**Práctica # 11 Modularidad**

Nombre: GIAN MOLINA RAIGOZA Matrícula:1636155 Calificación:\_\_\_\_\_\_ Ponderación: 7% Día: \_\_\_\_\_\_\_\_\_\_\_\_ Hora: \_\_\_\_\_\_:\_\_\_\_\_\_

**ObjetivosGenerales:**

* Reforzar el conocimiento de modularidad.
* Observar la importancia y las ventajas de organizar el código.

**Actividad #1 – Modularidad y abstracción**

**Ponderación:** 80%

Se elaborará un sistema para una Tienda. Las funcionalidades de este sistema será dar de alta Productos y realizar recibos.

**Procedimiento**

Nota: Tomar en cuenta los modificadores de los atributos y los métodos.

* + Public
* - Private
* # Protected

1. Cree un programa llamado Tienda.java aquí va a radicar nuestro método main.
2. Crear un paquete llamado **Producto**.
3. Crear una clase llamada Producto en el **paquete Producto** con las siguientes características.

|  |
| --- |
| **Producto** |
| # String nombre  # double precio  # int cantidad |
| + void imprimir() |

1. El método imprimir() imprime todos los atributos de la clase.
2. Crear una clase llamada ProductoPerecedero en el **paquete Producto** que herede de la clase Producto con las siguientes características.

|  |
| --- |
| **ProductoPerecedero** |
| # String fechaCaducidad  # int caducado |
| + void imprimir() |

1. El método imprimir() sobre escribe al método de la clase padre e imprime los datos de la clase padre y la clase hija.
2. Crear una clase llamada ProductoElectronico en el **paquete Producto** que herede de la clase Producto con las siguientes características.

|  |
| --- |
| **ProductoElectronico** |
| # String marca  # int watts |
| + void imprimir() |

1. El método imprimir() sobre escribe al método de la clase padre e imprime los datos de la clase padre y la clase hija.
2. Crear un paquete llamado **Recibo**.
3. Crear una clase abstracta llamada Recibo en el **paquete Recibo** con las siguientes características.

|  |
| --- |
| **Recibo** |
| - Producto productos[10]  - int cantidadProductos = 0  - final int maxProductos = 10 |
| + void imprimir()  + void agregarProducto(Producto p) |

1. El método agregarProducto añadirá un objeto Producto al arreglo de productos, este objeto deberá de haber sido leído previamente en el main y lo recibirá como parámetro en el método y lo agregará a la siguiente posición disponible del arreglo productos del objeto Recibo donde se utilizará el atributo cantidadProductos para manejar cuantos productos hay en el arreglo y verificar que no sobrepase el limite de maxProductos.
2. El método imprimir mostrara en consola la información de todos los productos y el total de todos los productos.
3. En la clase en donde se tiene el método main (Tienda.java), el objetivo será declarar un objeto de la clase Recibo.
4. Posteriormente realice el siguiente flujo de instrucciones que dictará su menú:
5. Agregar Producto
6. Imprimir Recibo
7. Salir

Descripción del menú

1. Se agregará un Producto al objeto Recibo a través del método agregarProducto, el programa preguntará si se trata de un ProductoPerecedero o un ProductoElectronico y leerá los datos correspondientes para asignárselos a un objeto Producto para agregárselo al objeto Recibo.
2. Se imprimirá toda la información que haya en el objeto Recibo.
3. El programa finalizará.

Nota: Si se llenó nuestro arreglo de objetos, desplegarle el mensaje de “Se ha llegado al Límite de Productos” al usuario.

**Actividad #2 – Importancia de modularidad**

**Ponderación:** 20%

Describa con sus palabras la importancia de mantener un código en donde se implemente la modularidad.

La modularidad nos sirve para dividir el programa en diferentes paquetes es decir pequeños sub programas para llevar acabo un orden del programa que estamos realizando ya que estos subprogramas colaboran entre si y así poder conectarlos.

Es muy bueno en la programación orientada a objetos para simplificar programas por sus distintas clases por sus tipos de objetos por ejemplo llevar a cabo una clase de carros con otra de motos, también es importante debido a que nos sirve para poder reutilizar parte de ese código para futuros programas o proyectos, además de que es fácil de entender y fácil de sustituir, agregar o poder modificar con facilidad dicho código.

-Gian Molina Raigoza