```java
/*
 * El kiosco AllDay necesita implementar un sistema que le permita administrar los ingresos diarios.

Actualmente  está generando diversas promociones por cantidades del mismo producto.

Por ejemplo, si se compran 6 unidades de coca cola, el subtotal a pagar por las mismas es $270 ($300 sería el precio por las 6 unidades, $30 el descuento generado)

Se pide como ingreso el DNI del cliente para comenzar a procesar las ventas.
El DNI ingresado como 0 marca el fin de datos de todas las ventas.

Luego, se realiza el ingreso de los siguientes datos por cada producto deseado:
Código de Producto (“g” = Gaseosa, “a” = Alfajor, “p” = Pancho, “f” para fin de ingreso).
Cantidad (entero positivo, con un máximo de 15)

El ingreso de “f” como código de producto marca el fin de ingreso de productos a esa venta.

Los precios unitarios de los productos son: 50, 80 y 120 respectivamente.

Se debe mostrar por cada venta, el importe a pagar en pesos, que se calcula a través de la suma de los subtotales de cada producto.
Luego, se solicitará nuevamente el DNI para comenzar a procesar una nueva venta.

Al finalizar la carga de todas las ventas, se debe mostrar por pantalla:
a) Cantidad de ventas procesadas
b) ¿Cuál fue el DNI del cliente, que realizó la venta con mayor monto? Suponer único máximo
c) ¿Cuál fue el producto con más cantidad de unidades solicitadas?
d) Recaudación total del kiosco en $


 */

package ort.thp.parcial1;

import java.util.Scanner;

public class AllDay {

    private static Scanner input = new Scanner(System.in);

    private static final String MSJ_INICIAL = "Bienvenido/a al kiosco AllDay!";

    private static final String COD_FIN = "0";
    private static final int MAX_PROD_POR_UNIDAD = 15;

    private static final double PRECIO_PRODUCTO_1 = 50;
    private static final double PRECIO_PRODUCTO_2 = 80;
    private static final double PRECIO_PRODUCTO_3 = 120;

    private static final String COD_PRODUCTO_1 = "G";
    private static final String COD_PRODUCTO_2 = "A";
    private static final String COD_PRODUCTO_3 = "P";
    private static final String COD_FIN_VENTA = "F";

    private static final String NOMBRE_PRODUCTO_1 = "Gaseosa mini";
    private static final String NOMBRE_PRODUCTO_2 = "Alfajor";
    private static final String NOMBRE_PRODUCTO_3 = "Pancho";

    private static final int SIN_DESCUENTO = 0;
    private static final int CANT_DESCUENTO_MIN = 1;
    private static final int CANT_DESCUENTO_MED = 3;
    private static final int CANT_DESCUENTO_MAX = 7;
    private static final double DESCUENTO_MAXIMO = 0.15;
    private static final double DESCUENTO_MEDIO = 0.1;
    private static final double DESCUENTO_MIN = 0.05;

    private static final String MSJ_DNI = "Por favor, ingrese a continuación el DNI del cliente (el " + COD_FIN
            + " marca fin de datos): ";
```

```java
    private static final String MSJ_CODIGO = "Ingrese el código del producto: ";
    private static final String MSJ_CANTIDAD_UN_PRODUCTO = "Ingrese la cantidad de unidades de ese item que quiera comprar: ";
    private static final String MSJ_IMPORTE_VENTA = "El importe a pagar en pesos por esta venta es: $ ";
    private static final String MSJ_FIN_VENTA = "Fin de la Venta";
    private static final String MSJ_VENTA_NULA = "No se registraron movimientos en esta venta!";

    private static final String MSJ_FINAL_VENTAS_TOTALES = "La cantidad de ventas totales es: ";
    private static final String MSJ_FINAL_MAYOR_VENTA = "El DNI del cliente que realizó la mayor venta es: ";
    private static final String MSJ_FINAL_RECAUDACION = "La recaudación total del kiosco hoy es de $";
    private static final String MSJ_VENTAS_NULAS = "No se registraron movimientos durante el día!";

    public static void main(String[] args) {
        // Declaración de variables
        String dni;
        String dniMayorVenta;
        int cantidadDeVentasTotales;

        double precioTotalVenta;
        double maximoTotalVenta;
        double recaudacionDiaria;

        // Inicializar variables generales
        cantidadDeVentasTotales = 0;
        recaudacionDiaria = 0;

        maximoTotalVenta = -Double.MAX_VALUE;
        dniMayorVenta = "";

        // Mensaje inicial de bievenida
        System.out.println(MSJ_INICIAL);

        /* - CICLO GENERAL - POR VENTA */
        dni = pedirDni();
        while (!dni.equals(COD_FIN)) {
            // Inicializar el acumulador que obtiene el precio total de la venta
            precioTotalVenta = procesoVenta();
            // Verifico si se compro algo
            if (precioTotalVenta > 0) {
                // Muestro por pantalla el importe total de la venta
                mostrarImporte(precioTotalVenta);
                // Incremento la cantidad de ventas
                cantidadDeVentasTotales++;
                // Acumulo el total de la venta en la recaudacion del kiosco
                recaudacionDiaria += precioTotalVenta;
                // Verifico si este cliente realizo la compra con mayor importe
                if (precioTotalVenta > maximoTotalVenta) {
                    maximoTotalVenta = precioTotalVenta;
                    dniMayorVenta = dni;
                }
            } else {
                System.out.println(MSJ_VENTA_NULA);
            }
            // Vuelvo a requerir el ingreso del DNI
            dni = pedirDni();
        }
        // Verifico si hubo ventas
        if (cantidadDeVentasTotales > 0) {
            // Muestro la información general del día
            mostrarInformacionGeneral(cantidadDeVentasTotales, dniMayorVenta,
            recaudacionDiaria);
        } else {
            System.out.println(MSJ_VENTAS_NULAS);
        }
        // Cerramos la conexión con el teclado
        input.close();
    }
```

```java

    /**
     * Realiza el procesamiento de una venta mientras se ingrese un codigo de
     * producto.
     *
     * @return el importe de la venta.
     */
    private static double procesoVenta() {
        int unidades;
        String codigoProducto;
        double precioUnitario;
        double subtotal;
        double descuentoAAplicar;
        double precioTotalVenta = 0;
        /* - CICLO INTERNO - POR PRODUCTO */
        codigoProducto = pedirCodigoProducto();
        while (!codigoProducto.equals(COD_FIN_VENTA)) {
            unidades = pedirUnidades();
            // Obtengo precio unitario y descuento en base a los datos
            // ingresados
            precioUnitario = obtenerPrecioUnitario(codigoProducto);
            descuentoAAplicar = obtenerDescuentoPorCantidad(unidades);
            // Genero el subtotal de ese producto
            subtotal = precioUnitario * unidades * (1 - descuentoAAplicar);
            // Acumulo el subtotal al total de la venta
            precioTotalVenta += subtotal;
            // Vuelvo a requerir el ingreso del código de producto
            codigoProducto = pedirCodigoProducto();
        }
        return precioTotalVenta;
    }

    /**
     * Pide la candidad de unidades de un producto determinado
     *
     * @return la cantidad, siempre en el rango deseado.
     */
    private static int pedirUnidades() {
        return pedirEnteroEnRango(MSJ_CANTIDAD_UN_PRODUCTO, 1, MAX_PROD_POR_UNIDAD);
    }

    /**
     * Pide el codigo de un producto asegurandose de que sea valido
     *
     * @return
     */
    private static String pedirCodigoProducto() {
        System.out.println("\n*** Producto y precio unitario ***");
        mostrarProducto(COD_PRODUCTO_1, NOMBRE_PRODUCTO_1, PRECIO_PRODUCTO_1);
        mostrarProducto(COD_PRODUCTO_2, NOMBRE_PRODUCTO_2, PRECIO_PRODUCTO_2);
        mostrarProducto(COD_PRODUCTO_3, NOMBRE_PRODUCTO_3, PRECIO_PRODUCTO_3);
        System.out.println(COD_FIN_VENTA + " - " + MSJ_FIN_VENTA);
        String codProd = pedirTextoNoVacio(MSJ_CODIGO).toUpperCase();
        while (!esCodigoValido(codProd)) {
            System.out.println("ERROR! El codigo de producto ingresado es
            incorrecto");
            codProd = pedirTextoNoVacio(MSJ_CODIGO).toUpperCase();
        }
        return codProd;
    }

    /**
     * Muestra por pantalla el Codigo, el nombre y el precio unitario en pesos del
     producto.
     * @param nombreProducto
     * @param nombreProducto1
     * @param precioProducto
     */
    private static void mostrarProducto(String codigoProducto, String
    nombreProducto, double precioUnitario) {
        System.out.println(codigoProducto + " - " + nombreProducto + " ($" +
        precioUnitario + ")");
    }
```

```java

    /**
     * Verifica que el codigo recibido corresponda a un producto o sea la marca
     * de final de venta.
     *
     * @param codProd
     * @return un booleano que indica si lo recibido es valido.
     */
    private static boolean esCodigoValido(String codProd) {
        return codProd.equals(COD_PRODUCTO_1) || codProd.equals(COD_PRODUCTO_2) ||
        codProd.equals(COD_PRODUCTO_3)
                || codProd.equals(COD_FIN_VENTA);
    }

    /**
     * Pide un DNI asegurandose de que no este vacio ni mida mas de 8 digitos
     *
     * @return
     */
    private static String pedirDni() {
        return pedirStringConLongitudMaxima(MSJ_DNI, 8);
    }

    /**
     * Función que obtiene el precio unitario de un producto a partir de su
     * código
     *
     * @param codigo
     *            El código del producto
     * @return El precio unitario del producto ingresado
     */
    private static double obtenerPrecioUnitario(String codigo) {
        double precio;

        switch (codigo) {

        case COD_PRODUCTO_1:
            precio = PRECIO_PRODUCTO_1;
            break;

        case COD_PRODUCTO_2:
            precio = PRECIO_PRODUCTO_2;
            break;

        case COD_PRODUCTO_3:
            precio = PRECIO_PRODUCTO_3;
            break;

        default:
            precio = 0;
            break;
        }

        return precio;

    }

    /**
     * Función que obtiene el descuento a aplicar en base a la cantidad de
     * productos solicitada
     *
     * @param cantidad
     *            La cantidad de unidades del producto solicitado
     * @return El descuento a aplicar
     */
    private static double obtenerDescuentoPorCantidad(int cantidad) {
        double descuento = SIN_DESCUENTO;
        if (cantidad >= CANT_DESCUENTO_MAX) {
            descuento = DESCUENTO_MAXIMO;
        } else if (cantidad >= CANT_DESCUENTO_MED) {
            descuento = DESCUENTO_MEDIO;
        } else if (cantidad >= CANT_DESCUENTO_MIN) {
            descuento = DESCUENTO_MIN;
```

```
273             }
274             return descuento;
275         }
276 
277         /**
278          * Procedimiento que muestra el importe a pagar por cada una de las ventas
279          *
280          * @param importe
281          *            El importe total a pagar por una venta
282          */
283         private static void mostrarImporte(double importe) {
284             System.out.println(MSJ_IMPORTE_VENTA + importe);
285         }
286 
287         /**
288          * Procedimiento que muestra los valores totales obtenidos durante el dia en
289          * el Kiosco
290          *
291          * @param cantidadTotalDeVentas
292          *            La cantidad total de ventas procesadas durante el día
293          * @param dniMayorVenta
294          *            El DNI del cliente que realizo la compra de mayor importe
295          * @param recaudacion
296          *            La recaudacion total del local
297          */
298         private static void mostrarInformacionGeneral(int cantidadTotalDeVentas, String
             dniMayorVenta, double recaudacion) {
299             System.out.println(MSJ_FINAL_VENTAS_TOTALES + cantidadTotalDeVentas);
300             System.out.println(MSJ_FINAL_MAYOR_VENTA + dniMayorVenta);
301             System.out.println(MSJ_FINAL_RECAUDACION + recaudacion);
302         }
303 
304         // *** FUNCIONES DISPONIBLES, USALAS DONDE CREAS CONVENIENTE ***
305 
306         /**
307          * Funcion que pide por teclado la carga de un string con longitud maxima.
308          *
309          * @param mensaje
310          * @param longitudMaxima
311          * @return EL TEXTO
312          */
313         private static String pedirStringConLongitudMaxima(String mensaje, int
             longitudMaxima) {
314             String dni;
315             System.out.println(mensaje);
316             dni = input.nextLine();
317             while (dni.length() > longitudMaxima || dni.isEmpty()) {
318                 System.out.println("ERROR! El dato solicitado no debe estar vacio ni
                     contener mas de " + longitudMaxima
319                         + " digitos\r\n" + mensaje);
320                 dni = input.nextLine();
321             }
322             return dni;
323         }
324 
325         /**
326          * Funcion que pide por teclado la carga de un string validando que no este
327          * vacio.
328          *
329          * @param mensaje
330          * @return El texto ingresado, nunca vacio.
331          */
332         private static String pedirTextoNoVacio(String mensaje) {
333             String dato;
334             do {
335                 System.out.print(mensaje);
336                 dato = input.nextLine();
337             } while (dato.isEmpty());
338             return dato;
339         }
340 
341         /**
342          * Funcion que pide por teclado un entero con rango minimo y maximo
```

```
343         *
344         * @param msj
345         *            El mensaje a mostrar por pantalla
346         * @param min
347         *            El valor mínimo aceptado
348         * @param max
349         *            El valor máximo aceptado
350         * @return El entero solicitado
351         */
352        private static int pedirEnteroEnRango(String msj, int min, int max) {
353            int entero;
354            System.out.println(msj);
355            entero = input.nextInt();
356            input.nextLine();
357            while (entero < min || entero > max) {
358                System.out.println("ERROR ! " + msj);
359                entero = input.nextInt();
360                input.nextLine();
361            }
362            return entero;
363        }
364    }
365
```