LABORATORIO #6

JULIAN CAMILO LOPEZ BARRERO

JUAN SEBASTIAN PUENTES JULIO

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

POOB

BOGOTÁ,COLOMBIA

7 DE MAYO DE 2025

PROGRAMACIÓN ORIENTADA A OBJETOS
Persistencia
2025-01
Laboratorio 6/6 [:)]
OBJETIVOS
1. Completar el código de un proyecto considerando requisitos funcionales.
2. Diseñar y construir los métodos básicos de manejo de archivos: abrir, guardar, importar y exportar.
3. Controlar las excepciones generadas al trabajar con archivos.
4. Experimentar las prácticas XP : Only one pair integrates code at a time.
Use collective ownership.

ENTREGA
Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios preparados para tal fin.

DESARROLLO
Preparando
En este laboratorio vamos a extender el proyecto schelling adicionando un menú barra con las opciones básicas de entrada-salida y las opciones estándar nuevo y salir.
1. En su directorio descarguen la versión del proyecto realizado por ustedes para el laboratorio 03 y preparen el ambiente para trabajar desde CONSOLA
2. Ejecuten el programa, revisen la funcionalidad.

Creando la maqueta

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

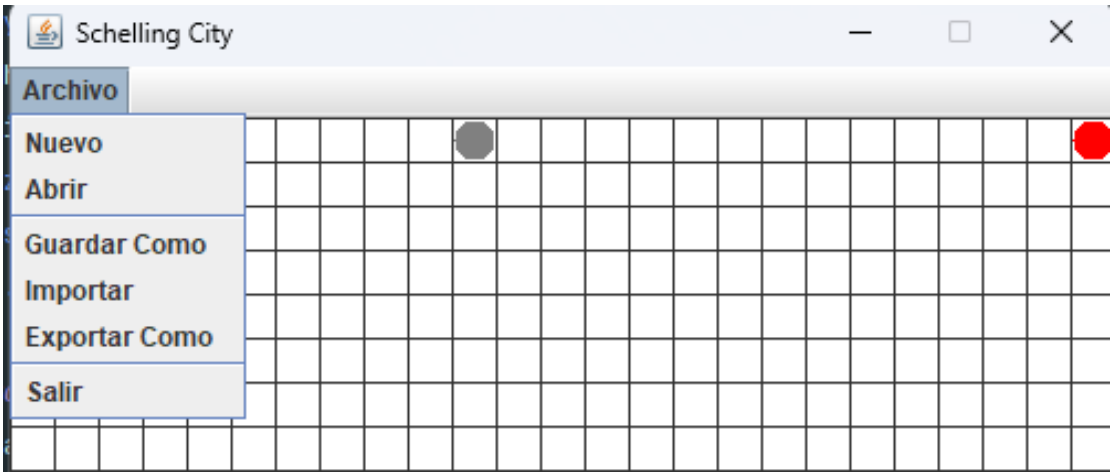
En este punto vamos a construir la maqueta correspondiente a esta extensión siguiendo el patrón MVC.

1. **MODELO**: Preparen en la clase fachada del dominio los métodos correspondientes a las cuatro opciones básicas de entrada-salida (open, save, import y export). Los métodos deben simplemente propagar una CityException con el mensaje de "Opción nombreOpción en construcción. Archivo nombreArchivo". Los métodos deben tener un parámetro File.

```
//----------------------NORMAL----------------------
public City open(File file) throws CityException {  1 usage
    throw new CityException("Opción open en construcción. Archivo " + file.getName());
}

public void save(File file) throws CityException {  1 usage
    throw new CityException("Opción save en construcción. Archivo " + file.getName());
}

public void importFile(File file) throws CityException {  no usages
    throw new CityException("Opción import en construcción. Archivo " + file.getName());
}

public void export(File file) throws CityException {  no usages
    throw new CityException("Opción export en construcción. Archivo " + file.getName());
}
```

2. **VISTA :** Construyan un menú barra que ofrezca, además de las opciones básicas de entrada-salida, las opciones estándar de nuevo y salir (Nuevo, Abrir, Guardar como, Importar, Exportar como, Salir). No olviden incluir los separadores. Para esto creen el método prepareElementsMenu. Capturen la pantalla del menú.

```
private void prepareElementsMenu(){  1 usage
    menuBar = new JMenuBar();
    menuPrincipal = new JMenu( s: "Archivo");
    //Opciones
    nuevo = new JMenuItem( text: "Nuevo");
    abrir =  new JMenuItem( text: "Abrir");
    guardarComo = new JMenuItem( text: "Guardar Como");
    importar = new JMenuItem( text: "Importar");
    exportarComo = new JMenuItem( text: "Exportar Como");
    salir = new JMenuItem( text: "Salir");
    //Añadirlas
    menuPrincipal.add(nuevo);
    menuPrincipal.add(abrir);
    menuPrincipal.addSeparator();
    menuPrincipal.add(guardarComo);
    menuPrincipal.add(importar);
    menuPrincipal.add(exportarComo);
    menuPrincipal.addSeparator();
    menuPrincipal.add(salir);
    menuBar.add(menuPrincipal);
    //Set Del Menu
    setJMenuBar(menuBar);
}
```



3. **CONTROLADOR:** Construyan los oyentes correspondientes a las seis opciones. Para esto creen el método prepareActionsMenu y los métodos base del controlador (optionOpen, optionSave, optionImport, optionExport, optionNew, optionExit),
En las opciones que lo requieran usen un FileChooser y atiendan la excepción. Estos métodos llaman el método correspondiente de la capa de dominio que por ahora sólo lanza una excepción. Ejecuten las diferentes acciones del menú y para cada una de ellas capture una pantalla significativa.

```
abrir.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {optionOpen();}
});
```

```
guardarComo.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {optionSave();}
});
```

```
importar.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) { optionImport(); }
});
```

```
exportarComo.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) { optionExport(); }
});
```

```
private void prepareActionsMenu(){  1 usage
    nuevo.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) { optionNew(); }
    });
```

```
salir.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) { optionExit(); }
});
```

Implementando salir y nuevo

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

Las opciones salir y nuevo van a ofrecer los dos servicios estándar de las aplicaciones. El primero no requiere ir a capa de dominio y el segundo sí.

1. Construyan el método optionExit que hace que se termine la aplicación. No es necesario incluir confirmación.

```
//Exit Action Menu
private void optionExit(){  1 usage
    System.exit( status: 0);
}
```

2. Construyan el método optionNew que crea una nueva ciudad. Capturen una pantalla significativa.

```
private void optionNew(){  1 usage
    theCity = new City();
    photo.repaint();

}
```

Implementando salvar y abrir

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

Las opciones salvar y abrir van a ofrecer servicios de persistencia de la ciudad como objeto. Los nombres de los archivos deben tener como extensión .dat.

1. Copien las versiones actuales de open y save y renómbrenlos como open00 y save00
2. Construyan el método save que ofrece el servicio de guardar en un archivo el estado actual de la ciudad. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.

```
//Save Con Parametro Archivo
public void save00(File file) throws CityException{  no usages
    try {
        ObjectOutputStream out = new ObjectOutputStream((new FileOutputStream(file)));
        out.writeObject(this);
        out.close();
    } catch (IOException e) {
        System.out.println("Error al guardar el archivo");
    }
}
```

```
✓ shouldSave00CityToFile()  11 ms
✓ shouldOpen00CityFromFile()  12 ms
```

3. Validen este método guardando el estado obtenido después de dos clics como oneCity.dat. ¿El archivo se creó en el disco? ¿Cuánto espacio ocupa?

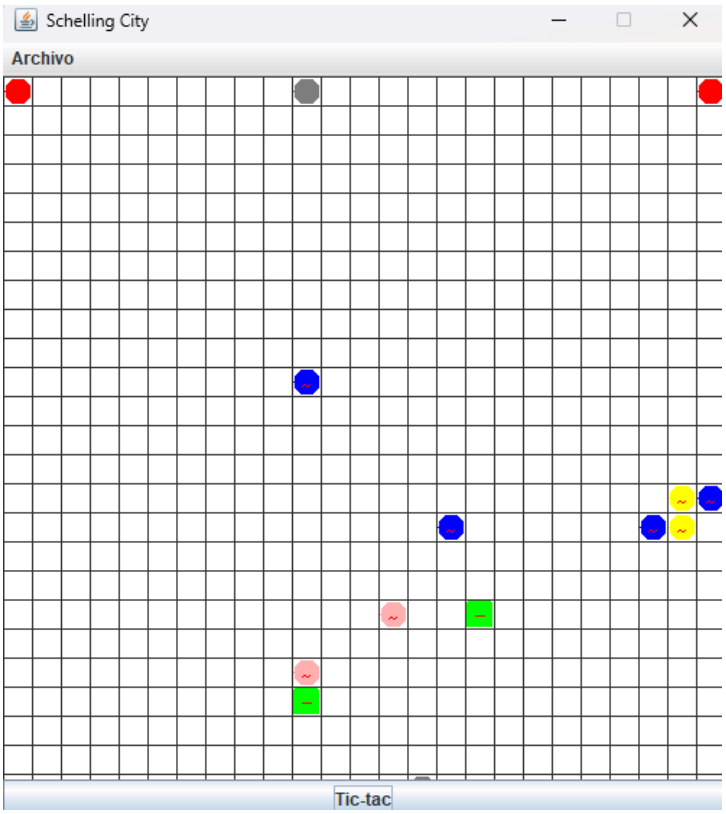
Si se crea en disco y ocupa 3KB.

4. Construyan el método open que ofrece el servicio de leer una ciudad de un archivo. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.

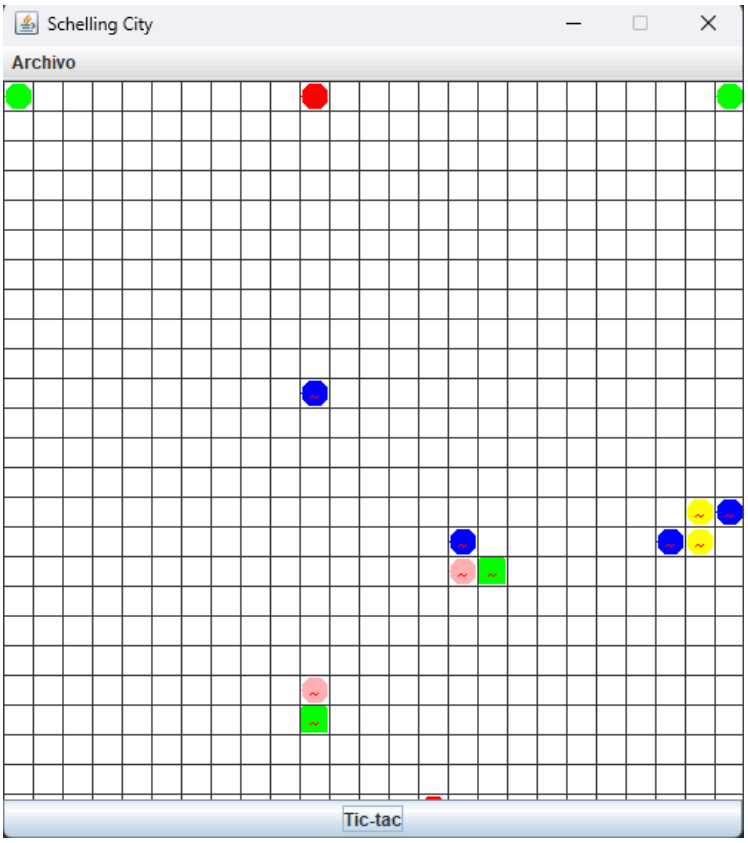
```java
//Open Con Parametro Archivo
public City open00(File file) throws CityException {  no usages
    if (!file.exists()) {
        System.out.println("No se encontro el archivo");
        return null;
    }
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(file))) {
        Object deserialize = in.readObject();
        if (deserialize instanceof City) {
            City newCity = (City) deserialize;
            return newCity;
        } else {
            System.out.println("Error al deserializar el archivo");
            return null;
        }
    } catch (IOException | ClassNotFoundException e) {
        throw new CityException("Opción open en construcción. Archivo " + file.getName());
    }
}
```

5. Realicen una prueba de aceptación para este método iniciando la aplicación, creando un nuevo estado y abriendo el archivo oneCity.dat. Capturen imágenes significativas de estos resultados.

CITY INICIAL SIN TIC-TACS:



ABRIENDO EL ARCHIVO oneCity.dat:



ES CORRECTO.

Implementando importar y exportar

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

Estas operaciones nos van a permitir importar información de la ciudad desde un archivo de texto y exportarla. Los nombres de los archivos de texto deben tener como extensión .txt

Los archivos texto tienen una línea de texto por cada elemento

En cada línea asociada a un elemento se especifica el tipo y la posición.

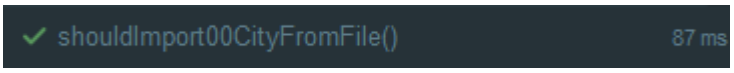
Person 10 10

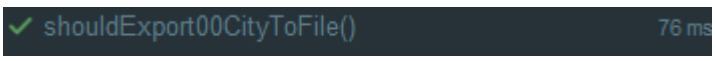
Walker 20 20

1. Copien las versiones actuales de import y export y renómbrenlos como import00 y export00.

2. Construyan el método export que ofrece el servicio de exportar a un archivo texto, con el formato definido, el estado actual. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.

```java
//Exportar Archivo 00
public void export00(File file) throws CityException {  1 usage
    try (FileWriter writer = new FileWriter(file)) {
        for (int r = 0; r < SIZE; r++) {
            for (int c = 0; c < SIZE; c++) {
                Item item = locations[r][c];
                if (item != null) {
                    String type = item.getClass().getSimpleName();
                    writer.write( str: type + " " + r + " " + c + "\n");
                }
            }
        }
    } catch (IOException e) {
        throw new CityException("No se pudo exportar el archivo: " + file.getName());
    }
}
```





3. Realicen una prueba de aceptación de este método: iniciando la aplicación y exportando como oneCity.txt. Editen el archivo y analicen los resultados. ¿Qué pasó?
Se visualizan los diferentes ítems con el formato indicado en el laboratorio en un archivo .txt.

```
Light 0 0
Identifier 0 10
Light 0 24
Person 10 10
Schelling 14 23
Person 14 24
Person 15 15
Person 15 22
Schelling 15 23
Katalan 17 14
Walker 17 16
Katalan 20 10
Walker 21 10
Identifier 24 14
```

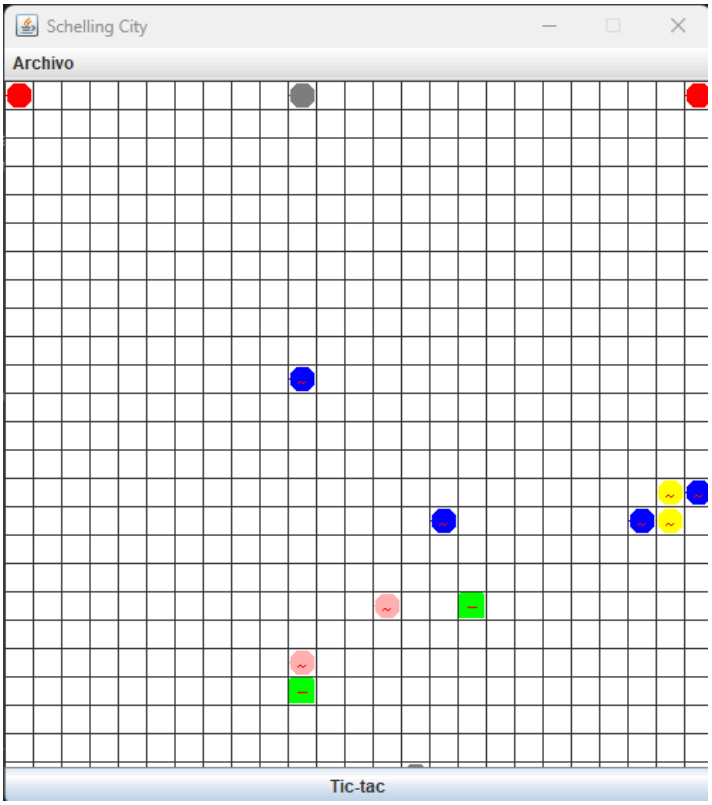
4. Construyan el método import que ofrece el servicio de importar de un archivo texto con el formato definido. Por ahora sólo considero un mensaje de error general. No olviden diseño y pruebas de unidad.

```java
//Importar Archivo 00
public void import00(File file) throws CityException {  1 usage
    try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.trim().split( regex: " ");
            if (parts.length != 3) continue;
            String type = parts[0];
            int row = Integer.parseInt(parts[1]);
            int col = Integer.parseInt(parts[2]);
            Item item = null;
            if ("Person".equals(type)) {
                item = new Person( city: this, row, col);
            } else if ("Walker".equals(type)) {
                item = new Walker( city: this, row, col);
            } else if ("Light".equals(type)) {
                item = new Light( city: this, row, col);
            } else if ("Katalan".equals(type)) {
                item = new Katalan( city: this, row, col);
            } else if ("Identifier".equals(type)) {
                item = new Identifier( city: this, row, col);
            } else if ("Schelling".equals(type)) {
                item = new Schelling( city: this, row, col);
            } else {
                System.out.println("Tipo desconocido: " + type);
            }
            if (item != null) {
                locations[row][col] = item;
            }
        }
    } catch (IOException | NumberFormatException e) {
        throw new CityException("Error al importar el archivo " + file.getName());
    }
}
```

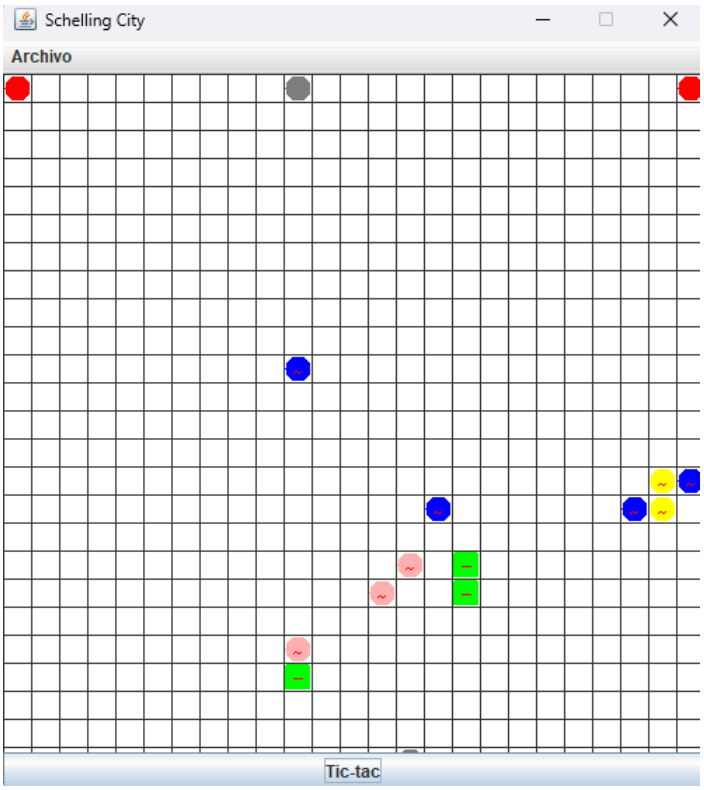
(Consulten en la clase String los métodos trim y split)

5. Realicen una prueba de aceptación de este par de métodos: iniciando la aplicación exportando a oneCity.txt. saliendo, entrando, creando una nueva e importando el archivo otherCity.txt. ¿Qué resultado obtuvieron? Capturen la pantalla final.

Inicialmente Se Ve Asi:



Importando, se visualiza:

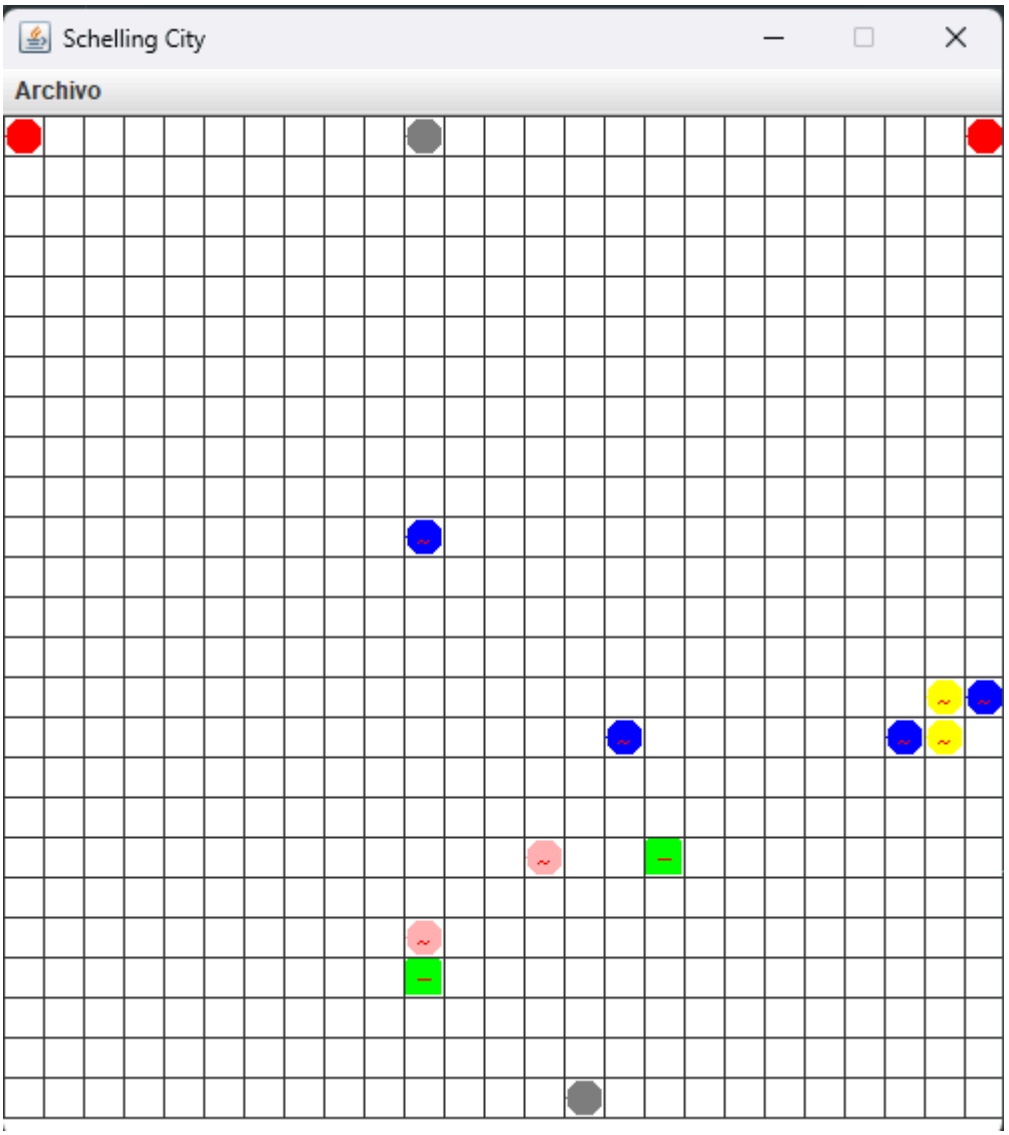


6. Realicen otra prueba de aceptación de este método escribiendo un archivo de texto correcto en oneCity.txt. e importe este archivo. ¿Qué resultado obtuvieron? Capturen la pantalla.

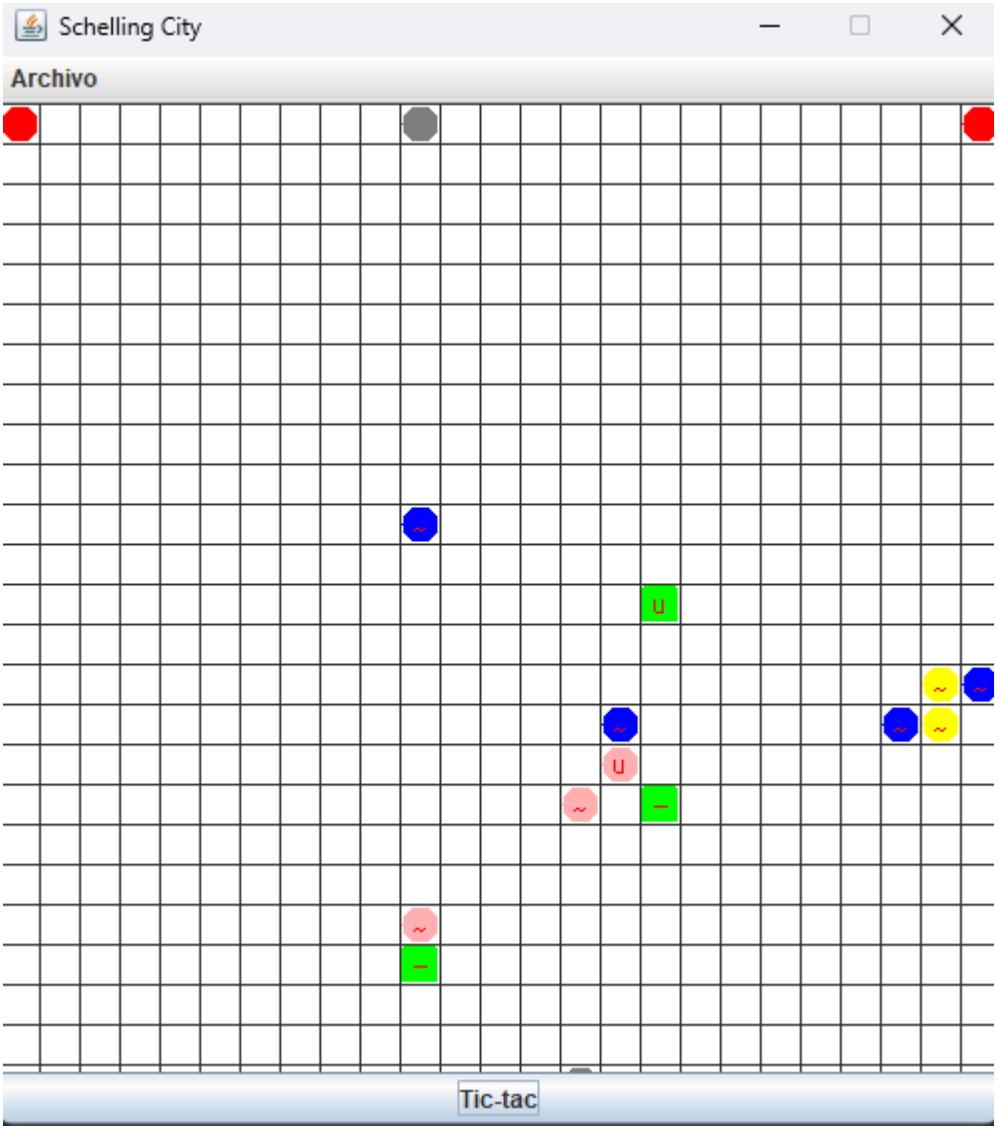
Archivo.txt para la prueba de aceptacion.

```
Light 0 15
Identifier 0 14
Light 0 22
Schelling 17 22
Katalan 19 14
Walker 11 15
Katalan 10 5
Walker 19 10
Identifier 10 14
```

INICIALMENTE:



IMPORTANDO EL ARCHIVO “oneCity.txt” con sus respectivos cambios , se ve tal que:



Analizando comportamiento

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

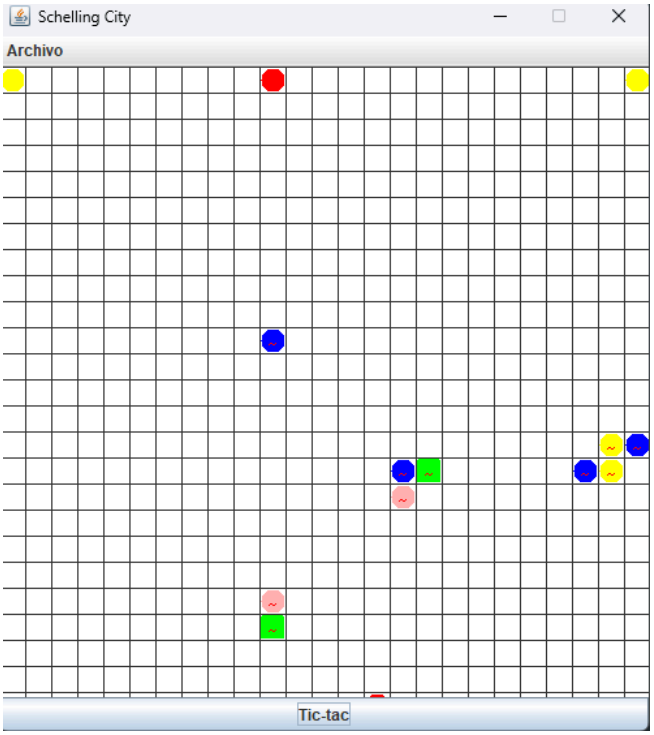
1. Ejecuten la aplicación, den tres clics, salven a un archivo cualquiera y ábranlo. Describan el comportamiento.

Cuando este se abre justamente luego de salvarlo, inmediatamente se actualiza el frame y se observan los diferentes items que se guardaron en determinado click en sus respectivas posiciones, sin embargo, su estado de ánimo inicial no concuerda con su estado de ánimo antes de salvar el archivo.

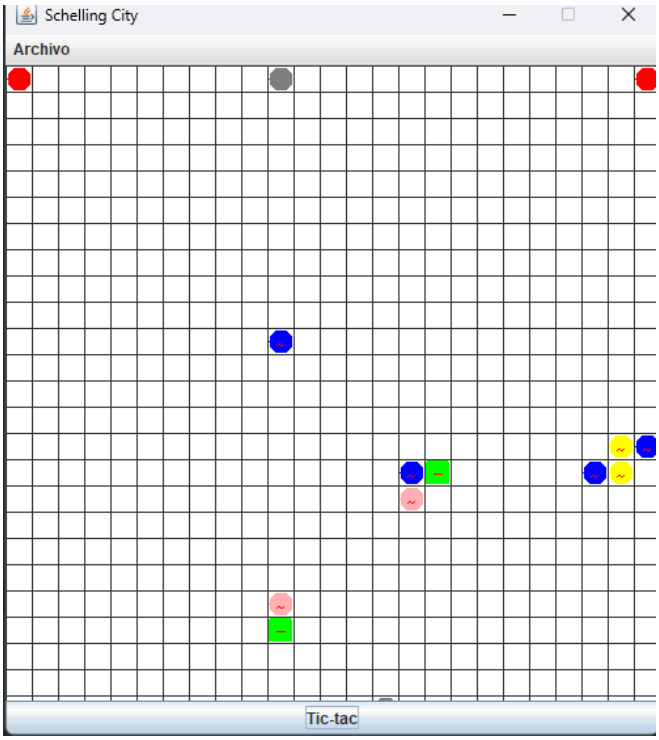
2. Ejecuten la aplicación, tres clics, exporten a un archivo cualquiera e importen. Describan el comportamiento

3. ¿Qué diferencias ven el comportamiento 1. y 2.? Expliquen los resultados.

Comportamiento 1:



Comportamiento 2:



Con respecto a las posiciones de los ítems en ambas ocasiones son iguales, sin embargo, para el segundo caso cuando importamos, no se mantiene el color final de los semáforos ya que se crean nuevas instancias de estos. Por otro lado, en el primer caso si se mantiene el color final del semáforo antes de ser salvado.

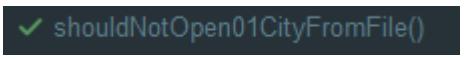
Perfeccionando salvar y abrir

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de open y save y renómbrenlos como open01 y save01
2. Perfeccionen el manejo de excepciones de los métodos open y save detallando los errores. No olviden pruebas de unidad.

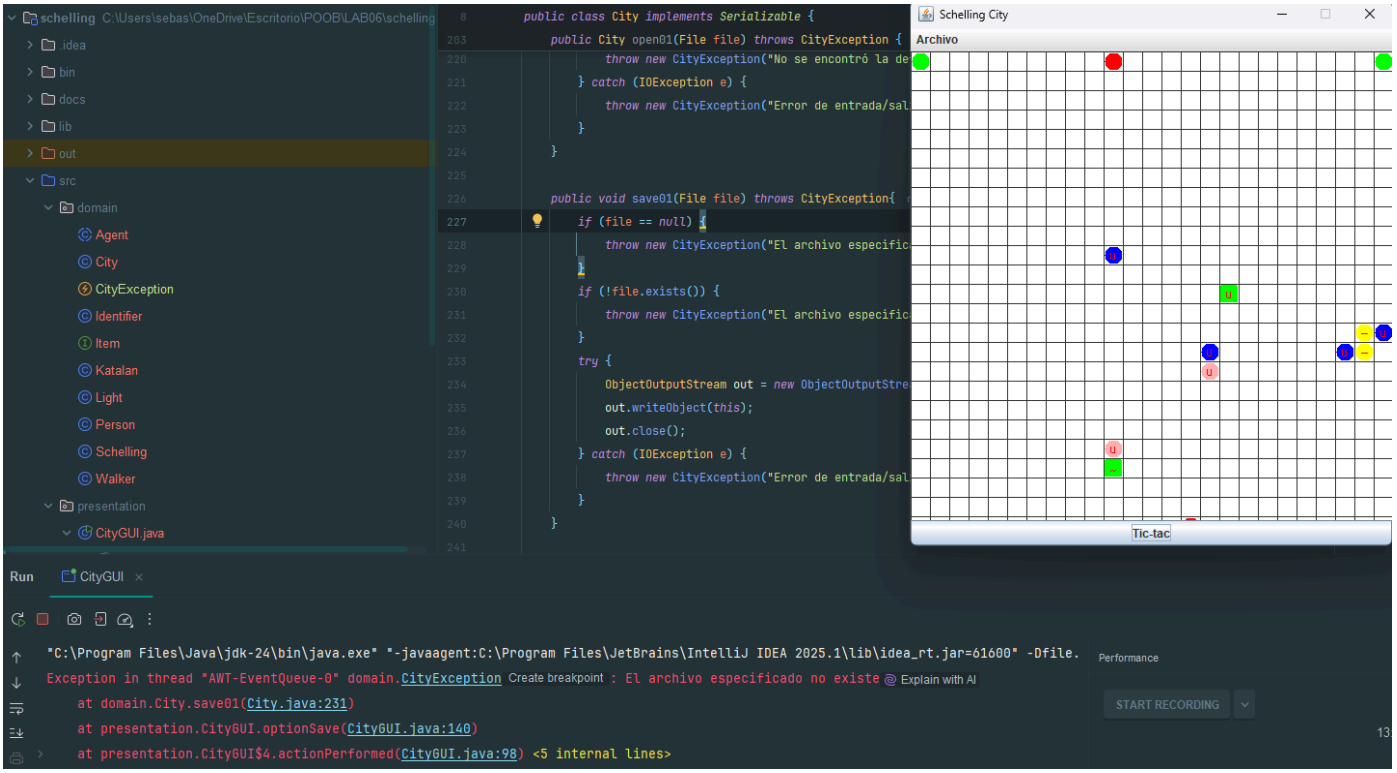
```java
public void save01(File file) throws CityException{  no usages
    if (file == null) {
        throw new CityException("El archivo especificado es nulo");
    }
    if (!file.exists()) {
        throw new CityException("El archivo especificado no existe");
    }
    try {
        ObjectOutputStream out = new ObjectOutputStream((new FileOutputStream(file)));
        out.writeObject(this);
        out.close();
    } catch (IOException e) {
        throw new CityException("Error de entrada/salida al escribir el archivo: " + file.getName());
    }
}
```

```java
//--------------------------------01----------------------
public City open01(File file) throws CityException {  no usages
    if (file == null) {
        throw new CityException("El archivo especificado es nulo");
    }
    if (!file.exists()) {
        throw new CityException("No se encontró el archivo: " + file.getAbsolutePath());
    }
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(file))) {
        Object deserialize = in.readObject();
        if (deserialize instanceof City) {
            return (City) deserialize;
        } else {
            throw new CityException("El archivo no contiene un objeto de tipo City válido");
        }
    } catch (FileNotFoundException e) {
        throw new CityException("No se pudo encontrar el archivo: " + file.getName());
    } catch (ClassNotFoundException e) {
        throw new CityException("No se encontró la definición de la clase al deserializar");
    } catch (IOException e) {
        throw new CityException("Error de entrada/salida al leer el archivo: " + file.getName());
    }
}
```

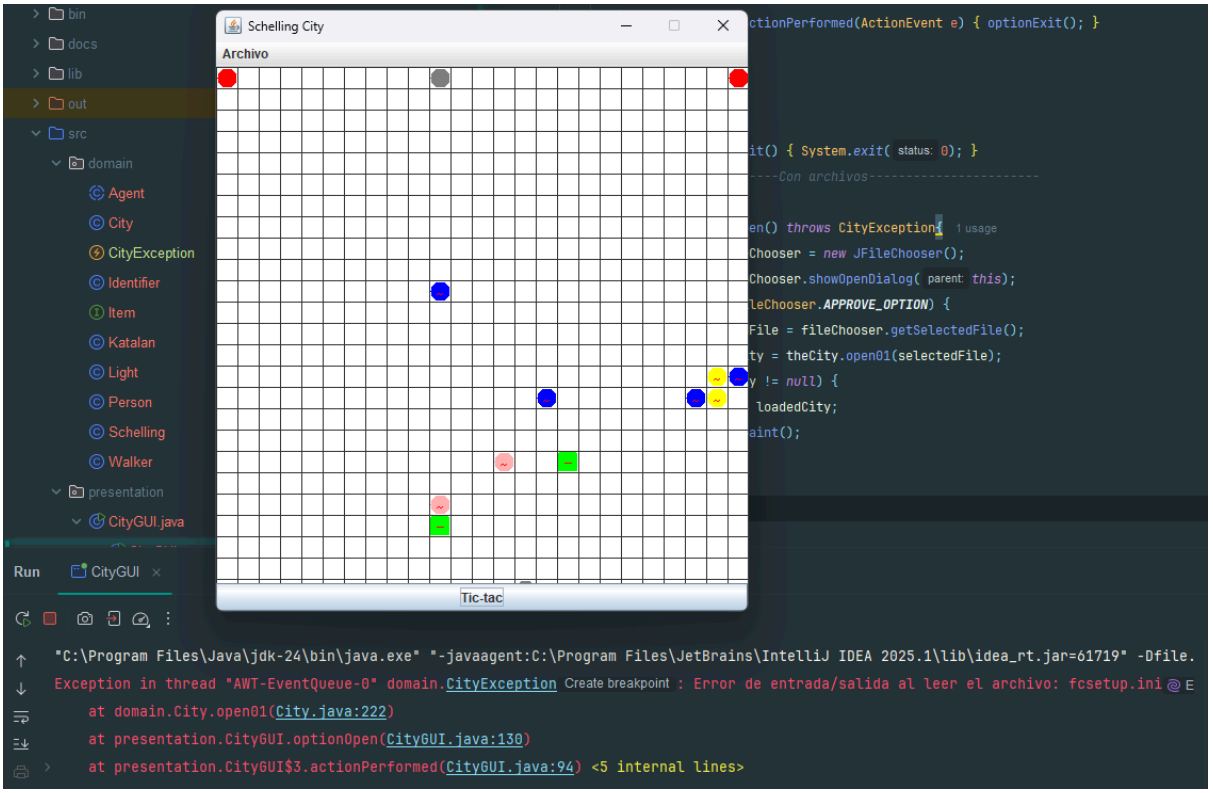


3. Realicen una prueba de aceptación para validar **UNO** de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.

PARA GUARDAR UN ARCHIVO SIN NOMBRE:



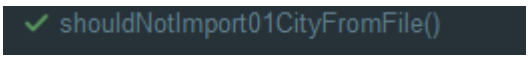
PARA ABRIR UN ARCHIVO QUE NO ES DE CITY:



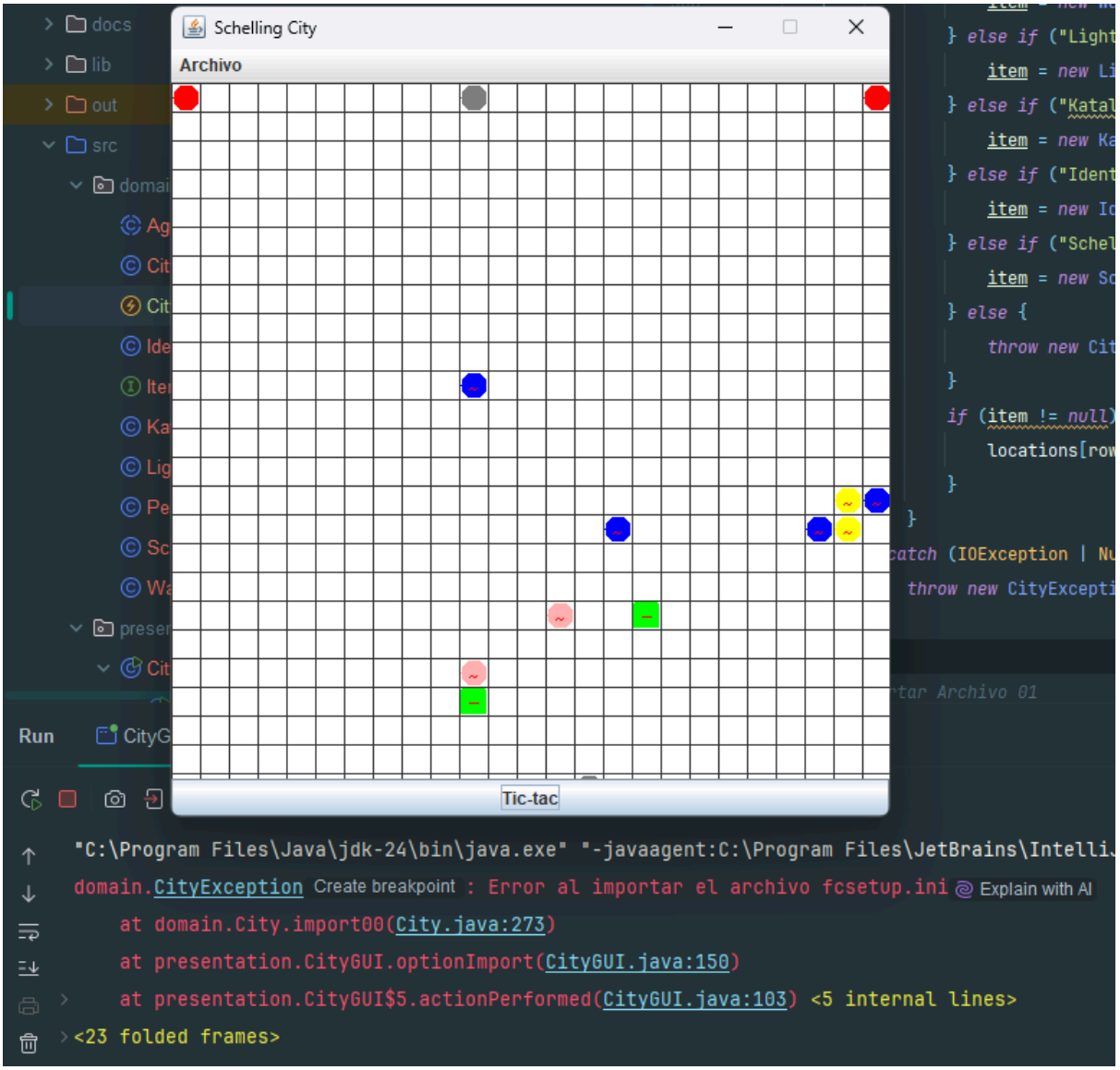
Perfeccionando importar y exportar.

[En lab06.doc, *.asta , cityErr.txt *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de import y export y renómbrenlos como import01 y export01
2. Perfeccionen el manejo de excepciones de los métodos import y export detallando los errores. No olviden pruebas de unidad.



3. Realicen una prueba de aceptación para validar uno de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.


Schelling City
Archivo
Tic-tac
Run
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Intelli
domain.CityException Create breakpoint : Error al importar el archivo fcsetup.ini Explain with AI
at domain.City.import00(City.java:273)
at presentation.CityGUI.optionImport(CityGUI.java:150)
at presentation.CityGUI$5.actionPerformed(CityGUI.java:103) <5 internal lines>
<23 folded frames>

Perfeccionando importar. Hacia un minicompilador.

[En lab06.doc, *.asta , cityErr.txt *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de import y export y renómbrenlos como import02 y export02

```
public void import02(File file) throws CityException {  no usages
```

```
public void export02(File file) throws CityException {  no usages
```

2. Perfeccionen el método import para que, además de los errores generales, en las excepciones indique el detalle de los errores encontrados en el archivo (como un compilador) : número de línea donde se encontró el error, palabra que tiene el error y causa de error.

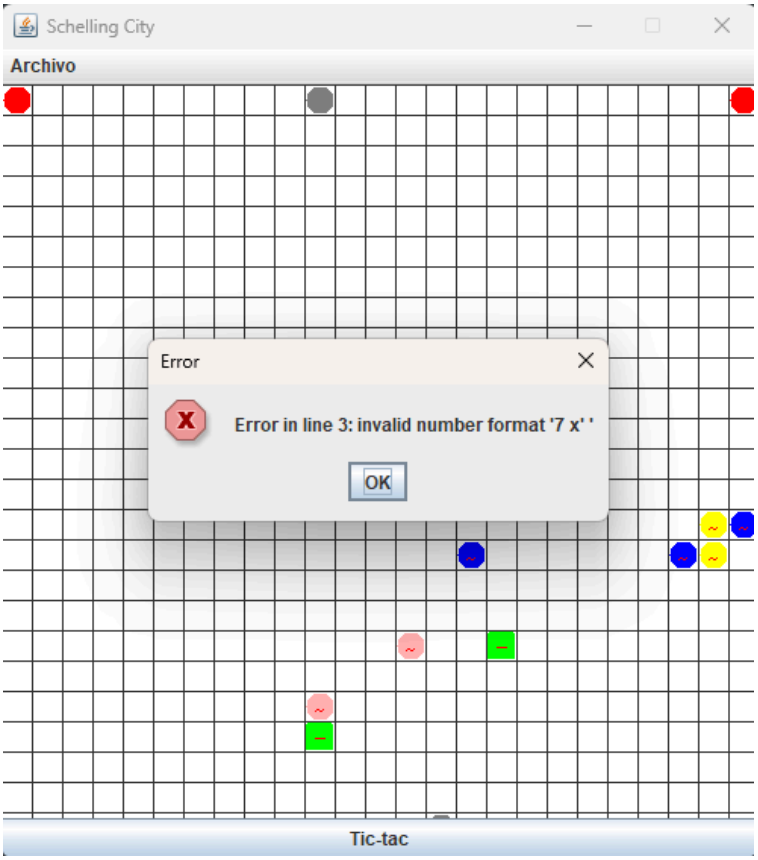
En City.java.

3. Escriban otro archivo con errores, llámelo cityErr.txt, para ir arreglándolo con ayuda de su "importador". Presente las pantallas que contengan los errores.

En cityErr.txt se escribe los siguiente:

```
Light 5 5
Person 6 6
Walker 7 x
Katalan x x
Michal 10 10
Identifier 8 8
```

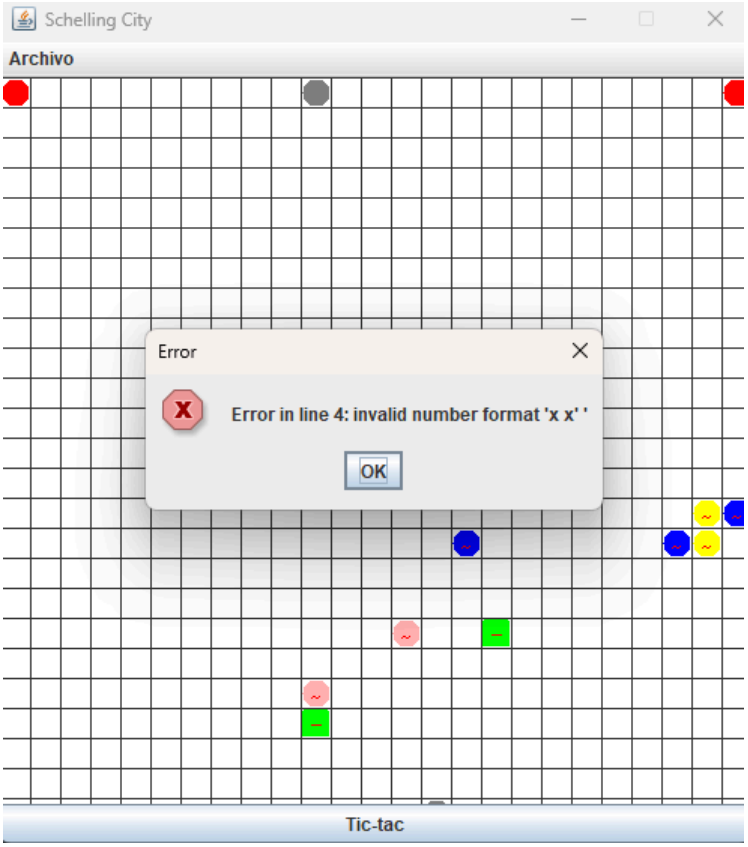
Intentando importar CityErr.txt:



Arreglando el archivo texto:

```
Light 5 5
Person 6 6
Walker 7 0
Katalan x x
Michal 10 10
Identifier 8 8
```

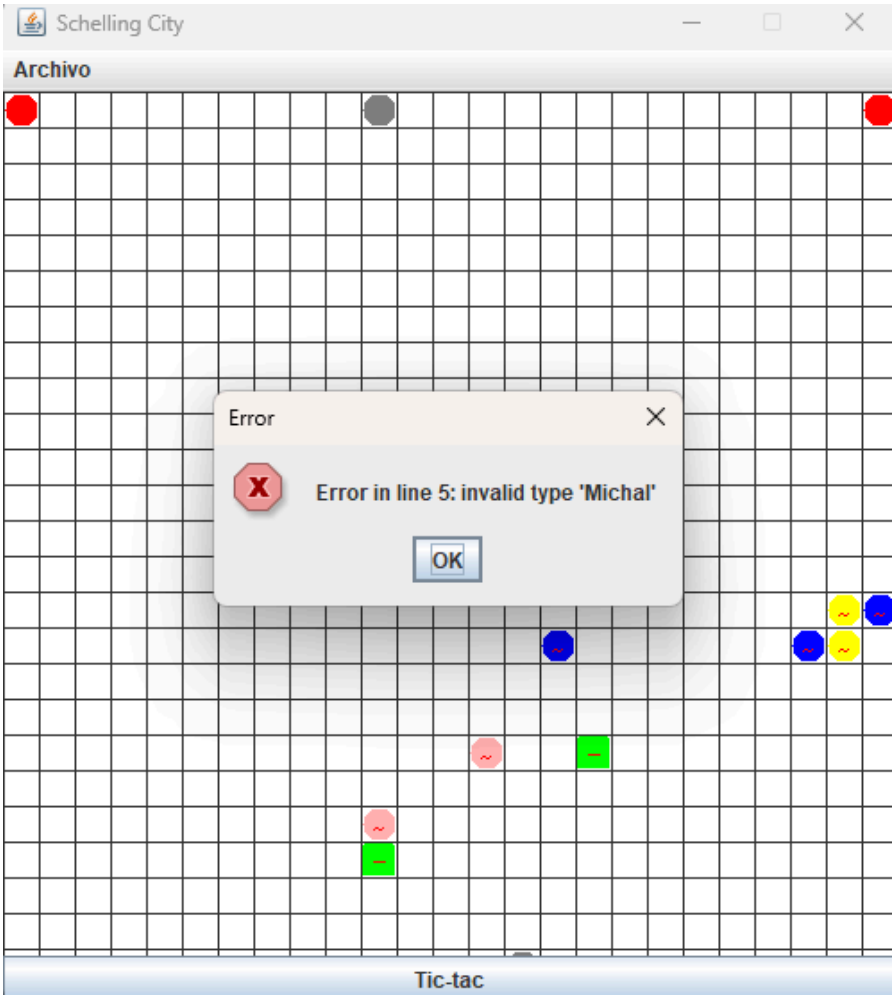
Importando nuevamente:



Arreglando el archivo texto:

```
Light 5 5
Person 6 6
Walker 7 0
Katalan 9 9
Michal 10 10
Identifier 8 8
```

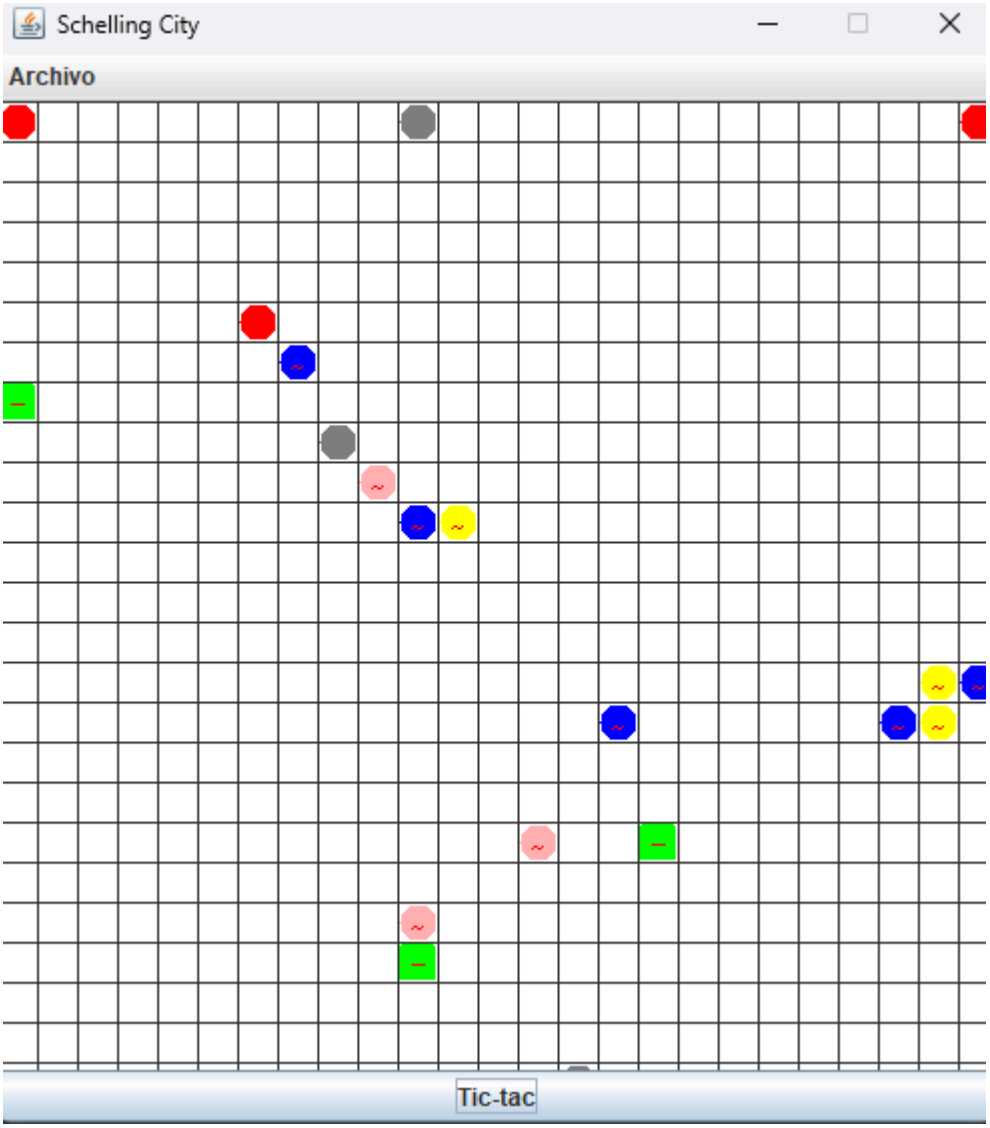
Importando nuevamente:



Arreglando el archivo texto:

```
Light 5 6
Person 6 7
Walker 7 0
Katalan 9 9
Schelling 10 10
Identifier 8 8
```

Finalmente se importa y se vería tal que:



BONO. XREFLEX(POSICIONES)

Perfeccionando importar. Hacia un minicompilador flexible.

[En lab06.doc, *.asta , schellingFlex.txt *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de import y export y renómbrenlos como import03 y export03

```
public void import03(File fileName) throws CityException
```

```
public void export03(File fileName) throws CityException {
```

2. Perfeccionen los métodos import y export para que pueda servir para cualquier tipo de elementos creados en el futuro. No olviden pruebas de unidad.

(Investiguen cómo crear un objeto de una clase dado su nombre)

En city.java.

3. Escriban otro archivo de pruebas, llámelo cityErrG.txt, para probar la flexibilidad. Presente las pantallas que contengan un error significativo.

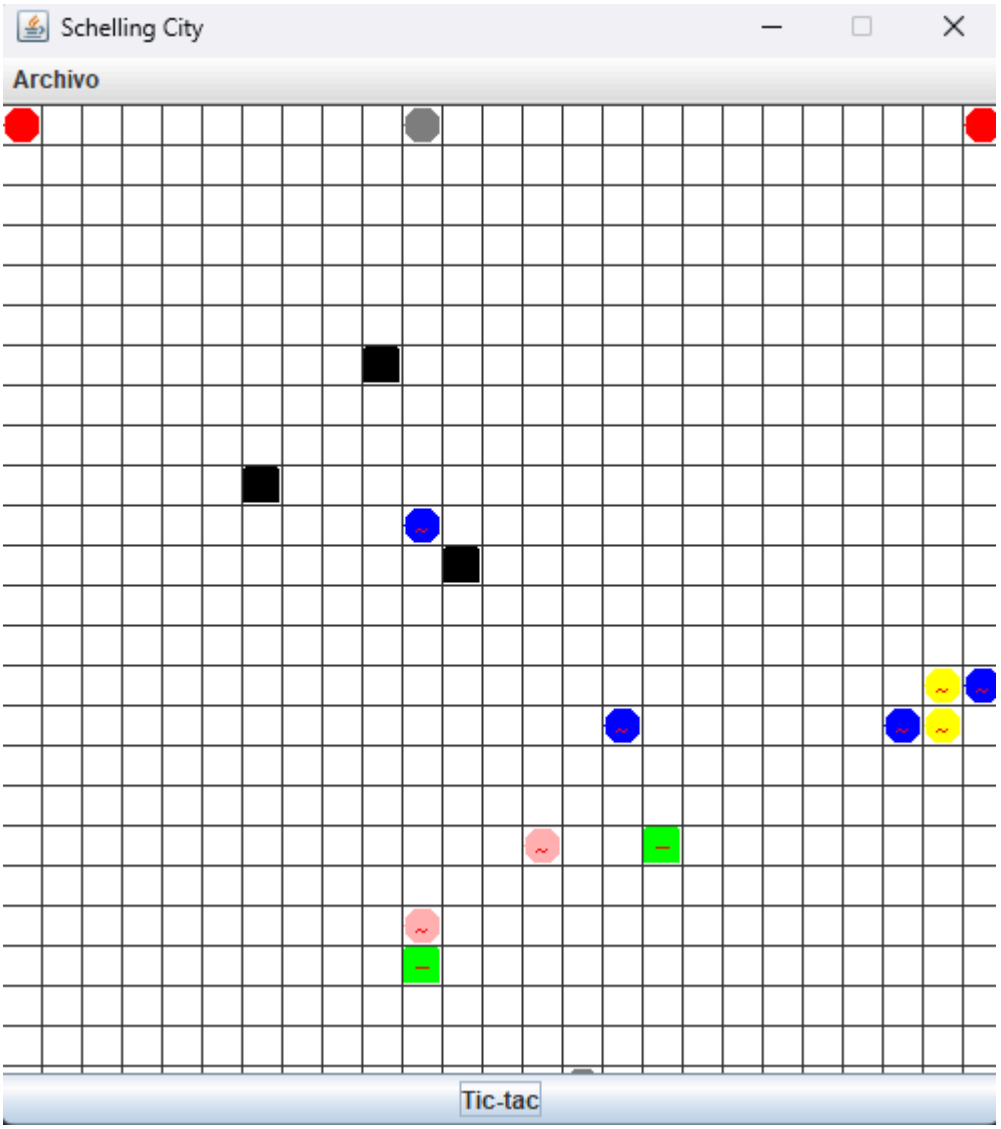
Para empezar, creamos un nuevo item llamado Block:

```java
package domain;

import java.awt.*;
import java.io.Serializable;

public class Block implements Item, Serializable {  no usages  new *

    private int row,column;  1 usage
    private City city;  2 usages
    private Color color;  2 usages

    public Block(City city, int row, int column){  no usages  new *
        this.city=city;
        this.row=row;
        this.column=column;
        this.city.setItem(row,column, e: this);
        color = Color.BLACK;
    }

    @Override  1 usage  new *
    public int shape() {
        return SQUARE;
    }

    public void decide(){  1 usage  new *
    }

    public Color getColor(){  new *
        return color;
    }
}
```

Creando un nuevo archivo con block:

```
Block 6 9
Block 9 6
Block 11 11
```

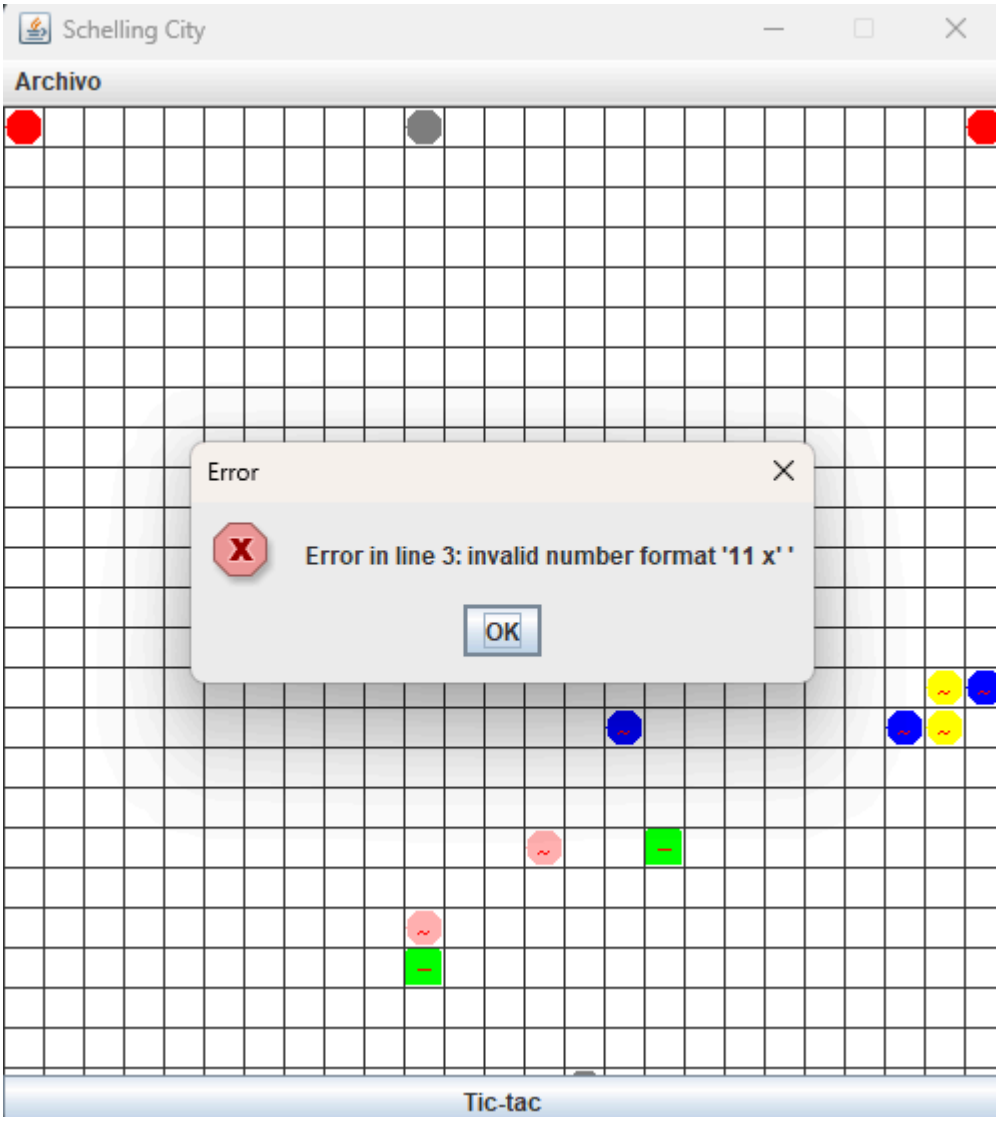
Importando este archivo se tal que:



Si a este archivo le ingresamos coordenadas inválidas, entonces;

```
Block 6 9
Block 9 6
Block 11 x
```

Se ve tal que:



Por lo que es correcto.

RETROSPECTIVA

1. **¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes ?(Horas/Hombre)**

+15 horas por cada uno.

2. **¿Cuál es el estado actual del laboratorio? ¿Por qué?**

El estado actual del laboratorio está completado ya que se realizó todo lo solicitado por el profesor.

3. **Considerando las prácticas XP del laboratorio. ¿Cuál fue la más útil? ¿por qué?**

Creemos útil pair programming ya que se corrigen los diferentes errores con mayor facilidad y así mismo cada uno es capaz de aprender de una mejor manera.

4. **¿Cuál consideran fue el mayor logro? ¿Por qué?**

Realizar la codificación del método import() por que en un principio este no podía leer los archivos txt que se exportaban.

5. **¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?**

Hacer la implementación de la función para crear instancias de una clase. Investigamos a profundidad cómo usar esta.

6. **¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?**

La comunicación en este laboratorio fue clave para el desarrollo del mismo. Nos comprometemos a seguir con dicha sincronía para realizar los laboratorios.

7. **¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.**

-https://docs.oracle.com/javase/8/docs/api/

-Medina, D. (2020, 26 de mayo). Using split() and trim() for data cleaning in JavaScript. Medium. https://medium.com/@davidmedina0907/using-split-andtrim-for-data-cleaning-in-javascript-1167ceb1d4d6