README P1 MyP

1 Información del equipo:

Nombre del equipo: Quantum Gang.
Integrantes:

- López Cortes Adamari Gianina 320268458
- Gonzalez Gutierrez Antonio Tonatiuh 321195476
- Zapata Amezcua Santiago 321251796

2 Descripcion y precio de productos

- Procesador (CPU)
  - Intel
    - Core i3-13100: $1,750.00
      procesador de escritorio de la 13ª generación con 4 núcleos.
    - Core i5-13600K: $2,600.00
      procesador de sobremesa de 13ª generación con 14 núcleos
    - Core i7-13700K: $6,200.00
      procesador de la 13a generacion que cuenta con 16 núcleos (8 núcleos de rendimiento y 8 núcleos eficientes).
    - Core i9-13900K: $5,100.00
      Cantidad de núcleos. 24.
  - AMD (Estos fueron agregados por nosotros)
    - Ryzen 5 5600G: $2,449.00
      Procesador con 6 nucleos
    - Ryzen 5 7600X: $4,789.00
      Procesador con 6 nucleos
    - Ryzen 7 7700X: $7,043.00
      Procesador con 8 nucleos
    - Ryzen 9 7950X3D: $15,059.99
      Procesador con 16 nucleos.
- RAM

  - Adata
    * 8 GB: $350.00
      ADATA Memoria RAM DDR4 PC4-24000
    * 16 GB: $550.00
      ADATA, Memoria RAM Premier DDR4 de 16GB
    * 32 GB: $980.00
      Memoria RAM Adata Premier DDR4
  - Kingston
    * 8 GB: $400.00
      Kingston Fury Beast DDR4
    * 16 GB: $680.00
      Kingston Fury Beast DDR4 CL16
    * 32 GB: $1,240.00
      Kingston Fury Beast Black DDR4 CL16

- Motherboard
  - ASUS
    * ROG Maximus Z790 Hero: $12,110.00
      Tarjeta Madre ASUS ATX, 192GB DDR5
    * TUF Gaming B760-Plus WIFI D4 $4,650.00
      Tarjeta Madre ASUS TUF GAMING B760-PLUS WIFI
  - MSI
    * MEG Godlike: $13,600.00
      Placa Madre MSI MEG Z790 GODLIKE LGA 1700
    * MAG B760 Tomahawk WIFI DDR4: $4,710.00
      Tarjeta Madre MSI ATX, 128GB DDR4

- HDD (Discos duros mecanicos)
  - Western Digital Blue
    * 500 GB: $1,100.00
      WD Blue 3D NAND SSD 500GB
    * 1 TB: $1,300.00
      Disco Duro Interno 1tb Western Digital Blue 7200rpm
  - Seagate Barracuda
    * 1 TB: $1,150.00
      Seagate Disco Duro Barracuda 1TB
    * 2 TB: $1,600.00
      Disco Duro Interno Seagate Barracuda 2TB plata

- SSD (Discos de Estado Sólido)
  - Kingston

        * 500 GB: $820.00
          Kingston SSD NV3 500GB
        * 1 TB: $1,190.00
          Kingston SSD NV3 1000GB
        * 2 TB: $2,160.00
          Kingston SSD 2000GB USB 3 2 Gen
        * 4 TB: $5,540.00
          SSD Externo Kingston 4TB

- Fuente de alimentación
  - EVGA
    * 800 W: $1,920.00
      Fuente de Poder EVGA 800 GE 80 PLUS Gold 800W
    * 1000 W: $4,010.00
      Fuente Poder EVGA 1000W
    * 1500 W: $7,970.00
      Fuente Poder NZXT 1500W 80 PLUS Plat.
  - Corsair
    * 800 W: $2,800.00
      Fuente de Poder Corsair Gaming Series GS800 de 800W
    * 1000 W: $3,120.00
      CORSAIR Fuente 1000W
    * 1500 W: $8,600.00
      Corsair 1500W
  - XPG
    * 800 W: $2,460.00
      FUENTE PODER XPG 800W
    * 1000 W: $3,220.00
      Fuente Poder XPG 1000W
    * 1500 W: $3,840.00
      Fuente Poder XPG 1500W

- Tarjeta gráfica (GPU)
  - NVIDIA
    * GTX 1660: $4000.00
      Tarjeta de video Nvidia GDDR6
    * RTX 3060: $5,470.00
      Tarjeta de Video NVIDIA GDDR6
    * RTX 4070: $13,400.00
      Tarjeta de video Nvidia GDDR6
    * RTX 4080: $43,630.00
      Tarjeta gráfica NVIDIA GDDR6X

    * RTX 4090: $48,720.00
      Tarjeta de VIDEO NVIDIA GDDR6X

- Gabinetes
  - NZXT H6 Flow: $ 2,180.00
    GABINETE NZXT H6 Flow
  - Yeyian Lancer: $1,320.00
    GABINETE YEYIAN LANCER

Justificacion para el uso de cada patron:

- **Adapter:** Se utiliza para resolver incompatibilidades entre componentes de diferentes marcas (por ejemplo, CPU AMD con GPU Nvidia o motherboard Intel). Gracias al Adapter, podemos envolver un componente no compatible en un adaptador que lo haga funcionar sin modificar su código original, reduciendo la rigidez y evitando duplicar clases.
- **Builder:** Permite ensamblar instancias de `Computadora` paso a paso, ya sea personalizada o pre-armada, sin exponer la lógica de construcción al cliente. Así se mantiene el código limpio, se evita la fragilidad al cambiar el orden de ensamblaje y se mejora la legibilidad.
- **Strategy (Regla de compatibilidad):** Cada regla de compatibilidad implementa la misma interfaz `ReglaCompatibilidad`, lo que facilita añadir nuevas reglas sin tocar el resto del sistema. El `CompatibilidadManager` aplica dinámicamente la estrategia adecuada según el conflicto detectado, promoviendo la extensibilidad.
- **Decorator:** Permite añadir software (Windows, Office, Photoshop, AutoCAD, WSL) de forma modular a una `Computadora` existente sin alterar su estructura interna. Se evita la proliferación de subclases y se previene la viscosidad al agregar nuevas opciones de software.
- **Abstract Factory:** Separa la creación de familias de componentes (Intel+Nvidia vs AMD) de su uso en la aplicación. Las fábricas `IntelNvidiaFactory` y `AmdFactory` proporcionan catálogos y productos consistentes, facilitando la compatibilidad y evitando la inmovilidad al introducir nuevos proveedores.
- **MVC:** Organiza la aplicación en tres capas:
  - `Model`: lógica de negocio y construcción de computadoras.
  - `View`: interacción con el usuario (menús, entradas, salidas).
  - `Controller`: coordina el flujo de la aplicación (selección de sucursal, compra, compatibilidad, generación de ticket).

  Esta separación mejora la mantenibilidad y reduce la rigidez, ya que cambios en la interfaz de usuario no afectan la lógica de negocio.

Uso de Ptrones en nuestro Proyecto:

3 Adapter

Usamos adapter para cuando el usuario eliga dos componentes que no son compatible entre si, entonces se adapte a una solucion en donde si funcione con los componentes que eligio previamente el usuario.

4 Builder

Usamos builder para construir las computadoras por componentes.
En la interfaz **ComputadoraBuilder** por cada metodo que tiene esta claase es que se le va agregando un componente a la computadora que vamos a construir; al final retorna el objeto de computadora con todos los componentes elegidos por el usuario si es personalizable o si es prearmada los ya establecidos.
En la clase **ComputadoraDirector** hace la computadora builder pero con cierto orden logico. Agrega la CPU, agrega módulos de RAM (pueden ser múltiples), agrega la GPU, agrega discos de almacenamiento, agrega la fuente de poder, agrega la placa basey agrega el gabinete; y retorna la computadora ya construida.
La clase **ComputadoraPersonalizadaBuilder** construye la computadora componente por componente con los metodos de la interfaz de ComputadoraBuilder; hay componentes que tienen set y agregar la diferencia de esto es que el set se usa cuando solo puede haber un componente y el agregar cuando puede haber mas de uno como en el caso de la RAM y del disco.
La clase de **ComputadoraPrearmadaBuilder** recibe la fabrica de componenetes y el nombre del modelo (gamer, basica o de estudio); la configuraciones añade componentes usando la fabrica, para asi al final devolver la computadora; lanza una excepcion si el usuario agrega el nombre de un modelo que no existe.

5 Strategy

LDefinimos varias reglas que implementan una interfaz común llamada **ReglaCompatibilidad**, la cual permite detectar en tiempo de ejecución si existe un conflicto de compatibilidad, determinar si dicho conflicto es adaptable y, en caso afirmativo, aplicar el adaptador correspondiente para resolverlo. Todas esas reglas quedan agrupadas en **CompatibilidadManager** (por ejemplo, AMD vs Nvidia, AMD vs Motherboard, etc.). Durante el ensamblaje de una computadora, el sistema invoca **manager.verificar(pc)** para obtener la lista de conflictos; si se detecta alguno y es adaptable, el usuario puede optar por llamar a**manager.adaptar(pc)** para corregirlo. De este modo centralizamos la lógica de detección y adaptación, pero dejamos en manos del usuario la decisión final de aplicar los adaptadores.

6 Decorator

Usamos decorator para que se pueda ir agregando el software que requiera el usuario. La clase **OfficeDecorator** añade el software Microsoft Office 365 a una computadora, siguiendo el patrón Decorator. Su función es agregar esta funcionalidad extra de forma modular y reutilizable. al igual que las clases de **PhotoshopDecorator, WindowsDecorator y WSLDecorator**.
La clase abstracta de **SoftwareDecorator** sirve como base para decoradores que añaden software a una computadora, siguiendo el patrón Decorator y evita la duplicación de software.

7 Abstract Factory

Usamos este patron para construir los componentes de las computadoras por NVIDIA e INTEL La interfaz **Almacenamiento** define el contrato que deben cumplir los componentes de almacenamiento (discos duros, SSDs, etc.) en una computadora.
La clase **AmdFactory** es una fábrica concreta que implementa la interfaz ComponenteFactory, diseñada para crear componentes de hardware compatibles con sistemas AMD.
La clase abstracta **Componente** la base para todos los componentes de hardware: CPU, GPU, RAM, etc., heredan de esta clase. Una vez creado un componente, sus datos no pueden modificarse y centraliza atributos comunes como modelo, precio, marca y tipo.
La interfaz **ComponenteFactory** define una fabrica abstracta para crear componentes de hardware y gestionar su compatibilidad basica.
La interfaz **CPU** es la base para todas las CPUs en el sistema, establece un contrato que deben cumplir las implementaciones concretas.
La clase **CPUAmd** representa procesadores de la marca AMD con funcionalidades específicas.
La clase **CPUIntel** representa procesadores de la marca INtel con ciertas especificaciones.
La interfaz **FuneteDePoder** establece un contrato común para descripción técnica detallada, información de precios,especificación de potencia.
La interfaz **gabinete** es la base para todos los tipos de gabinetes. Establece contrato común para la descripción técnica detallada y la nformación de precios.
La interfaz **GPU** es la base para todas las unidades gráficas. Establece contrato común para la descripción técnica detallada y la información de precios.
La clase **GPUNvidia** representa los GPU de la marca Nvidia con sus funcionalidades especificas, asi como su descripción y precios. La clase **IntelNvidiaFactory** es la fabrica que hace todos los componentes de dicha marca como el CPU, GPU, RAM, etc.
La clase **Marca** son las marcas usadas por los componentes.
La interfaz **Motherboard** es la base para crear todas las motherboard, establece un contrato conlas especificaciones que debe de tener este componente.

Al igual que en la interfaz **RAM**.
La clase **MotherboardIntel** son las placas madres de la marca Intel con las especificaciones y funcionalidades.
La clase **ProductoAlmacenamiento** crea el producto con las especificaciones; al igual que en las clases **ProductoFuenteDePoder**, **ProductoGabinete** y **ProoductoRAM**.
En la clase **TipoComponente** son las categorias que son validas o que existen dentro del programa.

8 MVC

En la clase **Controller** hace el proceso de construcción de computadoras (personalizadas o prearmadas), la gestión de compatibilidad entre componentes, la adición de software adicional y la generación de tickets finales.
En la clase **Model** accede al catalogo, hace la construccion de las computadoras prearmadas y personalizadas, verifica la compatibilidad, aplica adaptadores y genera el ticket. En la clase **View** genera el menu principal y agrega los sotware que quiera el cliente, selecciona el catalogo y cuantas unidades quiera de los componentes.
En la clase **Computadora** crea las computadoras y calcula el precio de esta misma.
Y la clase **Ticket** imprime el ticket con los datos obtenidos.

Acerca del diagrama de clases....

consideramos el uso de colores para distringuir los patrones acontinuacion damos una breve descripcion del color y con que patron esta asociado azul-AbstracFactory verde-mvc naranja-adapter morado-Builder amarillo-Strategy rosa-Decorator

9 Instrucciones de Compilaccion y Ejecucion

La implementacion fue desarrollada y probada utilizando JDK 22. Se compilo y ejecuto en un entorno Linux, utilizando los comandos estandar de java: **Compilación:** Desde la carpeta raiz del proyecto, ejecutar:
copiar:
mkdir -p out
javac -encoding UTF-8 -d out$(find src -name"*.java")
java -cp out src.Main
Asegurese de que la clase Main se encuentre en el paquete correcto o ajustar el comando según la estructura de paquetes, tambien que el archivo de Ticket.txt