Universidad San Carlos de Guatemala

Facultad de Ingeniería

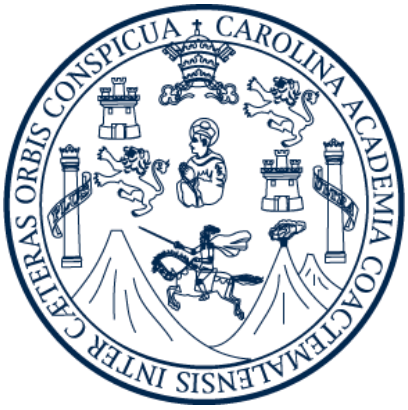
Escuela de Ingeniería en Ciencias y Sistemas

Introducción a la Programación y Computación 1, Sección A

Ing. Marlon Francisco Orellana López

Auxiliar: José Daniel Fajardo

Primer Semestre 2023



Manual Técnico – Práctica 2

Nombre: Carlos Manuel Lima y Lima

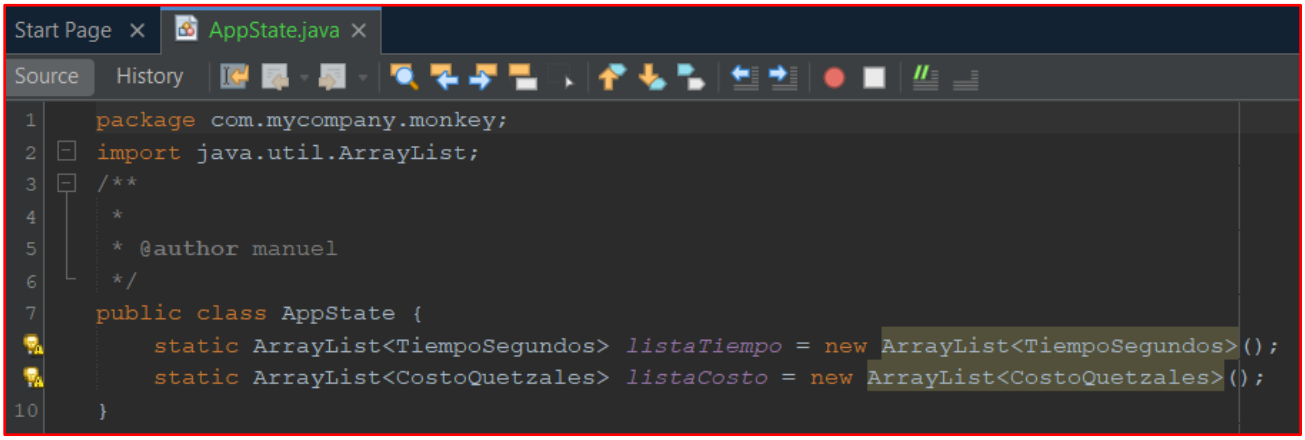
Registro Académico: 202201524

CUI: 3009368850101

Guatemala, 25 de marzo del 2023

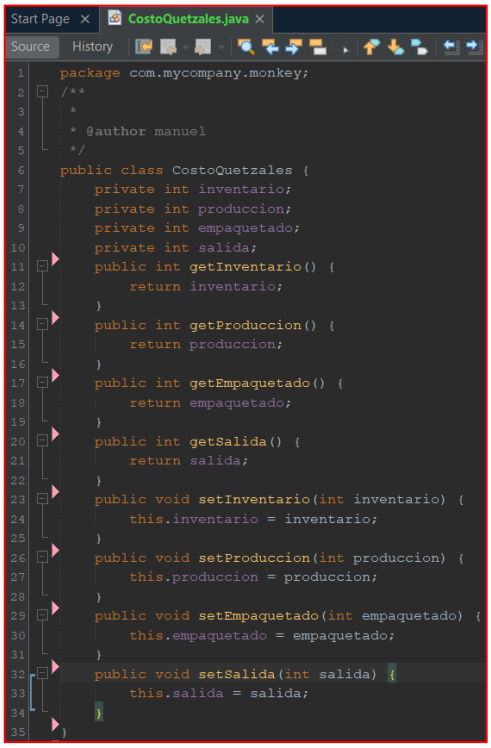
1. Clase AppState

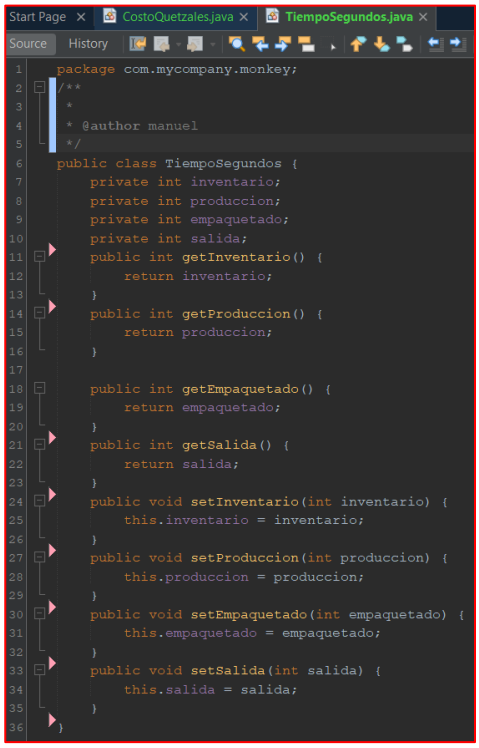
Se creó la clase AppState en donde se declararon 2 ArrayList del tipo TiempoSegundos y CostoQuetzales. Dichos ArrayList son utilizados para almacenar los datos registrados por el usuario al momento de iniciar la simulación.



2. Clases Del Tipo Objeto (CostoQuetzales, TiempoSegundos)

Se declaró 4 variables del tipo entero para almacenar los datos que el usuario registre y se realizados los métodos de get y set para enviar y obtener datos.





3. Función Número Entero

Se declara una función booleana a la cual se le pasará un parámetro del tipo String, que será la cadena que el usuario ingrese en las cajas de texto de tiempo y costo. Se castea el parámetro tipo cadena a int y se evalúa si número ingresado es mayo a cero (en caso de tiempo). Para el caso de costo, se evalúa que el número sea mayor a cero. Se retornará verdadero si la condición se cumple, de lo contrario se retornará un falso.

```
public boolean esNumTiempo(String cadena){
        int cadenaNum=0;
        cadenaNum=Integer.parseInt( s:cadena);
        if(cadenaNum>4){
            return true;
        }else{
            return false;
        }
}
public boolean esNumEntero(String cadena){
        int cadenaNum=0;
        cadenaNum=Integer.parseInt( s:cadena);
        if(cadenaNum>0){
            return true;
        }else{
            return false;
        }
}
```

4. Función Incremento Y Decremento

Se declara una función estática con la palabra reservada sychronized para que todas las funciones trabajen de forma sincronizada. Si las funciones son incremento, el contador se incrementará de 1 en 1. Si las funciones son decremento, el contador se decrementará de 1 en 1.

```
public static synchronized void incInventario(){
   ContadorInventario++;
}
public static synchronized void decInventario(){
    ContadorInventario--;
}
public static synchronized void incProduccion(){
   ContadorProduccion++;
}
public static synchronized void decProduccion(){
    ContadorProduccion--;
}
public static synchronized void incEmpaquetado(){
   ContadorEmpaquetado++;
}
public static synchronized void decEmpaquetado(){
    ContadorEmpaquetado--;
}
public static synchronized void incSalida(){
   ContadorSalida++;
}
public static synchronized void decSalida(){
    ContadorSalida--;
}
public static synchronized void incFinal(){
   ContadorFinal++;
}
```

5. Función Pintar y Despintar Panel

Se declara una clase para pintar y despintar, se le agrega un constructor a la clase con los parámetros de Graphics, x, y. En el constructor se crear un nuevo color para pintar el circulo, se agrega el color y se crear un circulo de 20x20. Para la función despintar, la diferencia es que el circulo será del color del panel en donde se estén creando los círculos. Se repite esto para todas las estaciones (ya que se utiliza un panel por estación).

```
public class PintarCiculoInventario{
    public PintarCiculoInventario(Graphics g, int x, int y) {
        Color Inventario = new Color( r:110,  g:77,  b:252);
        g.setColor( c:Inventario);
        g.fillOval(x, y,  width:20,  height:20);
    }
}
public class DespintarCiculo{
    public DespintarCiculo(Graphics g, int x, int y) {
        g.setColor( c:Color.WHITE);
        g.fillOval(x, y,  width:20,  height:20);
    }
}
```

6. Funciones Hilos

Se crea una clase como implementación ejecutable para declarar que es un hilo. Se crea un constructor para la clase y se le pasan los parámetros de contadores y paneles. Se sobrescribe el método run y en ducho método se duerme un hilo por el tiempo establecido por el usuario. Se valida si la posición x del panel es igual o mayor a 300 (largo del panel), si la condición es verdadera se incrementa la posición y en 30 y la posición x volverá a ser 6. Se llamara a la función incrementar contador para incrementar el contador del hilo, se crea un objeto del tipo pintar para pintar el circulo, al cual se le pasan los parámetros de panel, posición x, posición y. Se repite lo mismo con la función decrementar y el objeto despintar. Por último se valida si los contadores anteriores son cero, si la condición se cumple, el hilo se detendrá.

```
public class HiloProduccion implements Runnable{
    public boolean ValidarHilo2=true;
    JLabel contadorInventarioLBL, contadorProduccionLBL; JPanel Panel1, Panel2;
    public HiloProduccion(JLabel contadorInventarioLBL, JLabel contadorProduccionLBL, JPanel Panel1, JPanel Panel2){
        this.contadorInventarioLBL = contadorInventarioLBL;
        this.contadorProduccionLBL = contadorProduccionLBL;
        this.Panel1=Panel1;
        this.Panel2=Panel2;
    }
    @Override
    public void run() {
        int x=6; int y=40;
        //Tiempo Para Pasar de Inventario a Produccion
        while(ValidarHilo2){
            try {
                Thread.sleep(TiempoProduccion*1000);
            } catch (InterruptedException ex) {
                Logger.getLogger( name:SimulacionJFrame.class.getName()).log( level:Level.SEVERE,  msg:null,  thrown:ex);
            }
            System.out.println( x:"Trabaja Hilo Produccion");
            if(x>=300){y=y+30;x=6;}
            incProduccion();
            PintarCiculoProduccion CirculoProduccion = new PintarCiculoProduccion( g:Panel1.getGraphics(),x,y);
            contadorProduccionLBL.setText( text:String.valueOf( i:ContadorProduccion));
            DespintarCiculo Circulo = new DespintarCiculo( g:Panel2.getGraphics(),x,y);
            decInventario();
            contadorInventarioLBL.setText( text:String.valueOf( i:ContadorInventario));
            if(ContadorInventario==0&&ContadorIncial==0){
                ValidarHilo2=false;
                HiloProduccion.stop();
            }
            x=x+50;
        }
    }
}
```

7. Función Generar Código Aleatorio

Esta función genera una cadena de caracteres de forma aleatoria. Para realizar esto se deben declarar una constante del tipo String en estarán todas las posibles letras que el código puede tomar y una constante del tipo int que será igualada a la cantidad de letras que el código tomará de la constante declarada anteriormente.

Se declara una variable del tipo StringBuilder para almacenar los caracteres, con un ciclo for, se realiza un recorrido con la cantidad de letras que el código puede tomar, se declara una variable random para obtener una letra aleatoria, se obtiene la posición del carácter, se convierte dicha posición a una variable carácter y finalmente se agrega el carácter a la variable del tipo StringBuilder.

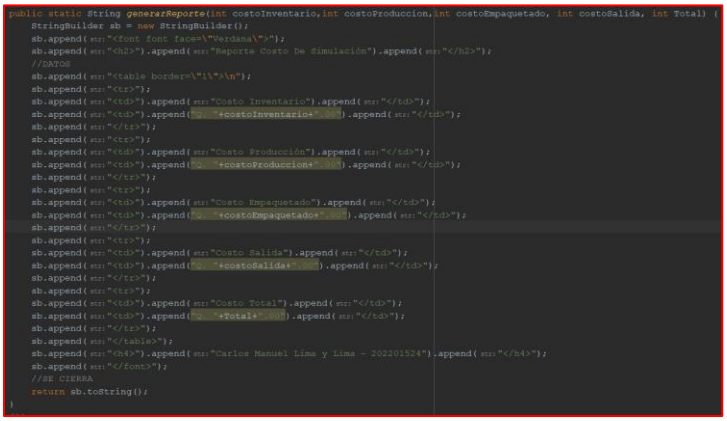
Par los números, se crea una variable del tipo int para almacenar los números, se declara una viable random para obtener números aleatorios del 0 hasta el límite que se declara.

Por último, se concatena todas las variables obtenidas para generar el código de forma aleatorio en una variable del tipo String y se retorna dicha variable.

```
public static String generarCodigo(){
    //LETRAS ALEATORIAS
    final String cadena="QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm";
    final int longi=3;
    StringBuilder sb1 = new StringBuilder();
    for (int i = 0;i<longi; i++) {
        double aleatorio1=Math.random()*cadena.length();
        int posicion1=(int)aleatorio1;
        char letra1 =cadena.charAt( index: posicion1);
        sb1.append( c: letra1);
    }
    //NUMEROS ALEATORIOS
    int aleatorio1=0;
    Random codigoRandom = new Random();
    aleatorio1=(int)(codigoRandom.nextDouble()*10);
    //CODIGO
    String codigo="Reporte_"+sb1.toString()+aleatorio1;
    //RETORNO
    return codigo;
}
```

8. Función Generar Reporte HTML

Esta función genera un archivo HTML con el contenido que se desea. Para realizar esto, se declara una variable del tipo StringBuilder con la cual, por medio de la función append, se agregará todas las etiquetas de HTML que se desean. También se podrán agregar variables de cualquier tipo con forme se necesitan. Para generar el archivo, primero se declaran las variables con su respectivo contenido, se llama a la función creada anteriormente, se crea un variable del tipo archivo en donde se almacenará la dirección del nuevo archivo, por último se creará un nuevo archivo con la extensión .html y se mostrará un mensaje que el archivo fue creado exitosamente.



```
public static String generarReporte(int costoInventario,int costoProduccion,int costoEmpaquetado, int costoSalida, int Total) {
    StringBuilder sb = new StringBuilder();
    sb.append( str: "<font font face=\"Verdana\">");
    sb.append( str: "<h2>").append( str: "Reporte Costo De Simulación").append( str: "</h2>");
    //DATOS
    sb.append( str: "<table border=\"1\">\n");
    sb.append( str: "<tr>");
    sb.append( str: "<td>").append( str: "Costo Inventario").append( str: "</td>");
    sb.append( str: "<td>").append("Q. "+costoInventario+".00").append( str: "</td>");
    sb.append( str: "</tr>");
    sb.append( str: "<tr>");
    sb.append( str: "<td>").append( str: "Costo Producción").append( str: "</td>");
    sb.append( str: "<td>").append("Q. "+costoProduccion+".00").append( str: "</td>");
    sb.append( str: "</tr>");
    sb.append( str: "<tr>");
    sb.append( str: "<td>").append( str: "Costo Empaquetado").append( str: "</td>");
    sb.append( str: "<td>").append("Q. "+costoEmpaquetado+".00").append( str: "</td>");
    sb.append( str: "</tr>");
    sb.append( str: "<tr>");
    sb.append( str: "<td>").append( str: "Costo Salida").append( str: "</td>");
    sb.append( str: "<td>").append("Q. "+costoSalida+".00").append( str: "</td>");
    sb.append( str: "</tr>");
    sb.append( str: "<tr>");
    sb.append( str: "<td>").append( str: "Costo Total").append( str: "</td>");
    sb.append( str: "<td>").append("Q. "+Total+".00").append( str: "</td>");
    sb.append( str: "</tr>");
    sb.append( str: "</table>");
    sb.append( str: "<h4>").append( str: "Carlos Manuel Lima y Lima - 202201524").append( str: "</h4>");
    sb.append( str: "</font>");
    //SE CIERRA
    return sb.toString();
}
```