DISEÑO DE CLASES PARA PEPSICO INC

Clases para almacenar el **PERSONAL** de la empresa:

Clase **Persona** Abstracta

```
public abstract class Persona {

    private int dni;
    private String nombreApellido;
    private String usuario;
    private String pass;
```

Clase **Empleado,** que hereda de **Persona**

```
public class Empleado extends Persona {

    private int cantPedidos;
    private int antiguedad;
    private int comision;
```

Clase **Admin,** que hereda de **Persona**

```
public class Admin extends Persona {

    String categoria;

    // jerarquia de las categorias del Admin
    //A: puede realizar acciones con cualquier dato de la empresa (excepto admins)
    //B: puede realizar acciones con cualquier dato de la empresa, excepto con los empleados
    //B: puede realizar acciones solo con pedidos y camiones
```

Clases para almacenar los **PRODUCTOS** de la empresa:

Clase **Producto** Abstracta

```
public abstract class Producto implements Serializable{

    private int idProducto;
    private String nombre;
    private String tipo;
    // en bebida, el tipo es: agua, gaseosa, jugo, etc
    // en snack, el tipo es: papas, nachos, palitos, etc
    // en cereal, el tipo es: chocolate, avena, miel, et
    private int stockCajas;
```

Clase **Bebida**, que hereda de **Producto**

```
public class Bebiba extends Producto {

    private String sabor;
    // sabor:   si es gaseosa, coca, sevenup, fanta, etc.
    //          si es jugo: naranja, manzana, multifruta
    //          si es agua: natural, soda
```

Clase **Snack**, que hereda de **Producto**

```
public class Snack extends Producto {

    private boolean esSalado;
```

Clase **Cereal**, que hereda de **Producto**

```
public class Cereal extends Producto {

    private boolean azucar;
```

Clases para almacenar los **PEDIDOS** de la empresa y sus envíos:

Clase **Caja**, que almacena un tipo de Producto.

```
public class Caja implements Serializable {

    private Producto tipoProducto;
    private static int cantProductosPorCaja = 10;
    private String tamanioProductos;     // chico, mediano o grande
```

Clase **Pedido**, que almacena un ArrayList de Cajas.

```
public class Pedido implements Serializable {

    private ArrayList<Caja> arrayListCajas;
    private String destinatario;
    private String direccion;
```

Clase **Camión**, que almacena un pedido para realizar su envío.

```
public class Camion implements Serializable {

    private String patente;
    private String Marca;
    private int modelo;
    private boolean disponible;
    private Pedido pedido;
```

Clases con métodos para el **FUNCIONAMIENTO Y USO** de Archivos de Datos y de Json, y la clase Fila:

Clase **FilesUtiles**, para leer y grabar archivos.

```
public class FilesUtiles {

// clase contenedora de los metodos de grabar y
// leer archivos para distintos tipos de datos.
```

Clase **JsonUtiles**, para leer y grabar archivos.json.

```
public class JsonUtiles {

// clase contenedora de los metodos de grabar y
// leer archivos.json
```

Clase **Fila<T>**, para almacenar distintos objetos durante el programa.

```
public class Fila<T> {

// clase generica que simula una Fila y
// su funcionamiento a traves de nodos,
// para almacenar en ellas objetos de tipo Pedido

    private Nodo<T> primero, ultimo;
```

Clase **Nodo**, que se usa para el funcionamiento de la clase **Fila**.

```
public class Nodo<T> {

    T info;
    Nodo<T> siguiente;
```

Clases para **EXCEPCIONES PERSONALIZADAS**, las cuales heredan de Exception:

Clase **UsuarioIncorrecto**, que se lanza en el login.

```
public class UsuarioIncorrecto extends Exception{
```

Clase **PasswordIncorrecto**, que se lanza en el login.

```
public class PasswordIncorrecto extends Exception{
```

Clase **FaltaStock**, que se lanza cuando se quiere hacer un pedido y no hay stock suficiente.

```
public class FaltaStock extends Exception{
```

Clase **FaltaPedidos**, que se lanza cuando se quiere preparar un pedido pero no hay ninguno para hacerlo.

```
public class FaltaPedidos extends Exception{
```

Clase **FaltaCamiones**, que se lanza cuando se quiere usar un camión para enviar un pedido pero no hay ninguno disponible para hacerlo.

```
public class FaltaCamiones extends Exception{
```

Clases para el **FUNCIONAMIENTO E INVOCACIÓN DE LOS MÉTODOS**, las cuales le dan vida al programa:

Clase **App**, le da funcionamiento al programa a través de distintos tipos de listas de Pedidos, Productos, Empleados, Admins y Camiones.

```
public class App{

    private Fila<Pedido> pedidos;
    private ArrayList<Producto> productos;
    private HashSet<Empleado> empleados;
    private HashSet<Admin> admins;
    private HashMap<String,Camion> camiones;
    private Fila<Camion> camionesFuera;
```

Clase **Main**,donde se invoca el método **menu**, que es el más importante.

```
public class Main {

    static Scanner scan;

    public static void main(String[] args) {

        scan = new Scanner(System.in);

        App app = new App();

        menu(app);

        scan.close();
    }
```