POLITÉCNICO GRANCOLOMBIANO
INSTITUCIÓN UNIVERSITARIA

FACULTAD DE INGENIERÍA, DISEÑO E INNOVACIÓN

CONCEPTOS FUNDAMENTALES DE PROGRAMACIÓN

Entrega 3 – Semana 7

Integrantes

Subgrupo 4

YULY ANDREA BEDOYA BETANCUR
EVER LEONARDO GOMEZ CLAVIJO
JORGE ENRIQUE HERNANDEZ GUZMAN
DIEGO ALEJANDRO LEIVA MORALES
JUAN DAVID PINEDA BAUTISTA

abril 2024

Introducción

En este documento encontraras el desarrollo de un programa en el lenguaje de java que va a procesar información de ventas de muchos vendedores, el cual por medio de archivo planos va a mostrar los productos vendidos y quienes fueron los vendedores que más recaudaron. Al finalizar el proyecto podras con el analizar las ventas del negocio y obtener información al instante que te servirá para tomar decisiones, identificando asi mejoras para aumentar las ganancias y ventas.

Durante el desarrollo de este proyecto hemos adquirido valiosas destrezas y conocimientos en la programación con el lenguaje java, enfrentamos desafíos de conocimientos ya que no sabíamos bien del tema de programar.

INSTRUCCIONES PARA REALIZAR LA ENTREGA

Para el proyecto del Módulo, se espera que cada entrega se comparta mediante un hipervínculo a un repositorio, un programa que tome como entrada una serie de archivos con información de vendedores. Habrá un archivo de texto plano por cada vendedor, el cual tendrá el siguiente formato (de una venta por línea):
TipoDocumentoVendedor;NúmeroDocumentoVendedor
IDProducto1;CantidadProducto1Vendido; IDProducto2;CantidadProducto2Vendido;
IDProducto3;CantidadProducto3Vendido; La cantidad de vendedores es desconocida, pero como entrada el programa tendrá varios (posiblemente, muchos) archivos planos en una carpeta, cada uno con la información de ventas de un vendedor. Todos los archivos de vendedores deben estar en la misma carpeta de proyecto que el programa a entregar, con el fin de facilitar la exploración de estos archivos desde el programa. Adicionalmente, el programa tendrá como entrada un archivo con la información de los vendedores. El archivo de texto plano tendrá el formato que se describe a continuación, con un vendedor por línea.
Formato archivo de información vendedores:
TipoDocumento;NúmeroDocumento;NombresVendedor1;ApellidosVendedor1
TipoDocumento;NúmeroDocumento;NombresVendedor2;ApellidosVendedor2
TipoDocumento;NúmeroDocumento;NombresVendedor3;ApellidosVendedor3 El programa también tendrá como entrada un archivo con la información de todos los productos disponibles. Cada producto debe ir con el ID y el nombre. POLITÉCNICO GRANCOLOMBIANO 2 POLITÉCNICO POLITÉCNICO GRANCOLOMBIANO GRANCOLOMBIANO 3 Formato archivo de información productos:
IDProducto1;NombreProducto1;PrecioPorUnidadProducto1
IDProducto2;NombreProducto2;PrecioPorUnidadProducto2
IDProducto3;NombreProducto3;PrecioPorUnidadProducto3 La tarea mínima del proyecto consiste en diseñar e implementar un programa en Java bien sustentado y con buenas costumbres de programación que, tomando como entrada los archivos descritos, haga las siguientes tareas: 1. Muy buena documentación del código según los estándares de documentación en Java. 2. Excelentes prácticas de programación, especialmente en el espaciado y nombramiento de variables. 3. El programa debe crear un archivo con la información de todos los vendedores, de a uno por línea. Al frente del nombre de cada vendedor, separado por punto y coma, debe estar la cantidad de dinero que recaudó según los archivos. El archivo debe estar ordenado por cantidad de dinero, de mayor a menor, de a un vendedor por línea. Es básicamente un archivo de reporte de ventas de los vendedores, del mejor al peor; un archivo CSV. 4. El programa debe crear un archivo con la información de los productos vendidos por cantidad, ordenados en forma descendente. Deben ir el nombre y el precio, separados por punto y coma, y de a un producto por línea. Es básicamente un

archivo plano CSV. 5. Para propósitos de prueba, se deben crear métodos de generación de archivos de prueba para el programa en cuestión. Entre estos métodos deben estar al menos: a. createSalesMenFile(int randomSalesCount, String name, long id): dada una cantidad, un nombre y un id, crea un archivo pseudoaleatorio de ventas de un vendedor con el nombre y el id dados. b. createProductsFile(int productsCount): crea un archivo con información pseudoaleatoria de productos, con los datos de productsCount productos. c. createSalesManInfoFile(int salesmanCount): crea un archivo con información de salesmanCount vendedores; el número de estos según lo indique el argumento entero. La información debe ser generada de manera pseudoaleatoria y ser coherente, es decir, los nombres y apellidos pueden ser extraídos de listas de nombres reales de personas. POLITÉCNICO GRANCOLOMBIANO 3 POLITÉCNICO POLITÉCNICO GRANCOLOMBIANO GRANCOLOMBIANO 4 Extra: el programa puede tener al menos uno de los siguientes elementos, lo cual influirá muy positivamente en la nota asignada: a. La posibilidad de procesar más de un archivo por vendedor. b. La posibilidad de trabajar con archivos serializados. c. La posibilidad de detectar archivos con formato erróneo o con información incoherente, como un id de producto que no exista, o precios o cantidades negativas. Dado que hay 5 elementos en la lista de instrucciones, cada uno tiene la misma prioridad en la calificación. Se invita a todos los estudiantes a compartir aproximaciones a soluciones y a funciones (métodos) cortas en los foros. La participación mínima es aquella en la que hay un hipervínculo asociado a un repositorio, donde esté el proyecto de Eclipse, que tenga todas las clases y archivos necesarios para ejecutarse. En este proyecto deben haber dos (y solo dos) clases con método main. La primera clase debe llamarse GenerateInfoFiles y al ejecutarse debe generar los archivos planos pseudoaleatorios que servirán como entrada para la ejecución de la segunda clase con método main. El programa debe mostrar un mensaje de finalización exitosa o un mensaje de error, en caso de que algo salga mal. La segunda clase debe llamarse main y al ejecutarse debe realizar las tareas de creación de los archivos solicitados de reportes en los puntos 3 y 4 de la lista de requisitos señalados anteriormente. El programa debe mostrar un mensaje de finalización exitosa o un mensaje de error, en caso de que algo salga mal. Vale la pena anotar que ninguno de los dos programas a ejecutar puede solicitar información al usuario. En todos los programas es espera que el estudiante o grupo de estudiantes compartan un hipervínculo a un repositorio de código como github o bitbucket en donde tengan su proyecto almacenado. El IDE a usar es Eclipse para Java Developers. La versión de Java a usar es la 8.

ENTREGA 3 SEMANA 7

1 lo aprendido durante el desarrollo del proyecto.
2. Posibles aplicaciones en su vida profesional de las destrezas y conocimientos adquiridos y practicados durante el desarrollo del proyecto.
3. Las dificultades presentadas durante el desarrollo del proyecto

Este es el proyecto realizado por nosotros durante las entregas 1 y 2

Código de programación

```
//Import of all classes and utilities from the java. package

package productoVentas;

import java.util.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

@SuppressWarnings("unused")

public class GenerateInfoFiles { // Class that generates the information of the
requested files

	// Creation of the Product Class, whic its attributes.
	static class Product {
		String idProduct;
		String nameProduct;
		double priceByUnit;

		// Constructor method of the product class.
		public Product(String idProduct, String nameProduct, double priceByUnit)
{
			this.idProduct = idProduct;
			this.nameProduct = nameProduct;
			this.priceByUnit = priceByUnit;
		}
	}

	// Creation of the Seller class, whic its attributes.
	static class Seller {
		String documentType;
		String documentNumber;
		String namesSeller;
		String surnamesSeller;

		// Constructor method of the Seller class.
		public Seller(String documentType, String documentNumber, String
namesSeller, String surnamesSeller) {
			this.documentType = documentType;
			this.documentNumber = documentNumber;
			this.namesSeller = namesSeller;
			this.surnamesSeller = surnamesSeller;
		}

	}

	// Creation of the Product Sales, whic its attributes.
	static class Sales {
		Seller seller;
		Product product;
		int amount;
		double totalSalesProduct;
```

```
            // Constructor method of the Sales class.
            public Sales(Seller seller, Product product, int amount, double
totalSalesProduct) {
                  this.seller = seller;
                  this.product = product;
                  this.amount = amount;
                  this.totalSalesProduct = totalSalesProduct;
            }

            // Method of obtaining total sales
            double getTotalsSales() {
                  return product.priceByUnit * amount;
            }
      }

      // Creation of the main method
      public static void main(String[] args) {

            // Generating a seller list array
            List<Seller> seller = new ArrayList<>(Arrays.asList(new Seller("CC",
"1001", "Juan", "Perez"),
                        new Seller("CC", "1002", "Ana", "Gomez"), new Seller("CC",
"1003", "Carlos", "Martinez"),
                        new Seller("CC", "1004", "Luisa", "Rodriguez"), new
Seller("CC", "1005", "Pedro", "Lopez"),
                        new Seller("CC", "1006", "Andrea", "Bedoya"), new
Seller("CC", "1007", "Santiago", "Vera"),
                        new Seller("CC", "1008", "Sebastian", "Ruiz"), new
Seller("CC", "1009", "Lina", "Velez")));

            // Generating a product list array
            List<Product> product = new ArrayList<>(
                        Arrays.asList(new Product("P001", "Refrigerador", 1800000),
new Product("P002", "Lavadora", 650000),
                        new Product("P003", "Microondas", 350000), new
Product("P004", "Licuadora", 200000),
                        new Product("P005", "Horno electrico", 250000), new
Product("P006", "Tostadora", 80000),
                        new Product("P007", "Cafetera", 60000), new Product("P008",
"Aspiradora", 120000),
                        new Product("P009", "Plancha", 70000), new Product("P010",
"Ventilador", 65000),
                        new Product("P012", "celular", 70000), new Product("P011",
"Base Cama", 65000),
                        new Product("P013", "Portatil", 170000), new Product("P014",
"Diadema inalambrica", 500000)));


            // Generating a Sales list array
            List<Sales> sales = new ArrayList<>();
            Random random = new Random();
            for (Product product2 : product) {
                  Seller seller2 = seller.get(random.nextInt(seller.size()));
                  int amount = 1 + random.nextInt(10);
                  double totalSalesProduct = product2.priceByUnit * amount;
                  sales.add(new Sales(seller2, product2, amount,
totalSalesProduct));
            }
```

```
            // Called the method that generates the product file
            generateProductFile(product);

            // Called the method that generates the sellers file
            generateSellerFile(seller);

            // Called the method that generates the Sales file
            generateSalesFile(sales);

            // Called the method that generates the SalesMenReport file
            generateSalesMenReportFile (sales);

            // Called the method that generates the ProdctSalesReport file
            generateProductSalesReportFile(sales);

            // Called the method that generates the SalesBySellerFiles file
            generateSalesBySellerFiles(sales);

      }

      // Method that generates the product flat file
      private static void generateProductFile(List<Product> product) {
            try (BufferedWriter writer = new BufferedWriter(new
FileWriter("products.txt"))) {
                  for (Product products : product) {
                        writer.write(products.idProduct + "," + products.nameProduct
+ "," + products.priceByUnit + "\n");
                  }
                  // Completion message with the successful procedure
                  System.out.println("Products file generated successfully.");

                  // Completion message with the procedure with error
            } catch (IOException e) {
                  System.err.println("Error writing to products file: " +
e.getMessage());
            }
      }

      // Method that generates the seller flat file
      private static void generateSellerFile(List<Seller> seller) {
            try (BufferedWriter writer = new BufferedWriter(new
FileWriter("Sellers.txt"))) {
                  for (Seller sellers : seller) {
                        writer.write(sellers.documentType + "," +
sellers.documentNumber + "," + sellers.namesSeller + ","
                                    + sellers.surnamesSeller + "\n");
                  }

                  // Completion message with the successful procedure
                  System.out.println("Sellers file generated successfully.");

            }
            // Completion message with the procedure with error
            catch (IOException e) {
                  System.err.println("Error writing to Sellers file: " +
e.getMessage());
            }
      }
```

```
      // Method that generates the sales flat file
      public static void generateSalesFile(List<Sales> sales) {
            try (BufferedWriter writer = new BufferedWriter(new
FileWriter("Sales.txt"))) {
                  for (Sales sales2 : sales) {

                        writer.write(sales2.product.idProduct + "," +
sales2.product.nameProduct + ","
                                    + sales2.product.priceByUnit + "," +
sales2.amount + "," + "," + sales2.seller.documentType
                                    + ";" + sales2.seller.namesSeller + ";" +
sales2.seller.surnamesSeller + ","
                                    + sales2.seller.documentNumber + "," +
sales2.totalSalesProduct + "\n");
                  }
                  // Completion message with the successful procedure
                  System.out.println("Sales file generated successfully.");

                  // Completion message with the procedure with error
            } catch (IOException e) {
                  System.err.println("Error writing to Sales file: " +
e.getMessage());

            }

      }

      // Method to create the CSV file with the information of products sold by
quantity, ordered in descending order
      private static void generateProductSalesReportFile(List<Sales> sales) {
          Map<String, Integer> productQuantities = new HashMap<>();

          // Calculate the quantity and total per product sold
          for (Sales sale : sales) {
              String productName = sale.product.nameProduct;
              int quantity = productQuantities.getOrDefault(productName, 0);
              productQuantities.put(productName, quantity + sale.amount);
          }

          List<Map.Entry<String, Integer>> productList = new
ArrayList<>(productQuantities.entrySet());
          productList.sort((e1, e2) -> e2.getValue().compareTo(e1.getValue()));

          // Generates the CSV file with the information of the products sold,
separated by columns
          try (BufferedWriter writer = new BufferedWriter(new
FileWriter("ProductSalesReport.csv"))) {
              // Write CSV file headers
              writer.write("Nombre Producto;Precio\n");

              // Write the information of the sold products to the CSV file
              for (Map.Entry<String, Integer> entry : productList) {
                  // Find the corresponding product in sales
                  String productName = entry.getKey();
                  Sales productSale = sales.stream()
                          .filter(sale ->
sale.product.nameProduct.equals(productName))
                          .findFirst()
```

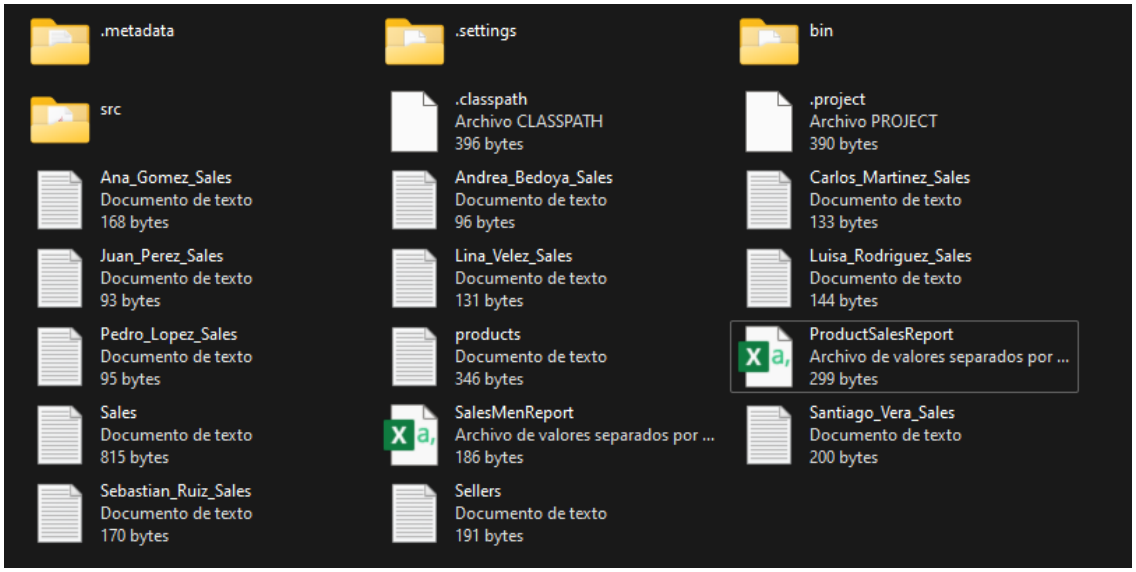
```
                        .orElse(null);

                // brings the price of the product in the CSV file
                if (productSale != null) {
                    writer.write(productName + ";" +
productSale.product.priceByUnit + "\n");
                }
            }

            System.out.println("Product sales report file generated
successfully.");
        } catch (IOException e) {
            System.err.println("Error writing to Product sales report file: " +
e.getMessage());
        }
    }

    // Method that adds the sales made per seller and outputs them in a CSV file

    private static void generateSalesMenReportFile(List<Sales> sales) {
        //Create a map to store total sales by seller
        Map<String, Double> salesBySeller = new HashMap<>();

        // Calculate total sales per seller
        for (Sales sale : sales) {
            String sellerName = sale.seller.namesSeller + " " +
sale.seller.surnamesSeller;
            double totalSales = salesBySeller.getOrDefault(sellerName, 0.0);
            totalSales += sale.totalSalesProduct;
            salesBySeller.put(sellerName, totalSales);
        }

        try (BufferedWriter writer = new BufferedWriter(new
FileWriter("SalesMenReport.csv"))) {
            writer.write("Nombre Vendedor;Total Vendido\n");

            // Write sales totals by seller to CSV file
            for (Map.Entry<String, Double> entry : salesBySeller.entrySet()) {
                writer.write(entry.getKey() + ";" + entry.getValue() + "\n");
            }
            System.out.println("Salesmen report file generated successfully.");
        } catch (IOException e) {
            System.err.println("Error writing to Salesmen report file: " +
e.getMessage());
        }
    }


    // Method to generate sales files by seller
    private static void generateSalesBySellerFiles(List<Sales> sales) {
        Map<String, List<Sales>> salesBySeller = new HashMap<>();

        // Group sales by seller
        for (Sales sale : sales) {
            String sellerName = sale.seller.namesSeller + "_" +
sale.seller.surnamesSeller;
            if (!salesBySeller.containsKey(sellerName)) {
                salesBySeller.put(sellerName, new ArrayList<>());
            }
```

```
                salesBySeller.get(sellerName).add(sale);
            }

            // Generate files per seller
            for (Map.Entry<String, List<Sales>> entry : salesBySeller.entrySet()) {
                String sellerName = entry.getKey();
                List<Sales> sellerSales = entry.getValue();
                generateSalesFileBySeller(sellerName, sellerSales);
            }
        }

        // Method to generate sales file by seller
        private static void generateSalesFileBySeller(String sellerName, List<Sales>
sales) {
            String fileName = sellerName + "_Sales.txt";

            try (BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName))) {
                writer.write("ID Producto,Nombre Producto,Precio por
unidad,Cantidad,Total\n");

                //Write seller sales to file
                for (Sales sale : sales) {
                    writer.write(sale.product.idProduct + "," +
sale.product.nameProduct + "," + sale.product.priceByUnit +
                            "," + sale.amount + "," + sale.totalSalesProduct + "\n");
                }

                System.out.println("Sales file for " + sellerName + " generated
successfully.");
            } catch (IOException e) {
                System.err.println("Error writing to sales file for " + sellerName +
": " + e.getMessage());
            }
        }


}
```

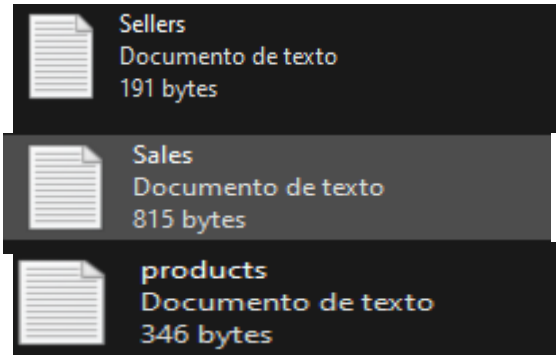
Se adjuntan las evidencias de compilación

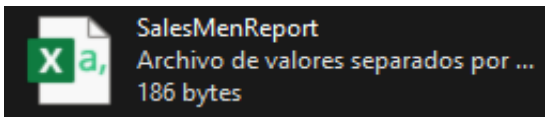
```
Javadoc  Declaration  Console ×
<terminated> GenerateInfoFiles (1) [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe  (15/04/2024, 12:39:03 p. m. – 12:39:04 p. m.) [pid: 13412]
Products file generated successfully.
Sellers file generated successfully.
Sales file generated successfully.
Salesmen report file generated successfully.
Product sales report file generated successfully.
Sales file for Santiago_Vera generated successfully.
Sales file for Luisa_Rodriguez generated successfully.
Sales file for Carlos_Martinez generated successfully.
Sales file for Lina_Velez generated successfully.
Sales file for Ana_Gomez generated successfully.
Sales file for Andrea_Bedoya generated successfully.
Sales file for Sebastian_Ruiz generated successfully.
Sales file for Juan_Perez generated successfully.
```

Los archivos son creados con éxito



Se evidencia en la ruta donde esta guardado el proyecto que crea los dos archivos uno de productos y otro con las ventas





se ingresa al archivo de productos se logra identificar la información contenida para este documento.

```
Archivo    Editar    Ver

P001,Refrigerador,1800000.0
P002,Lavadora,650000.0
P003,Microondas,350000.0
P004,Licuadora,200000.0
P005,Horno electrico,250000.0
P006,Tostadora,80000.0
P007,Cafetera,60000.0
P008,Aspiradora,120000.0
P009,Plancha,70000.0
P010,Ventilador,65000.0
P012,celular,70000.0
P011,Base Cama,65000.0
P013,Portatil,170000.0
P014,Diadema inalambrica,500000.0
```

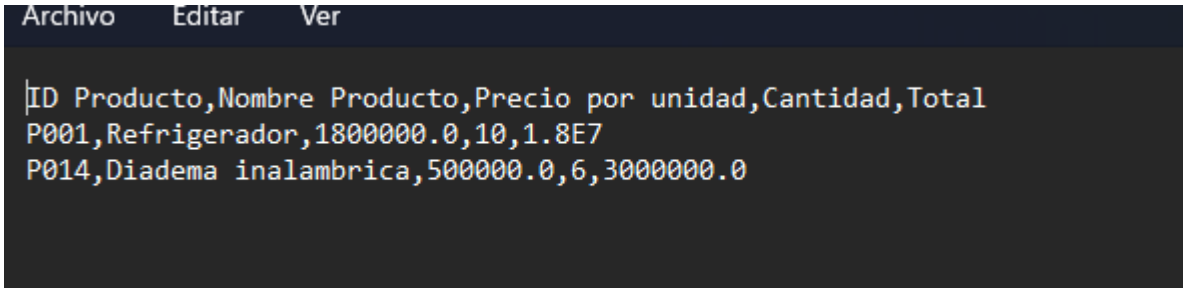
se ingresa al archivo de vendedores y se evidencia la información ingresada

```
CC,1001,Juan,Perez
CC,1002,Ana,Gomez
CC,1003,Carlos,Martinez
CC,1004,Luisa,Rodriguez
CC,1005,Pedro,Lopez
CC,1006,Andrea,Bedoya
CC,1007,Santiago,Vera
CC,1008,Sebastian,Ruiz
CC,1009,Lina,Velez
```

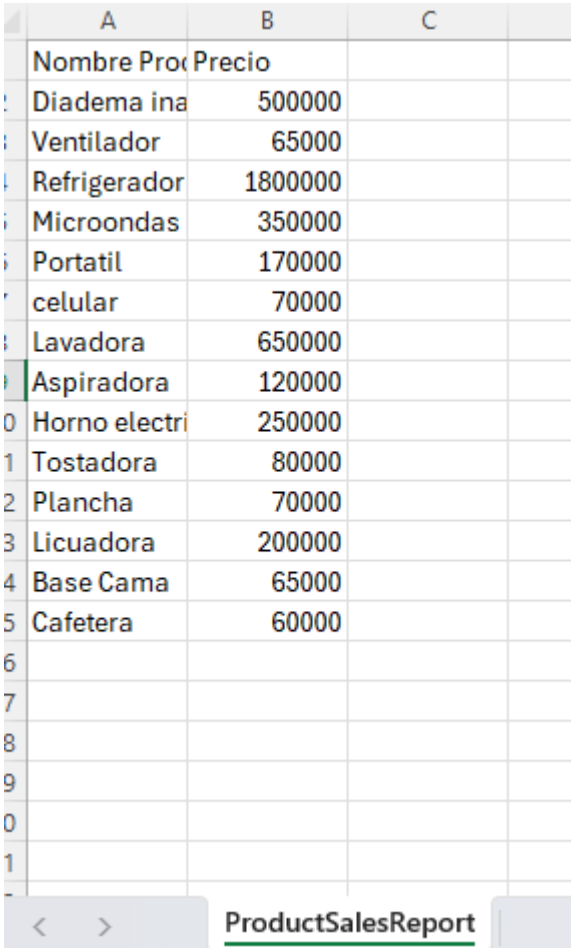
se ingresa al archivo de ventas y se evidencia la información ingresada

```
Archivo    Editar    Ver

P001,Refrigerador,1800000.0,10,,CC;Luisa;Rodriguez,1004,1.8E7
P002,Lavadora,650000.0,2,,CC;Andrea;Bedoya,1006,1300000.0
P003,Microondas,350000.0,4,,CC;Santiago;Vera,1007,1400000.0
P004,Licuadora,200000.0,3,,CC;Santiago;Vera,1007,600000.0
P005,Horno electrico,250000.0,8,,CC;Ana;Gomez,1002,2000000.0
P006,Tostadora,80000.0,4,,CC;Pedro;Lopez,1005,320000.0
P007,Cafetera,60000.0,1,,CC;Ana;Gomez,1002,60000.0
P008,Aspiradora,120000.0,1,,CC;Carlos;Martinez,1003,120000.0
P009,Plancha,70000.0,1,,CC;Santiago;Vera,1007,70000.0
P010,Ventilador,65000.0,10,,CC;Carlos;Martinez,1003,650000.0
P012,celular,70000.0,5,,CC;Juan;Perez,1001,350000.0
P011,Base Cama,65000.0,1,,CC;Ana;Gomez,1002,65000.0
P013,Portatil,170000.0,10,,CC;Santiago;Vera,1007,1700000.0
P014,Diadema inalambrica,500000.0,6,,CC;Luisa;Rodriguez,1004,3000000.0
```

se ingresa al archivo de un vendedor y se evidencia la información ingresada



Se envía la evidencia de cómo queda el Excel



	A	B	C
	Nombre Proc	Precio	
2	Diadema ina	500000	
3	Ventilador	65000	
4	Refrigerador	1800000	
5	Microondas	350000	
6	Portatil	170000	
7	celular	70000	
8	Lavadora	650000	
9	Aspiradora	120000	
10	Horno electri	250000	
11	Tostadora	80000	
12	Plancha	70000	
13	Licuadora	200000	
14	Base Cama	65000	
15	Cafetera	60000	
16			
17			
18			
19			
20			
21			

ProductSalesReport

Solución entrega 3 semana 7

1. Lo aprendido durante el desarrollo del proyecto.
2. Posibles aplicaciones en su vida profesional de las destrezas y conocimientos adquiridos y practicados durante el desarrollo del proyecto.
3. Las dificultades presentadas durante el desarrollo del proyecto

1. Lo aprendido durante el desarrollo del proyecto.

- Durante el curso, se desarrolló un proyecto colaborativo que, aunque inicialmente requería algunos conocimientos de programación en Java, se enriqueció gradualmente gracias a las tutorías del profesor y el aprendizaje autónomo de cada miembro del grupo. El grado de aprendizaje varió de acuerdo con el nivel de conocimientos previos de cada persona. Por ejemplo, para alguien sin experiencia en programación, los conocimientos básicos adquiridos representaron una introducción esencial al campo. En cambio, para aquellos con una base previa, el proyecto les permitió mejorar habilidades como el trabajo en equipo o profundizar en aspectos específicos del lenguaje Java que conocían superficialmente. Este aprendizaje es muy valioso y puede ser aprovechado en el futuro, siempre que el individuo así lo decida.
- Pudimos tener trabajo en grupo, aprender a dividirnos los puntos y seguir un correcto desarrollo del proyecto solicitado.
- Mejorar las habilidades no solo aprendidas no solo en esta materia si no a lo largo de los semestres, en nuestro caso que no tenemos mucha experiencia en programación represento una base inicial de como funcionara un proyecto en un posible empleo.

2. Posibles aplicaciones en su vida profesional de las destrezas y conocimientos adquiridos y practicados durante el desarrollo del proyecto.

- Aunque en este el equipo no se encuentra con un trabajo en temas referentes a la programación si sentimos que esta materia nos ayudó a fortalecer nuestras capacidades para mejorar el perfil, lo aprendido en esta materia es de suma importancia pues presenta
  las bases para poder empezar este arduo camino como lo puede llegar a ser la programación. Por otro lado, también se logró demostrar que la programación se puede emplear en casi cualquier cosa como lo fue un registro de ventas en un supermercado o un archivo plano con la información referente a los trabajadores de dicho lugar.

- Respecto al lenguaje de programación si esperamos poder afianzar más los conocimientos y poder dar uso de ellos en un futuro con un empleo a fin.

- Creemos que sabiendo cuales son las destrezas y debilidades de cada miembro del grupo nos podremos dividir el proyecto dada esas cualidades.
- Los conocimientos adquiridos con el lenguaje java esperamos lograr aplicarlos en nuestro ámbito profesional generando automatizaciones, puesto que el manejo de grandes cantidades de datos está ahora en su auge y podemos aplicar las destrezas obtenidas, con las cuales mejoraremos nuestras aptitudes profesionales.
- Con los conocimientos adquiridos, en archivos planos podemos aplicar en nuestras vidas cotidianas grandes mejoras asi no tengamos mucho que ver con el mundo de la programación.

3. Las dificultades presentadas durante el desarrollo del proyecto

- Poco conocimiento en lenguaje de programación, lo cual dificulto bastante la ejecución proyecto.
- La diferencia horaria para cada reunión o encuentro con el grupo.
- El manejo del IDE Eclipse generó que en el desarrollo del proyecto se presentaran algunos errores de compilación, por mala manipulación del entorno o desconocimiento de su configuración.
- Poder llegar a identificar rápidamente los errores en la lógica de la programación, más que en la sintaxis del código.
- Entregar el código en el idioma inglés, y los comentarios eso fue un reto grande, porque ninguno del grupo era teso en este idioma.

Podras ver este proyecto completo en el link

https://github.com/leogomez01/Sub_Grupo4_Concepto_Programacion