizar la práctica 1 antes de comenzar con esta.

Ejercicio 1

El comedor universitario de la UNNOBA nos solicita la creación de un sistema de reservas de los platos del día. Es decir que los platos del día solo se manejan por reserva anticipada.

Del plato del día se conoce su nombre (por ejemplo: arroz con pollo), su precio (todos los platos tienen diferentes precios) y el día de la semana en el que estará disponible. (Por ejemplo: los lunes hay arroz con pollo, los martes milanesa con puré, etc., etc.).

De la reserva se conoce el cliente, la fecha y el plato reservado donde las reservas pueden ser de un solo plato del día o de más de un plato (Sin límite). Se puede dar el caso que un cliente reserve todos los platos de la semana, de la quincena, del mes.
En el caso de que las reservas sean de más de un plato se realiza un descuento del 5% sobre la suma de todos los platos.

Además, los clientes pueden ser alumnos, profesores o particulares. En el caso de que la reserva sea para un alumno, el precio del plato del día tendrá un descuento del 20%, en cambio, si la reserva la realiza un profesor el plato tiene un descuento del 10% y si la realiza un particular la misma no posee descuento.

Del comedor se conocen las reservas, la dirección y el teléfono del mismo.

Nota: el plato del día hace referencia a la opción de comida de ese día.

Se pide que:

1. Realice el diagrama de clases en UML
2. Desarrolle en JAVA todos las clases y métodos necesarios teniendo en cuenta que el sistema debe tener como mínimo la siguiente funcionalidad:

a) */* Agrega una reserva a las reservas del comedor*/*
```
public void agregarReserva(Cliente cliente, Plato plato)
```

b) */* Agrega varias reservas a las reservas del comedor*/*
```
public void agregarReserva(Cliente cliente, Vector<Plato> platos)
```

c) */* Retorna el valor del plato teniendo en cuenta los descuentos pertinentes */*
```
public float valorReserva(Reserva reserva)
```

d) */* Retorna la cantidad total de todos los platos reservados */*
```
public int totalPlatosReservados()
```

e) */* Retorna la suma total de todos los valores platos reservados */*
```
public float totalValorReservas()
```

f) */* Retorna la reserva que tenga más platos reservados */*
```
public Reserva reservaMasPlatos()
```

Ejercicio 2

La Conmebol está interesada en conocer datos estadísticos de los partidos de la Copa América de fútbol "`USA 2024`" y además el monto recaudado por los mismos. Para ellos posee un sistema de administración de partidos. El sistema contiene todos los partidos de la copa..

De cada partido se conoce a los equipos que se enfrentaron y el estadio donde se jugó el partido. Además de cada partido se conoce el valor de la entrada. Si bien los diferentes partidos tienen valores diferentes, en el mismo partido todas las entradas tienen el mismo valor.

De cada estadio se conoce su capacidad total de espectadores, la cual una vez creado el estadio la misma no varía.

Se pide que:

1. Modele en UML una solución propuesta para el sistema.
2. Implemente en JAVA todas las clases y métodos necesarios para resolver los siguientes mensajes que recibe el sistema.

a) `public void agregarPartido(Equipo local, Equipo visita, Estadio estadio, int valorEntrada);` *`//Agrega un partido a la colección de partidos que tiene el sistema.`*

b) `public Estadio estadioConMayorCapacidad();` *`//Retorna el estadio con mayor capacidad de espectadores`*

c) `public int cantidadTotalEspectadores();` *`//Retorna la suma de todos los espectadores de todos los partidos. Es decir, la suma de la capacidad de cada estadio de cada partido.`*

d) `public int montoRecaudado(Partido partido);` *`//Retorna la suma recaudada en el partido que se pasa como parámetro.`*

e) `public int montoTotalRecaudado();` *`//Retorna la suma total recaudada por todos los partidos de la copa`*

Nota: Se supone que TODOS los partidos son a estadio lleno. No es necesario modelar la venta de entradas. La capacidad del estadio es suficiente para hacer el cálculo del monto recaudado.

Ejercicio 3

Se quiere automatizar un horno de pan. Se quieren producir diferentes tipos de pan. Es decir que cada tipo de pan puede ser diferente según lo siguiente:

- **Pan blanco:** Está hecho de harina que contiene solo la parte central del grano, lo que representa el 75% del grano entero.
- **Pan integral:** Está hecho de grano de trigo entero, conserva todos sus componentes.
- **Pan negro:** Se confecciona con la harina de la que se ha quitado salvado y germen de trigo. Representa el 85% del grano de trigo entero.

El proceso de producción del PAN es siempre el mismo y se realizan los siguientes pasos:

a. Comprobar la temperatura.
b. Preparar agua con sal.
c. Preparar la levadura.
d. Añadir el agua y mezclar.
e. Añadir la harina y remover.

f. Amasar.
g. Dejar reposar.
h. Cortar la masa.
i. Llevar al horno.

a) Se pide el desarrollo de un modelo que permita representar el proceso de producción del pan:

1. Modele el sistema en UML.
2. Indique el patrón utilizado
    a. Clasificación
    b. Intención
    c. Motivación
    d. Estructura
3. Desarrolle el sistema en JAVA.

Ejercicio 4

Se requiere implementar un programa en Java para realizar operaciones matemáticas simples y compuestas utilizando el patrón `Composite`. El programa debe permitir la creación de expresiones matemáticas que combinan números y operadores aritméticos básicos (suma, resta, multiplicación y división).

Cada expresión puede estar compuesta por números simples o por otras expresiones. Se debe utilizar el patrón Composite para representar tanto las hojas (números simples) como los nodos (operaciones compuestas).

Para las operaciones aritméticas, se deben implementar las siguientes clases:

- `Operacion`: Una clase abstracta que define el comportamiento común para todas las operaciones.
- `Numero`: Una clase que representa un número simple.
- `Suma, Resta, Multiplicacion y Division`: Clases concretas que representan las operaciones aritméticas de suma, resta, multiplicación y división, respectivamente.

Cada clase de operación debe implementar el método `calcular()`, que devuelve el resultado de la operación correspondiente.

El programa principal debe permitir la creación de expresiones matemáticas complejas mediante la combinación de números simples y operadores aritméticos.

NOTA: cada operación (como por ejemplo suma) está compuesta por una operación izquierda y una operación derecha.

Ejemplo de Uso:

```
// Ejemplo de uso con operaciones múltiples
public class Main {
    public static void main(String[] args) {
        // Crear números
        Operacion num1 = new Numero(10);
        Operacion num2 = new Numero(5);
        Operacion num3 = new Numero(2);

        // Crear expresiones usando clases concretas para cada operación
        Operacion sumaResta = new Resta(new Suma(num1, num2), num3);
        Operacion multiDiv = new Division(new Multiplicacion(num1,
num2), num3);

        // Calcular
        System.out.println("Suma y Resta: " + sumaResta.calcular());
        System.out.println("Multiplicación y División: " +
multiDiv.calcular());
    }
}
```

En este ejemplo, hemos creado dos expresiones compuestas `sumaResta` y `multiDiv` que combinan varias operaciones. La primera realiza la suma de `num1` y `num2`, y luego resta `num3`. La segunda realiza la multiplicación de `num1` y `num2`, y luego divide el resultado por `num3`. Luego, simplemente llamamos al método `calcular()` en cada expresión para obtener el resultado correspondiente.

Ejercicio 5

Imagina que estás creando un simulador de tienda de muebles. Tu código está compuesto por clases que representan lo siguiente:

Una familia de productos relacionados, digamos: **Silla + Sofá + Mesilla.**

Algunas variantes de esta familia. Por ejemplo, los productos Silla + Sofá + Mesita están disponibles en estas variantes: **Moderna, Victoriana, ArtDecó**

Necesitamos una forma de crear objetos individuales de mobiliario para que se combinen con otros objetos de la misma familia. Los clientes se enfadan bastante cuando reciben muebles que no combinan.

Además, no queremos cambiar el código existente al añadir nuevos productos o familias de productos. Los comerciantes de muebles actualizan sus catálogos muy a menudo, y debemos evitar tener que cambiar el código principal cada vez que esto ocurra.

1. Modele el sistema en UML.
2. Indique el patrón utilizado
    a. Clasificación
    b. Intención
    c. Motivación
    d. Estructura
3. Desarrolle el sistema en JAVA

Nota: Usar clases Abstractas en donde corresponda. NO usar interfaces

Ejercicio 6

Busque ejemplos en teoría, en la bibliografía o en internet sobre al menos 2 patrones más:

a) Describa su uso
b) ¿Qué problema resuelve y cómo?
c) Ejemplificar en UML

Ejercicio 7

Dado el siguiente código que calcula el factorial:

```
public class CalcularFactorial {

	public static void main(String[] args) {
		System.out.println(factorial(50));
	}

	static double factorial(double n){
		if (n == 1.0)
			return 1.0;
```

```
            else
                  return n*factorial(n-1);
      }
}
```

a. Calcule el factorial de 50
b. Calcule el factorial de 200.
c. Que resultado da b) y ¿por qué?
d. Reescriba el código usando `BigInteger`

Ejercicio 8

Busque ejemplos en teoría, en la bibliografía o en internet sobre `LocalDateTime` (introducido en *Java8*)

a) Describa su uso
b) ¿Qué problema resuelve y cómo?
c) Brinde Ejemplos de Uso
d) Ejemplifica en JAVA.

Ejercicio 9

Busque ejemplos en teoría, en la bibliografía o en internet sobre `Instant` (introducido en *Java8*)

e) Describa su uso
f) ¿Qué problema resuelve y cómo?
g) Brinde Ejemplos de Uso
h) Ejemplifica en JAVA.