Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Programación de Computadoras 2 Sección: R

Ing. HERMAN IGOR VELIZ LINARES

Aux. José Eduardo Morales García

Manual de usuario

Pedro Pablo Raúl López Vargas

202004730

Contenido

Introducción

Se necesita un programa en lenguaje Java que simule las acciones que realiza “SIMIO” ya que se no se cuenta con un gran presupuesta para realizar simulaciones y animaciones, con el fin de optimizar procesos.

Se desarrollo el programa “Monkey” Que simula las acciones que “SIMIO”, para esto se ha planteado un sistema de 4 sectores en los cuales con un respectivo tiempo de entrada y salida se pretende simular el proceso que se llevaría acabo dentro de una empresa de empaquetado.

Objetivos

Objetivo General:

Aplicar los conocimientos adquiridos en el curso sobre el lenguaje java.

Objetivos específicos

Proporcionar una rápida y eficaz solución ala simulación.

Manejar las clases del proyecto.

Manejar correctamente los hilos.

Diccionario de clases

En el programa se ha utilizado diferentes clases y métodos para llevar a cabo el funcionamiento del programa:

1. Clase practica 2

Esta clase es la principal donde se declarara el panel form a utilizar.

```
public class Practica2 {


    public static void main(String[] args) {
        ventana ven = new ventana();
        ven.setVisible(true);

    }

}
```

2. Acción del botón “Iniciar simulación”

Dentro del botón se programó y se añadió condiciones para que el programa se lleve a cabo sin ningún problema.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {


    if (invens.getText().isEmpty() || produs.getText().isEmpty() || empas.getText().isEmpty() || salis.getText().isEmpty() || invenqs.getText().isEmpty() || p
        JOptionPane.showMessageDialog(null, "Complete todos los campos","Error", JOptionPane.WARNING_MESSAGE);
    }else{
        if (esNumeroEntero(invens.getText()) && esNumeroEntero(invenqs.getText()) && esNumeroEntero(produs.getText()) && esNumeroEntero(produqs.getText()) &&
    jPanel1.setVisible(false);
    mostrarpa();
    jButton1.setEnabled(false);
    hiloverde.setDato1(Integer.parseInt(invens.getText()));
    hilorosa.setDato2(Integer.parseInt(produs.getText()));
    valorvv=Integer.parseInt(invens.getText());
    valorro=Integer.parseInt(produs.getText());
    valorojo=Integer.parseInt(empas.getText());
    valorfin=Integer.parseInt(salis.getText());
    cronometro();
    Q1=Integer.parseInt(invens.getText())*Integer.parseInt(invenqs.getText())*30;
    Q2=Integer.parseInt(produs.getText())*Integer.parseInt(produqs.getText())*30;
    Q3=Integer.parseInt(empas.getText())*Integer.parseInt(empaqs.getText())*30;
    Q4=Integer.parseInt(salis.getText())*Integer.parseInt(saliqs.getText())*30;
    totalQ=Q1+Q2+Q3+Q4;
```

3. Hilo

Para llevar a cabo la ejecución del programa con tiempos se utilizaron hilos con los cuales fueron declarados 4 hilos para llevar a cabo la simulación del transporte de las pelotas.

Al ser 30 pelotas, se utilizo un for hasta que se llegara a un limite de 30.

A su vez se encuentra la variable iniciov que da inicio al otro sector para que se empiece a extraer las unidades.

```
        new Thread(new Runnable() {
            public void run() {
                try {
                    for (int i = 1; i <= 60; i++) { // Realiza el parpadero 5 veces
                        Thread.sleep(1000); // Espera 1 segundo
                        if (cirama.isVisible()) {
                            coni=29-contador1;
                            iniciocon.setText( coni+" ");
                            contador1=contador1+1;
                            cirama.setVisible(false);
                            System.out.println(contador1);
                            if (iniciov==1) {
                                iniciarverde();
                                iniciov=2;
                            }
                            mostrar1();
                            contadorinventario=inventarioi-inventariof;
                            jLabel12.setText("Inventario: "+contadorinventario);
                        } else {
                            cirama.setVisible(true);
                        }
                    }
                } catch (InterruptedException ex) {
                    ex.printStackTrace();
                } finally {
                    SwingUtilities.invokeLater(new Runnable() {
                        public void run() {
                            jButton1.setEnabled(true); // Habilita el botón al finalizar
                        }
                    });
                }
            }
        }).start();
```

4. Método mostrarpa

Este método sirve para mostrar el panel que contiene la simulación luego de completar los campos.

```
public void mostrarpa(){
    jPanel2.setVisible(true);
}
```

5. botón reporte

El botón reporte contiene el código que anade en forma de HTML al “reportesformn” los datos que se encontraban almacenados en las textboxes.

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    String textoreporte="<html><head></head><body><table><tr><th>Sector</th><th>Tiempo(s)</th><th>Costo de produccion (Q/s)</th><th>Costo sector</th></tr>";

    textoreporte=textoreporte+"<tr><td>"+ "Inventario "+"</td><td>"+invens.getText()+"</td><td>"+invenqs.getText()+"</td><td>"+Integer.parseInt(invens.getText
       "<tr><td>"+ "Empaquetado" +"</td><td>"+ empas.getText() +"</td><td>"+empaqs.getText()+"</td><td>"+Integer.parseInt(empas.getText())*Integer.parseInt(em
    report.setTextArea(textoreporte);
    report.setVisible(true);
    report.setTextlabel("Total Q."+totalQ+" Pedro Pablo Raul Lopez Vargas, 202004730");
}
```

6. botón regresar

Este botón mostraba nuevamente el panel inicla donde se pedía ingresar los datos y reinicar todos los valores.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    jPanel2.setVisible(false);
    jPanel1.setVisible(true);
    contador1=0;
    contador2=0;
    contador3=0;
    contador4=0;
    contador5=0;
    jLabel9.setText("00:00");
    Q1=0;
    Q2=0;
    Q3=0;
    Q4=0;
    totalQ=0;
    segundos=0;
    minutos=0;
}
```

7. método hilos

Se declararon los hilos en métodos para que al cumplir una condición fueran llamados para dar continuidad a las otras etapas.

```
public void iniciarverde(){
        new Thread(new Runnable() {
        public void run() {
    try {
            for (int x = 0; x < 30; x++) {
            System.out.println(valorvv*1000);
            Thread.sleep(valorvv*1000);
            contador2=contador2+1;
            mostrar2();
            contadorinventario=inventarioi-inventariof;
            jLabel12.setText("Inventario: "+contadorinventario);
            contadorproduccion=produccioni-produccionf;
            jLabel13.setText("Produccion: "+contadorproduccion);
                if (inicior==1) {
                    iniciarmora();
                    inicior=2;
                }
        }
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    }
        }
    }).start();
}
```

8.Metodo SetImageLabel

Este método utiliza el Icon y se utiliza para que se le pueda añadir una imagen especifica ingresando la dirección local en el código.

```
private void SetImageLabel(JLabel labelName, String root){
    ImageIcon image = new ImageIcon(root);
    Icon icon = new ImageIcon( image.getImage().getScaledInstance(labelName.getWidth(), labelName.getHeight(), Image.SCALE_DEFAULT))
    labelName.setIcon(icon);
    this.repaint();
}
```

9. Métodos mostrar

Los métodos mostrar están conformados por los contadores, que al cumplirse cada condición se muestra el circulo que representa el ítem.

También están conformados por variables que dan inicio a otros métodos y otros que proporcionan el numero de valores que se encuentra dentro de cada panel.

```
public void mostrar1(){
    switch(contador1){
        case 1:
            jLabel15.setVisible(true);
            inventarioi=inventarioi+1;
            System.out.println("1");
            iniciov=1;
            break;
        case 2:
            jLabel16.setVisible(true);
            inventarioi=inventarioi+1;
            break;
        case 3:
            jLabel17.setVisible(true);
            inventarioi=inventarioi+1;
            break;
        case 4:
            jLabel18.setVisible(true);
            inventarioi=inventarioi+1;
            break;
        case 5:
            jLabel19.setVisible(true);
            inventarioi=inventarioi+1;
            break;
        case 6:
            jLabel20.setVisible(true);
            inventarioi=inventarioi+1;
            break;
        case 7:
            jLabel21.setVisible(true);
            inventarioi=inventarioi+1;
```

10. Métodos cronometro

Este método funciona para darle el contador de tiempo a todo el programa

```
    public void cronometro(){
        if (!corriendo) {
            inicioHilo=true;
            corriendo=true;
            iniciarCronometro();
        }
    }
public void iniciarCronometro(){
    if (inicioHilo==true) {
        System.out.println("inicia el hilo");
            Hilo miHilo = new Hilo(jLabel9);
            miHilo.start();
    }
}
```

11. Método detener

Este método detiene el contador una vez la simulación ha terminado.

```
public void detener(){
    corriendo=false;
    inicioHilo=false;
}
```

12. Método esNumeroEntero

Este método es utilizado para validar que solo se ingresen números enteros.

```
public boolean esNumeroEntero(String cadena){
    try{
        Integer.parseInt(cadena);
        return true;
    }catch(NumberFormatException e){
        return false;
    }
}
```

13. Constructor hilo Hilo

Este constructor inicializa la variable de la clase HILO.

```
public Hilo(JLabel cronometro){
    this.eti=cronometro;
}
```

14. Método ejecturarHilo

Este método ejecuta el contador de tiempo que lleva la simulación en progreso.

```
    public void ejecturaHilo(int x){
//        System.out.println(x+" " + Thread.currentThread().getName());
        ventana.segundos++;
        if (ventana.segundos>59) {
            ventana.segundos=0;
            ventana.minutos++;
        }
        String textSeg="",textMin="",textHora="";
        if (ventana.segundos<10) {
            textSeg="0"+ventana.segundos;
        }else{
           textSeg=""+ventana.segundos;
        }
        if (ventana.minutos<10) {
            textMin="0"+ventana.minutos;
        }else{
           textMin=""+ventana.minutos;
        }

        String reloj=textHora+":"+textMin+":"+textSeg;
        eti.setText(reloj);
    }
```

15. Método setTextArea

Este método obtiene la información obtenida en la clase ventana y la muestra en formato HTML en el textpanel.

```
    public void setTextArea(String texto){
        jTextPanel.setText(texto);

    }
```

16. Método setTextLabel

Este método muestra los valores almacenados del total y datos del estudiante en una label, estos datos los obtiene de la clase ventana.

```
    }
    public void setTextlabel(String labelt){
        jLabel1.setText(labelt);
    }
```