Documentación

Paradero autobús.

Integrantes:

Anguiano Morales Benjamin

Martin Mancilla Angel Omar

Planteamiento.

En los paraderos de autobuses suele haber conflictos al momento de abordar un autobús (excepto cuando hay una persona encargada de mediar el control en ese paradero), por un lado están los autobuses que intentan subir gente antes que otro que ya esté formado (lo cual es injusto), y por otro lado, también encontramos las personas que intentan subir antes que otras al autobús en cuestión.

Posibles soluciones.

Estas dos circunstancias pueden ser controladas en el programa. ¿Cómo? Podemos usar un multiplex para dejar subir a n cantidad de personas a cierto autobús y, por parte de los autobuses ocurriría algo similar, atendiendo los que puedan estar en la base y los que no, esperarse.

Debido a que la anterior solución no me controla el que algún autobús o persona acceda antes que otro, he optado por usar un mutex de 1 para ambos casos (o su simil).

Solución implementada.

Controlaremos la concurrencia en autobuses y personas dejándolos saber quién está por delante de la fila y así pueda esperar (por respeto) a que el autobús se vaya o termine su ejecución (caso autobús esperando), o que la persona suba al autobús o termine su ejecución (caso persona esperando).

El método con el que cuentan los hilos y que me puede servir como mutex es el método join(). El método join me permite esperar a que un proceso termine para continuar con la siguiente parte del código. Esto pasaría en ambos casos, autobús o persona.

Por si algo saliera mal, también creé dos mutex, llamados “semaforoAutobus” y “semaforoPersonas” que permiten acceder a la zona donde deben ser despachados. Se entiende que bastaría con el método join(), pero queremos evitar problemas.

LENGUAJE USADO: JAVA versión 7 o superior.

PROBADO EN LA VERSIÓN 7 O SUPERIOR.

Tanto Debian (LINEA DE COMANDOS), como en ide eclipse en Windows 10.

Librerías usadas:

java.util.ArrayList;

java.util.Random;

java.util.concurrent.Semaphore;

javax.swing.*;

javax.swing.JPanel;

java.awt.*;

java.io.File;

java.io.IOException;

javax.imageio.*;

javax.swing.JPanel;

Cómo ejecutar el programa en debian:

Desde línea de comandos, buscamos la carpeta donde se encuentra el "Manejador1.java" junto con las imágenes "persona.jpg", "persona2.jpg", "persona3.jpg" y "autobús.jpg" y nos colocamos en esa carpeta.

Compilamos usando: **javac Manejador1.java**

Ejecutamos usando: **java Manejador1**

Consideraciones al ejecutar.

Cuando se ejecuta, en línea de comandos se muestra lo que va ocurriendo en el programa. En la ventana se va mostrando una persona junto al bus cuando está

subiendo al autobús, se muestra el autobús solo cuando ya está lleno y se muestra el autobús desplazado a la izquierda cuando ya se está yendo.

Pequeño problema extraño:

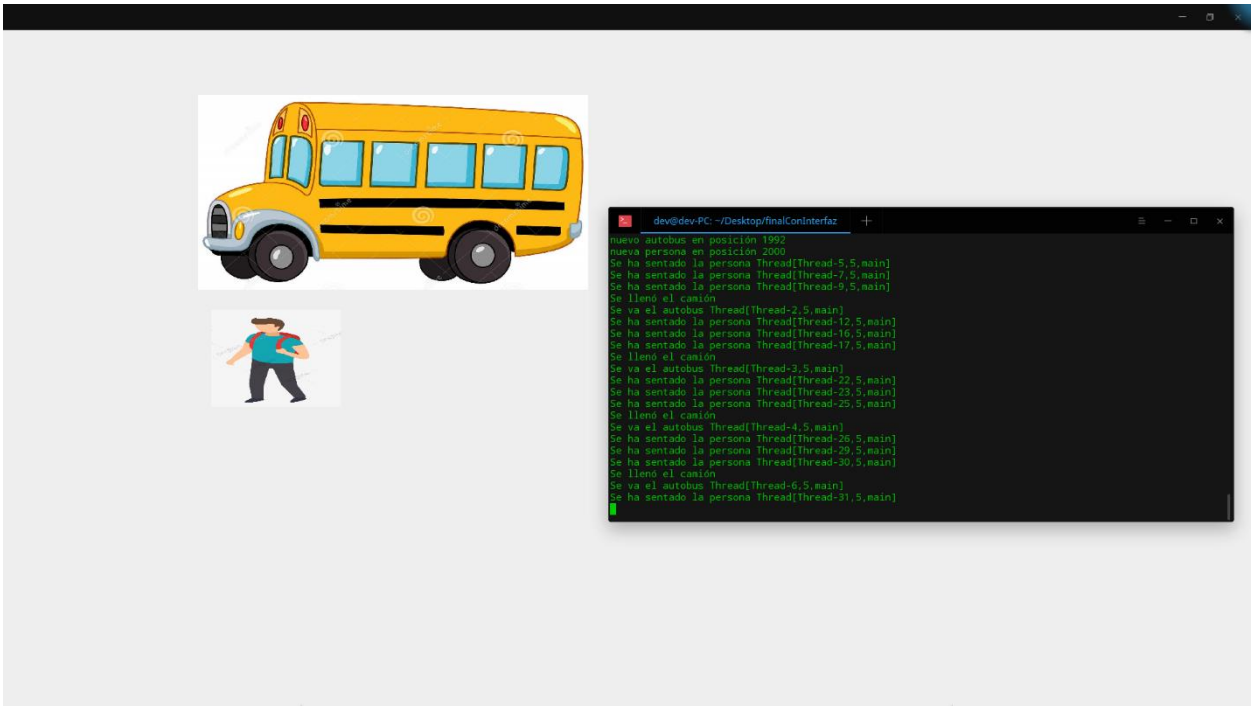
La ventana no muestra las imágenes en un principio, es necesario redimensionar la ventana repetidas veces (que no maximizar o minimizar) para que se muestren. En caso de que no se muestren algunas imágenes, al redimensionar aparecen. Una vez aparezca cada una de las **3** imágenes posibles (autobús con persona, autobús solo y autobús desplazado), ya no es necesario redimensionar, ya se mostrarán con normalidad.

¿A qué se deberá?

Problema común que me he encontrado al pintar cosas sobre una ventana en java.

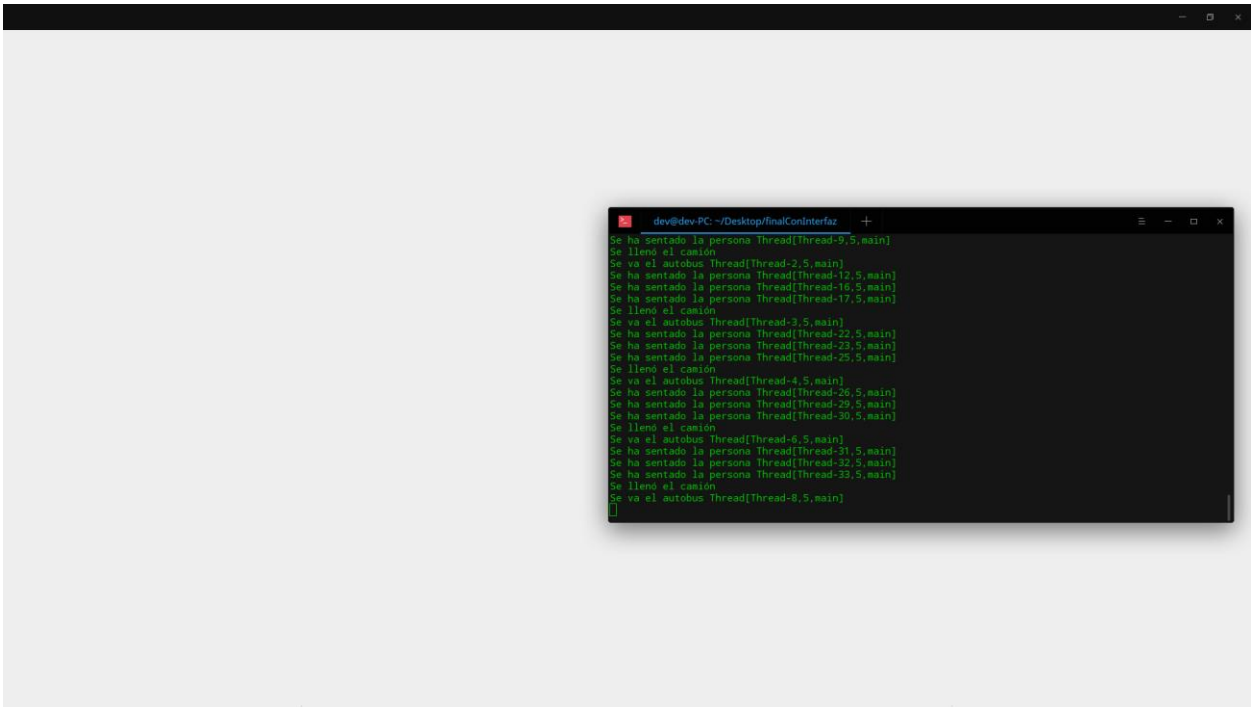
Posiblemente se deba al método Paint() y la forma en la que actualiza lo que hay en la ventana.

Cómo se ve cuando la ejecución es exitosa.



```
dev@dev-PC: ~/Desktop/finalConInterfaz
nueva persona en posición 2000
Se ha sentado la persona Thread[Thread-5,5,main]
Se ha sentado la persona Thread[Thread-7,5,main]
Se ha sentado la persona Thread[Thread-9,5,main]
Se llenó el camión
Se va el autobus Thread[Thread-2,5,main]
Se ha sentado la persona Thread[Thread-12,5,main]
Se ha sentado la persona Thread[Thread-16,5,main]
Se ha sentado la persona Thread[Thread-17,5,main]
Se llenó el camión
Se va el autobus Thread[Thread-3,5,main]
Se ha sentado la persona Thread[Thread-22,5,main]
Se ha sentado la persona Thread[Thread-23,5,main]
Se ha sentado la persona Thread[Thread-25,5,main]
Se llenó el camión
Se va el autobus Thread[Thread-4,5,main]
Se ha sentado la persona Thread[Thread-26,5,main]
Se ha sentado la persona Thread[Thread-29,5,main]
Se ha sentado la persona Thread[Thread-30,5,main]
Se llenó el camión
Se va el autobus Thread[Thread-6,5,main]
Se ha sentado la persona Thread[Thread-31,5,main]
Se ha sentado la persona Thread[Thread-32,5,main]
```

```
dev@dev-PC: ~/Desktop/finalConInterfaz
Se ha sentado la persona Thread[Thread-5,5,main]
Se ha sentado la persona Thread[Thread-7,5,main]
Se ha sentado la persona Thread[Thread-9,5,main]
Se llenó el camión
Se va el autobus Thread[Thread-2,5,main]
Se ha sentado la persona Thread[Thread-12,5,main]
Se ha sentado la persona Thread[Thread-16,5,main]
Se ha sentado la persona Thread[Thread-17,5,main]
Se llenó el camión
Se va el autobus Thread[Thread-3,5,main]
Se ha sentado la persona Thread[Thread-22,5,main]
Se ha sentado la persona Thread[Thread-23,5,main]
Se ha sentado la persona Thread[Thread-25,5,main]
Se llenó el camión
Se va el autobus Thread[Thread-4,5,main]
Se ha sentado la persona Thread[Thread-26,5,main]
Se ha sentado la persona Thread[Thread-29,5,main]
Se ha sentado la persona Thread[Thread-30,5,main]
Se llenó el camión
Se va el autobus Thread[Thread-6,5,main]
Se ha sentado la persona Thread[Thread-31,5,main]
Se ha sentado la persona Thread[Thread-32,5,main]
Se ha sentado la persona Thread[Thread-33,5,main]
```



Y se repite el proceso.