

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1

CATEDRÁTICO: ING. WILLIAM ESTUARDO ESCOBAR ARGUETA

TUTOR ACADÉMICO: JOSUÉ RODOLFO MORALES CASTILLO



EDUARDO SEBASTIAN GUTIERREZ

CARNÉ: 202300694

SECCIÓN: B

GUATEMALA, 18 DE SEPTIEMBRE DEL 2,024

ÍNDICE

ÍNDICE	1
INTRODUCCIÓN	2
OBJETIVOS	2
1. GENERAL	2
2. ESPECÍFICOS	2
ALCANCES DEL SISTEMA	3
ESPECIFICACIÓN TÉCNICA	4
• REQUISITOS DE HARDWARE	4
• REQUISITOS DE SOFTWARE	4
DESCRIPCIÓN DE LA SOLUCIÓN	5
LÓGICA DEL PROGRAMA	6

INTRODUCCIÓN

El objetivo de este manual es brindarle al usuario una documentación con información técnica acerca del funcionamiento del programa IPCTEC, un programa para la gestión de la producción masiva de diferentes productos. Con este manual, el usuario entenderá la lógica detrás del programa y cómo los elementos que lo componen interactúan entre sí para crear un programa consistente y robusto.

OBJETIVOS

1. GENERAL

- 1.1. Brindar al usuario información técnica sobre la lógica que compone el programa

2. ESPECÍFICOS

- 2.1. Describir las clases, métodos y variables más importantes del programa
- 2.2. Indicar las librerías utilizadas para realizar el programa

ALCANCES DEL SISTEMA

Con este manual, el usuario será capaz de entender la lógica que hay detrás del programa que utilizará, aún si no conoce los fundamentos de programación orientada a objetos. Por medio de este manual el usuario conocerá las clases, métodos, funciones, procedimientos y variables declaradas en su programa y la forma en que éstas interactúan entre sí para darle la funcionalidad y robustez al programa. Así mismo, se especificarán los tipos de archivos que el usuario puede cargar en el sistema y el formato que estos deben tener para que el programa funcione de manera correcta. Por último, se le hará saber al usuario qué requisitos debe cumplir su dispositivo para ejecutar el programa de forma eficiente.

ESPECIFICACIÓN TÉCNICA

● REQUISITOS DE HARDWARE

- Configuraciones mínimas de hardware:
 - Procesador: Intel Pentium III a 800 MHz o equivalente
 - Memoria: 512 MB
 - Espacio en disco: 1 GB de espacio libre en disco
- Configuración recomendada de hardware:
 - Procesador: Intel Core i5 9ª generación o posterior
 - Memoria: 4GB o superior
 - Espacio en disco: 2 GB de espacio libre en disco o más

● REQUISITOS DE SOFTWARE

- Requisitos mínimos de software:
 - Windows 7 Professional
 - Ubuntu 9.1
- Requisitos recomendados de software
 - Windows 10 Home o posteriores
 - Ubuntu 15.04 o posteriores

DESCRIPCIÓN DE LA SOLUCIÓN

- La solución propuesta se basa en el uso de hilos para gestionar el progreso de la producción. Para que los datos persistan dentro de la aplicación, se utilizó serialización de objetos. El objeto principal es el objeto de tipo Producto, el cual tiene varios atributos que son los que se muestran en la tabla. Para manejar la lista de productos, se utilizó un ArrayList estático para que pueda ser accedido desde cualquier parte del programa.

La aplicación fue desarrollada bajo el patrón de diseño MVC, en donde M es modelo, V es vista y C es controlador. En el paquete modelo se encuentran todas las clases que contienen los “moldes”, la lógica detrás del programa. En el paquete vista se encuentran todas las interfaces, los JFrame, es decir, todos los elementos gráficos que el usuario ve en pantalla. En el paquete de controlador se encuentran las clases que capturan los eventos de los botones y otros elementos del programa.

El uso de este patrón de diseño permite separar por piezas el programa, además facilita el mantenimiento y futuras modificaciones.

LÓGICA DEL PROGRAMA

❖ EstacionDeCarga

```
import javax.swing.table.DefaultTableModel;
```

➤ Librerías

La librería `java.swing.table.DefaultTableModel` sirve para darle el modelo a la tabla y para que se le puedan agregar filas.

➤ Variables Globales de la clase `_(EstacionDeCarga)`

```
public DefaultTableModel dtm_product;
```



El `DefaultTableModel` se usa para definir un modelo y dárselo a una o más tablas.

➤ Función Main

Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  `16 public EstacionDeCarga()`
Constructor de la clase
-  `28 public void table products()`
Función para crear la tabla de productos

❖ EstacionDeResultados

```
3  import javax.swing.JOptionPane;
```

➤ Librerías

La librería `java.swing.JOptionPane` sirve para mostrar ventanas emergentes.

➤ Variables Globales de la clase `_`(EstacionDeResultados)

Esta clase no contiene variables globales.

➤ Función Main

Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- ```
13 public EstacionDeResultados ()
```

Constructor de la clase

- ```
213 private void regresar btnActionPerformed(java.awt.event.ActionEvent evt)
```

Función para capturar el evento al presionar el botón
Regresar.

❖ EstacionDeTrabajo

Sin librerías adicionales.

➤ Librerías

Sin librerías.

➤ Variables Globales de la clase _(EstacionDeTrabajo)

Sin variables globales.

➤ Función Main

Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- `11 public EstacionDeTrabajo()`

Constructor de la clase

- `24 private void initComponents()`

Función para indicar los componentes del JFrame

❖ **Producir**

Sin librerías adicionales.

➤ **Librerías**

Sin librerías.

➤ **Variables Globales de la clase _(Producir)**

Sin variables globales.

➤ **Función Main**

Esta clase no contiene función main

➤ **Procedimientos, métodos y Funciones utilizadas**

A continuación se dará una explicación general de lo que hace cada función:

- 10  `public Producir()`

Constructor de la clase

❖ Actualizar_tabla

```
import java.util.List;|
import javax.swing.table.DefaultTableModel;
```

➤ Librerías

La librería `java.util.List` es utilizada para el manejo de listas dinámicas. La librería `java.swing.tabla.DefaultTableModel` se utiliza para dar formato y modelo a las tablas.

➤ Variables Globales de la clase `_(Actualizar_tabla)`


Sin variables globales

➤ Función Main

Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

● 12  `public void agregar_productos(DefaultTableModel dtm, List<Producto> productos)`

Función para agregar elementos del `ArrayList` a la tabla.

❖ Cargar_productosCSV

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.List;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
```

➤ Librerías

La librería java.io es utilizada para el flujo de datos, entrada y salida.

La librería java.util.List es para el manejo de listas dinámicas.

La librería java.swing se utiliza para importar elementos de la interfaz gráfica.

➤ Variables Globales de la clase _(Cargar_productosCSV)

```
private String ruta;
private boolean band = false;
```

La variable ruta guarda la ruta del archivo que el usuario seleccionó.


La variable band se usa para saber si el usuario seleccionó un archivo válido.

➤ Función Main


Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- 19  `public void elegir_archivo()`

Método para seleccionar el archivo

- 35  `public void Cargar_productos()`

Método para cargar los datos del csv al ArrayList y a la tabla.

❖ **Costo_producto**

Sin librería adicionales

➤ **Librerías**

Sin librerías.

➤ **Variables Globales de la clase _(Costo_producto)**


Sin variables globales.

➤ **Función Main**

Esta clase no contiene función main

➤ **Procedimientos, métodos y Funciones utilizadas**

A continuación se dará una explicación general de lo que hace cada función:

-  `public int costo(String descripcion)`

Método para calcular el costo de un producto según su material o color.

❖ Hilo_barra

```
import javax.swing.JLabel;  
import javax.swing.JOptionPane;  
import javax.swing.JProgressBar;  
import javax.swing.SwingUtilities;
```

➤ Librerías

La librería java.swing se utiliza para importar elementos de la interfaz gráfica.

➤ Variables Globales de la clase _(Hilo_barra)

```
private boolean band = true;  
private JProgressBar barra;  
private int tiempo;  
private double progress;  
private JLabel label;
```


Variables necesarias para el manejo de los hilos que controlan el progreso de la producción.

➤ Función Main


Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas




A continuación se dará una explicación general de lo que hace cada función:

- 22  `public Hilo barra(JProgressBar barra, String descripcion, JLabel label)`

Constructor para hilos de material y color.

- 69  `public Hilo barra(JProgressBar barra, JLabel label)`

Constructor para los hilos de empaquetado.

- 75    `@Override
public void run() {`

Método para arrancar los hilos.

```
public int getTiempo() {  
    return tiempo;  
}  
  
public JLabel getLabel() {  
    return label;  
}
```



Métodos getter.

❖ Hilo_contador

```
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
```

➤ Librerías

La librería java.util se utiliza para modificar los JprogressBar.

La librería java.swing se utiliza para importar elementos de la interfaz gráfica.

➤ Variables Globales de la clase _(Hilo_contador)

```
private int tiempo;
private JLabel label_sec;
private JLabel label_min;
```

La variable tiempo sirve para saber cuanto tiempo se va a ejecutar el hilo.


Los JLabel se utilizarán para el cronómetro.

➤ Función Main

Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  `19 public Hilo_contador(int tiempo, JLabel label_sec, JLabel label_min)`

Constructor del hilo.

-  `25 @Override`
 `public void run()`

Método para ejecutar el hilo.


```
public int getTiempo() {  
    return tiempo;  
}
```



Método getter.

❖ Hilo_contadorProductos

```
import javax.swing.JLabel;  
import javax.swing.JOptionPane;
```

➤ Librerías

La librería java.swing se utiliza para importar elementos de la interfaz gráfica.

➤ Variables Globales de la clase _(Hilo_contadorProductos)

```
private int tiempo1;  
private int tiempo2;  
private int tiempo3;  
private JLabel label_total;  
private JLabel label_contador;  
private int total;
```

Variables enteras para saber el tiempo de producción.


Los JLabel son para el contador de la cantidad producida

➤ Función Main

Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

●  `public Hilo_contadorProductos(int tiempo1, int tiempo2, int tiempo3, int totalDeProductos, JLabel label_total, JLabel label_contador)`

Constructor del hilo.

●  `@Override
public void run()`

Método para arrancar el hilo.

❖ Hilos_serie

```
import javax.swing.JOptionPane;  
import javax.swing.SwingUtilities;
```

➤ Librerías

La librería java.swing se utiliza para importar elementos de la interfaz gráfica.

➤ Variables Globales de la clase _(Hilos_serie)

```
private String material;  
private String color;  
  
int cantidadDeProductos;  
private EstacionDeTrabajo trabajo;  
private EstacionDeResultados resultados;
```

Las variables String son para almacenar el color y el material del producto a producir.

También hay una variable entera para la cantidad de productos a producir y dos objetos de tipo vista.

➤ Función Main

Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

```
21 public Hilos serie(String material, String color, int cantidadDeProductos, EstacionDeTrabajo trabajo, EstacionDeResultados resultados)
```

Constructor del hilo.

```
29 @Override  
public void run()
```

Método para arrancar el hilo.

❖ Lista_Productos

```
3  import java.util.List;  
4  import java.util.ArrayList;
```

➤ Librerías

Las librerías `java.util.List` y `java.util.ArrayList` son para el manejo de listas dinámicas.

➤ Variables Globales de la clase _(Lista_Productos)

```
public static List<Producto> productos = new ArrayList<>();
```

Lista estática de tipo `Producto`.

➤ Función Main

Esta clase no contiene función `main`

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- 17

```
public int Buscar producto(String codigo)
```

Método para buscar coincidencias de código en el `ArrayList`.

❖ Producto

```
3  import java.io.Serializable;
```

➤ Librerías

La librería `java.io.Serializable` es utilizada para serializar los objetos y poder guardarlos en archivos binarios.

➤ Variables Globales de la clase `_`(Producto)

```
private String codigo;  
private String nombre;  
private String material;  
private String color;
```

Variables de tipo `String`, que son los atributos que tiene el `Producto`.

➤ Función Main

Esta clase no contiene función `main`

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

```
public String getCodigo() {  
    return codigo;  
}  
  
public void setCodigo(String codigo) {  
    this.codigo = codigo;  
}  
  
public String getNombre() {  
    return nombre;  
}  
  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}  
  
public String getMaterial() {  
    return material;  
}  
  
public void setMaterial(String material) {  
    this.material = material;  
}  
  
public String getColor() {  
    return color;  
}  
  
public void setColor(String color) {  
    this.color = color;  
}
```

Métodos `setter` y `getter`.

❖ Productos_binarios

```
import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.List;
import javax.swing.JOptionPane;
```

➤ Librerías

La librería java.io es utilizada para el flujo de datos, entrada y salida.

La librería java.util.List es para el manejo de listas dinámicas.

La librería java.swing.JOptionPane se utiliza para mostrar ventanas emergentes.

➤ Variables Globales de la clase _(Productos_binarios)


Sin variables globales.

➤ Función Main


Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- 20  `public static void escribir_productoBinario()`

Método para guardar los datos en el archivo binario.

- 45  `public static void leer_productoBinario()`

Método para leer los datos del archivo binario.

❖ **Tiempo_producto**

Sin librerías extra.

➤ **Librerías**

Sin librerías.

➤ **Variables Globales de la clase _(Tiempo_producto)**


Sin variables globales.

➤ **Función Main**

Esta clase no contiene función main

➤ **Procedimientos, métodos y Funciones utilizadas**

A continuación se dará una explicación general de lo que hace cada función:

-  `public int tiempo(String descripcion)`

Método para calcular el tiempo que tarda un producto según su color y material.

❖ Cargar_productos

```
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;
```

➤ Librerías

La librería java.awt es utilizada para capturar eventos al hacer clic sobre algún botón.

➤ Variables Globales de la clase _(Cargar_productos)

```
private EstacionDeCarga view;
```


Objeto de tipo vista que es la que recibe los eventos.

➤ Función Main


Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  `22` `public Cargar productos(EstacionDeCarga view)`

Constructor de la clase.

-  `28` `@Override`
`public void actionPerformed(ActionEvent e)`

Función que captura los eventos al presionar los botones de las vistas.

❖ Producir_productos

```
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import javax.swing.JOptionPane;
```

➤ Librerías

La librería java.awt es utilizada para capturar los eventos.

La librería java.swing.JOptionPane se utiliza para mostrar ventanas emergentes.

➤ Variables Globales de la clase _(Producir_productos)

```
private Producir view;  
private EstacionDeCarga estacion;  
private EstacionDeTrabajo trabajo;
```


Objetos de tipo vista que reciben los eventos.

➤ Función Main


Esta clase no contiene función main

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  `public Producir_productos(Producir view, EstacionDeCarga estacion, EstacionDeTrabajo trabajo)`

Constructor de la clase.

-  `@Override`
`public void actionPerformed(ActionEvent e)`

Función que captura los eventos al presionar los distintos botones de las vistas.

❖ Main

Sin librerías extra.

➤ Librerías

Sin librerías.

➤ Variables Globales de la clase _(Main)

Sin variables globales.

➤ Función Main

```
17  public static void main(String[] args) throws ClassNotFoundException
```

Función main, en donde se crea una instancia de la vista y se le agrega el controlador.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- 17 public static void main(String[] args) throws ClassNotFoundException

Método main.