

PROGRAMACIÓN ORIENTADA A OBJETOS

Herencia e interfaces

ADEMÁS Java desde consola

Laboratorio 3/6

Integrantes:

Daniel Useche

Daniel Patiño

DESARROLLO

Conociendo

1. En el directorio descarguen los archivos contenidos en schelling.zip. Revisen el código:

- a) ¿Cuántos paquetes tiene?

R: Tiene dos paquetes.

- d) ¿Cuál es el propósito del paquete presentación?

R: El programa simula una ciudad con agentes y elementos gráficos, mostrando su estado en una cuadrícula. Al presionar "Tic-tac", la simulación avanza un paso y se actualiza la interfaz visualmente.

- e) ¿Cuál es el propósito del paquete dominio?

R: Realizar toda la funcionalidad de la ciudad, en pocas palabras, el funcionamiento fundamental de la ciudad, las personas.

2. Revisen el paquete de dominio:

- a) ¿Cuáles son los diferentes tipos de componentes de este paquete?

R: Hay 3 clases en total, 2 concretas y una abstracta y también tiene una interfaz.

- b) ¿Qué implica cada uno de estos tipos de componentes?

- R: Las clases concretas: Tienen como objetivo crear objetos dentro del programa.

La clase abstracta: Tiene como objetivo definir una estructura común para las clases relacionadas.

La interfaz: Tiene como objetivo especificar e implementar los métodos que tienen que utilizar las clases relacionadas con esta.

3. Revisen el paquete de presentación,

- a) ¿Cuántos componentes tiene?
 - Tiene solamente dos clases, una de ellas es una clase interna.
- b) ¿Cuántos métodos públicos propios (no heredados) ofrece?
 - Ofrece 6 métodos, pero solo 1 es público y no heredado.

4. Para ejecutar un programa en java, ¿Qué método se debe ejecutar? ¿En qué clase se encuentra?

- El método Main que está en la clase principal, en este caso CityGUI().

5. Ejecuten el programa.

- ¿Qué funcionalidades ofrece?

Las principales funcionalidades que ofrece es un tablero cuadriculado donde en la parte inferior hay un botón llamado “Tic Tac”, el cual actualmente no tiene ninguna funcionalidad.

- ¿Qué hace actualmente? ¿Por qué?
Actualmente no hace nada más que crear el tablero con un botón que todavía no tiene funcionalidad porque el código no se encuentra completo.

Arquitectura General

1. Consulte el significado de las palabras package e import de java. ¿Qué es un

paquete? ¿Para qué sirve? ¿Para qué se importa? Explique su uso en este programa.

Significado de package e import en Java

¿Qué es un package en Java?

Un paquete (package) en Java es una agrupación de clases e interfaces relacionadas dentro de un mismo espacio de nombres. Se utiliza para organizar mejor el código y evitar conflictos entre nombres de clases.

Ejemplo:

`package presentation;`

- En el código proporcionado, CityGUI y PhotoCity están dentro del paquete presentation.
- Esto indica que este archivo (CityGUI.java) pertenece a la carpeta presentation dentro del proyecto.

¿Para qué sirve un paquete?

- Organiza mejor el código.
- Permite reutilizar clases sin conflictos de nombres.
- Facilita el control de acceso a las clases (puede hacerlas públicas o restringidas).

¿Para qué sirve import en Java?

La palabra clave import permite usar clases de otros paquetes sin necesidad de escribir su ruta completa.

Ejemplo en el código:

```
import domain.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
```

- `import domain.*;` → Importa todas las clases del paquete `domain`, donde probablemente esté la clase `City` y `Item`.
- `import javax.swing.*;` → Importa las clases de la biblioteca `Swing` para crear la interfaz gráfica (`JFrame`, `JPanel`, `JButton`).
- `import java.awt.*;` → Importa clases de `java.awt` para dibujar y manejar gráficos.
- `import java.awt.event.*;` → Importa clases para manejar eventos (como hacer clic en el botón).
- `import java.io.*;` → Importa clases de entrada/salida (`File`, `BufferedReader`, etc.), aunque en este código no se están usando.

¿Para qué se importa?

- Para acceder a clases de otros paquetes sin escribir su ruta completa.
- Facilita la reutilización de código.
- Permite el uso de bibliotecas estándar de Java (`Swing`, `AWT`, etc.).

Uso en este programa

1. `package presentation;`
 - a. Indica que la clase `CityGUI` pertenece al paquete `presentation`.
2. `import domain.*;`
 - a. Permite usar `City`, `Item`, `Agent`, y otras clases sin escribir `domain.City`, `domain.Item`, etc.
3. `import javax.swing.*;` y `import java.awt.*;`
 - a. Permiten crear la interfaz gráfica con `JFrame`, `JPanel`, `JButton`, y `Graphics`.
4. `import java.awt.event.*;`
 - a. Permite manejar eventos como el clic en el botón "Tic-tac".

2. Revise el contenido del directorio de trabajo y sus subdirectorios.

- Describa su contenido.

R: Contenido del directorio de trabajo:

Este directorio contiene una carpeta `doc` la cual contiene toda la información de la documentación del programa, también contiene los paquetes `domain` y `presentation`, y un archivo `bluej` el cual contiene todo el código.

Contenido del subdirectorio de trabajo:

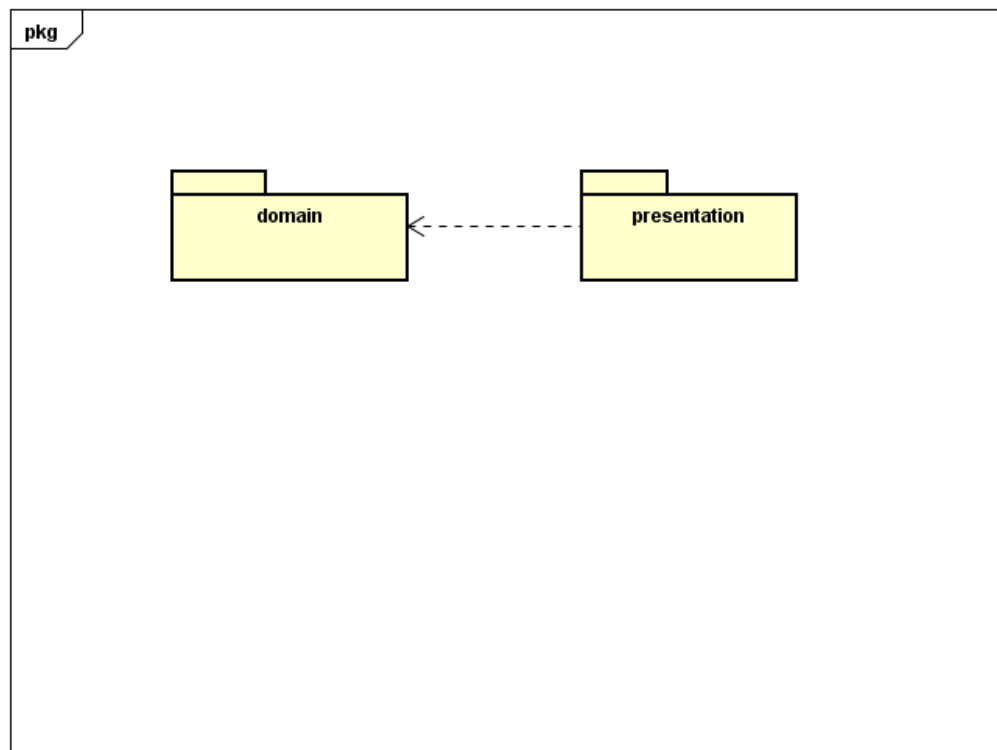
domain: Contiene todas las clases Agent, City y Person.

presentation: Contiene las clases CityGUI\$1, PhotoAManufacturing, PhotoAutomata, PhotoCity e informacion de ellas.

- ¿Qué coincidencia hay entre paquetes y directorios?

R: La coincidencia es que ambas guardan la información de las clases relacionadas entre sí.

3. Adicione al diseño la arquitectura general con un diagrama de paquetes en el que se presente los paquetes y las relaciones entre ellos. Consulte la referencia en moodle. En astah, crear un diagrama de clases (cambiar el nombre por Package Diagram0)



Arquitectura detallada.

1. Para preparar el proyecto para BDD. Completen el diseño detallado del paquete de dominio. Adicionen el diagrama de clases en el paquete correspondiente.

a) ¿Qué componentes hacían falta?
2. Completen el diseño detallado del paquete de presentación. Adicionen el diagrama de clases al paquete correspondiente. a) ¿Por qué hay dos clases y un archivo .java?
3. Adicione la clase de pruebas unitarias necesaria para BDD en un paquete independiente de test. (No lo adicione al diagrama de clases) ¿Qué paquete debe usar? ¿Por qué? ¿Asociado a qué clase? ¿Por qué?

Ciclo 1)

1. Estudie la clase City ¿Qué tipo de colección usa para albergar cosas? ¿Puede recibir personas? ¿Por qué?

Tiene una matriz bidimensional de objetos Item. Y sí puede recibir personas pues Agent .personas implementa Item.

2. Estudie el código asociado a la clase Person, ¿en qué estado se crea? ¿qué forma usa para pintarse? ¿cuándo aumenta su tiempo? ¿qué clases definen la clase Person ? Justifique sus respuestas.

Se crea en color azul, utiliza import java.awt.Color; para colorearse, aumenta su tiempo con el metodo steps que aumenta a su vez el atributo steps, las clases que definen a Person es Agent.

3. Person por ser un Agent, ¿qué atributos tiene? ¿qué puede hacer (métodos)? ¿qué decide hacer distinto? ¿qué no puede hacer distinto a todos los agentes? ¿qué debe aprender a hacer? Justifique sus respuestas.

Tiene los atributos:

```
public final static char HAPPY='h', INDIFFERENT='i', DISSATISFIED='d';  
protected char state;  
private int steps;
```

Los metodos son:

```
Step()  
GetSteps()  
IsAgent()  
IsHappy()  
isIndifferent()  
isDissatisfied()
```

¿qué puede hacer distinto a todos los agentes?

Todos los metodos no final.

¿qué no puede hacer distinto a todos los agentes?

Todos los metodos final.

Debe aprender a
de la clase Agent no tiene que aprender nada.

4. Por comportarse como un Item, ¿qué sabe hacer? ¿qué decide hacer distinto?
¿qué no puede hacer distinto? ¿qué debe aprender a hacer? Justifique sus
respuestas.

```
public default int shape(){
    return ROUND;
}

public default Color getColor(){
    return Color.black;
};

public default boolean isActive(){
    return true;
}

public default boolean isAgent(){
    return false;
}
```

¿qué decide hacer distinto?

```
public void decide(){
    state=(getSteps() % 3 == 0 ? Agent.HAPPY: (getSteps() % 3 == 1 ?
    Agent.INDIFFERENT:      Agent.DISSATISFIED));
}
```

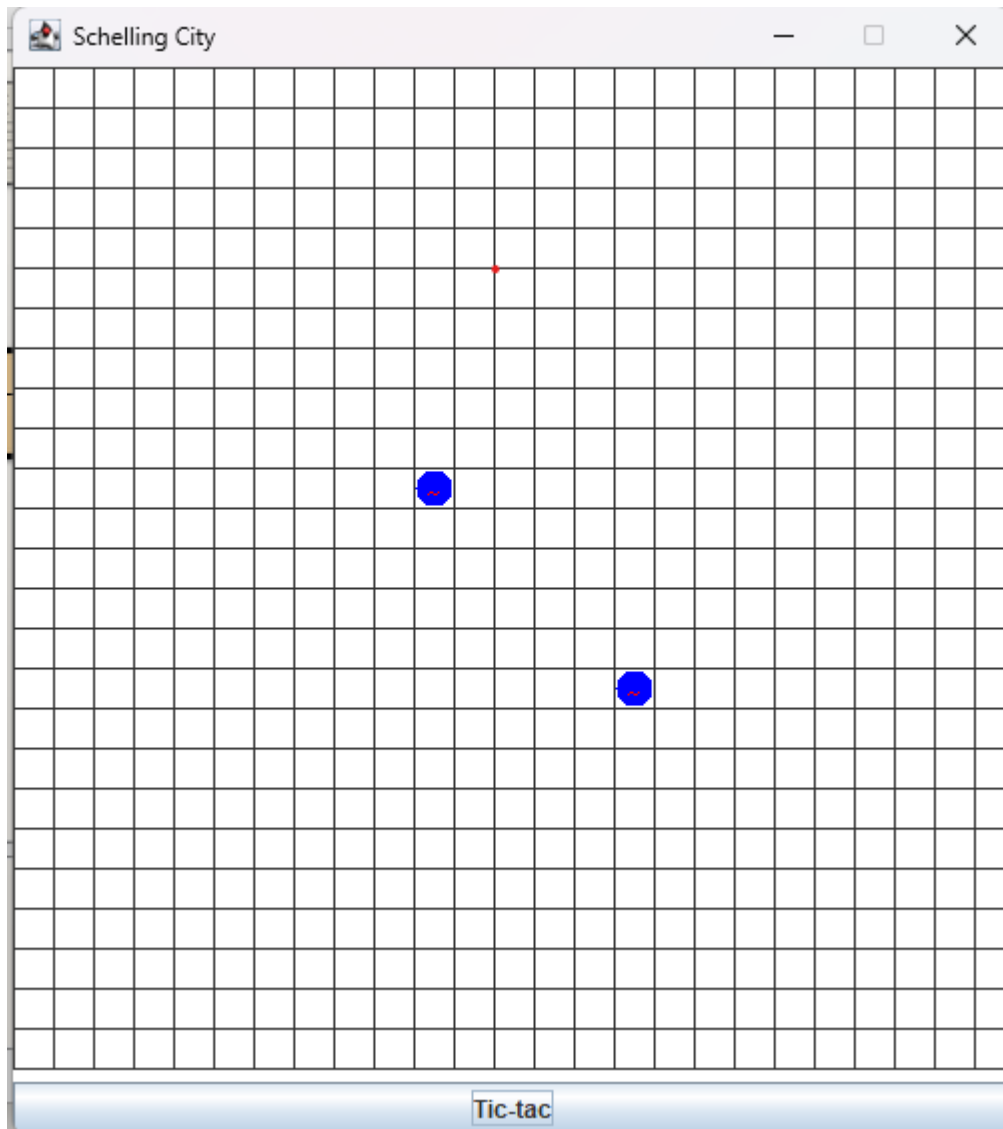
No puede hacer distintos a los demas son los default.

5. De acuerdo a lo anterior una Person, ¿Cómo actúa (decide+cambia)?

```
/**Act */ public void decide(){ state=(getSteps() % 3 == 0 ? Agent.HAPPY: (getSteps() % 3 ==  
1 ? Agent.INDIFFERENT: Agent.DISSATISFIED)); }  
  
/**Change its actual state  
*/  
public final void change(){  
    step();  
}
```

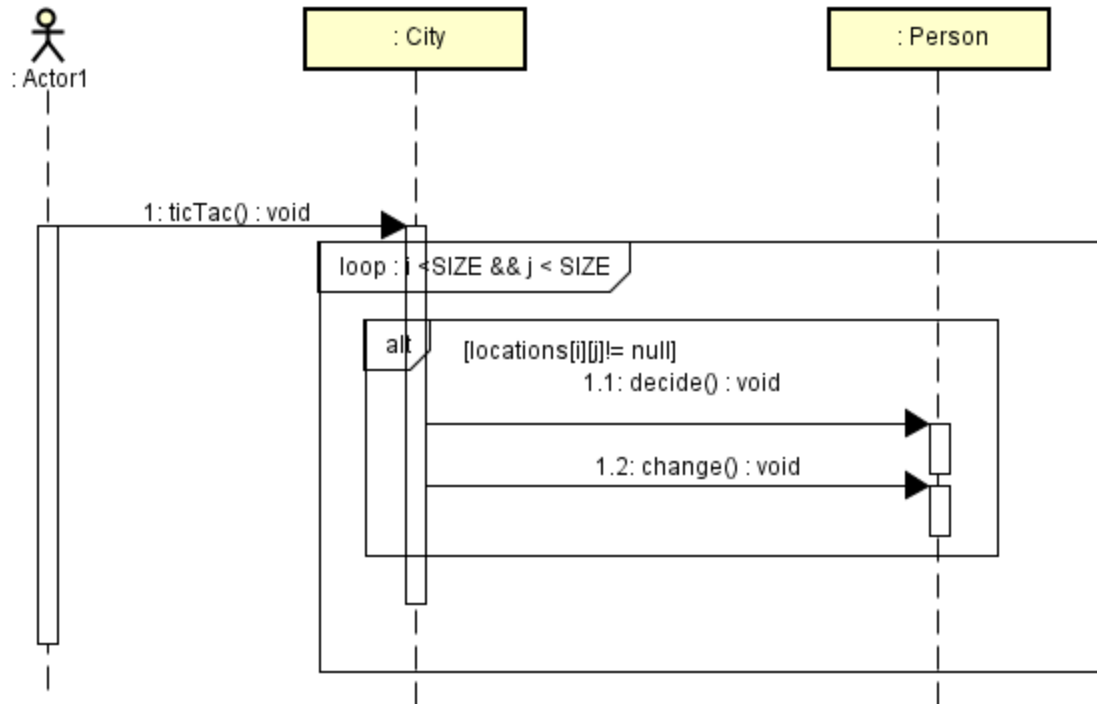
Sí, cada que cambia, vuelve a decidir y esto resulta en su actuar.

6. Ahora vamos a crear dos personas en diferentes posiciones (10,10) (15,15) llámelas adan y eva usando el método someltems() . Ejecuten el programa, ¿Qué pasa con las personas? ¿Por qué? Capturen una pantalla significativa.



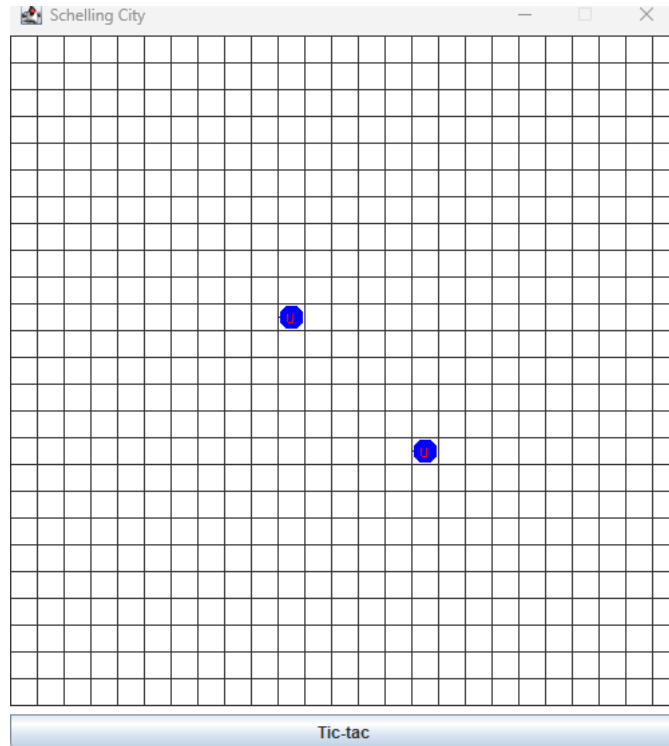
7. Diseñen, construyan y prueben el método llamado ticTac() de la clase City.

Diagram0

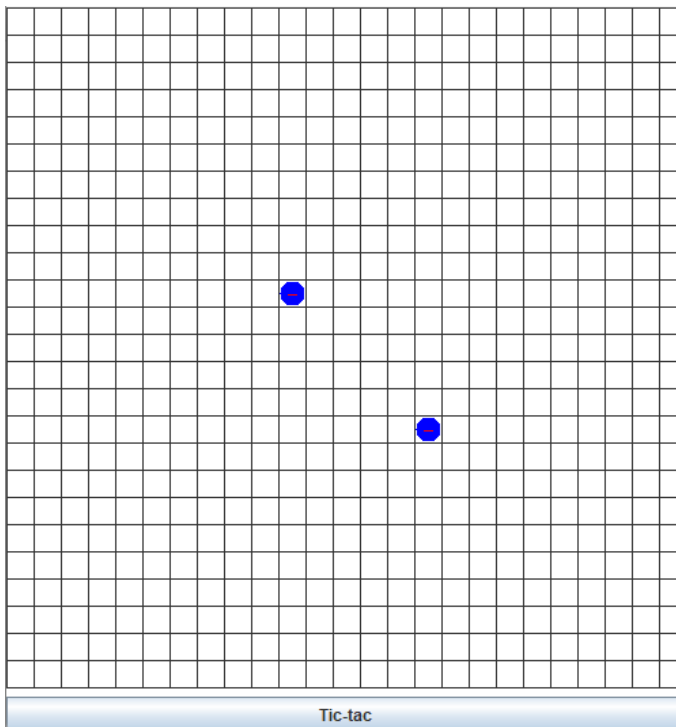


8. ¿Cómo quedarían adan y eva después de uno, dos, cuatro y seis Tic-tac? Ejecuten el programa. Capturen pantallas significativas en momentos correspondientes. ¿Es correcto?

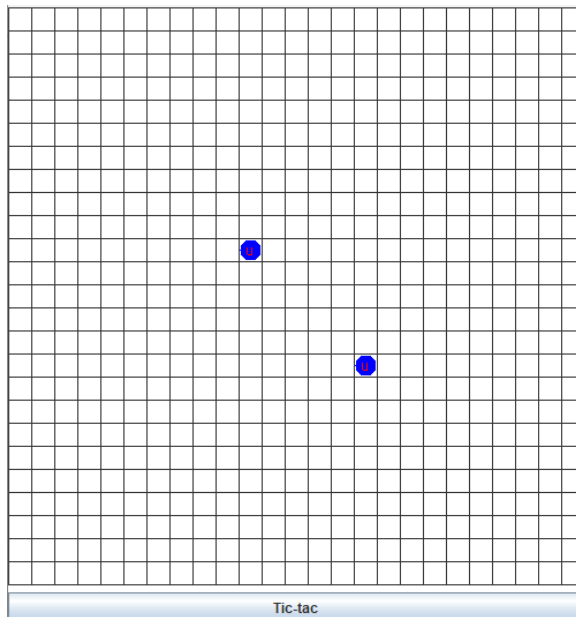
1.(uno)



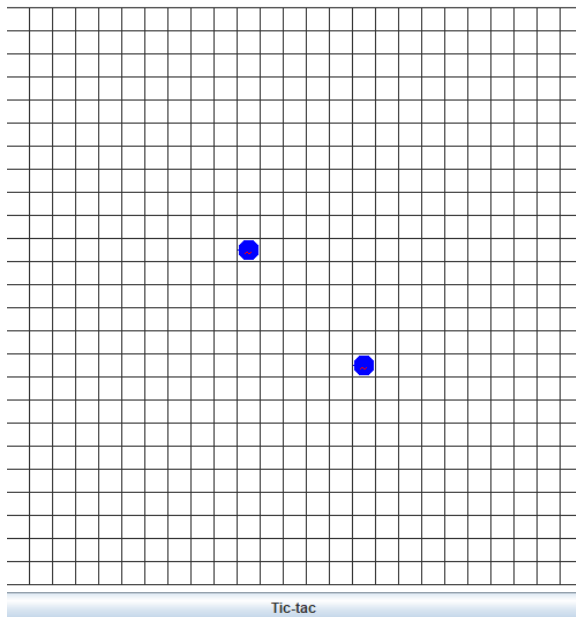
2.(dos)



3.(cuatro)



4.(seis)

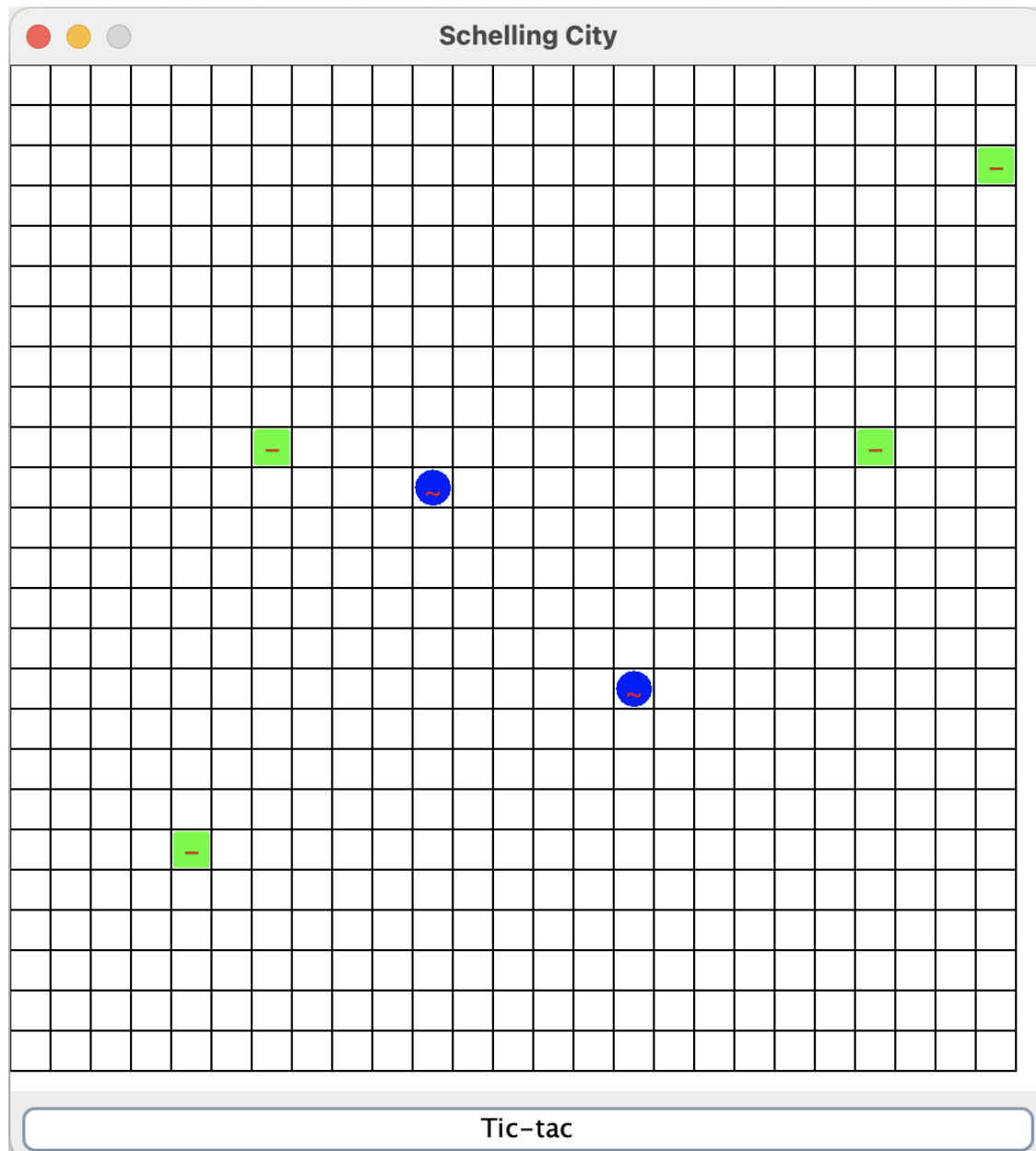


Es correcto pues primero decide y después cambia, por lo que la decisión es correcta.

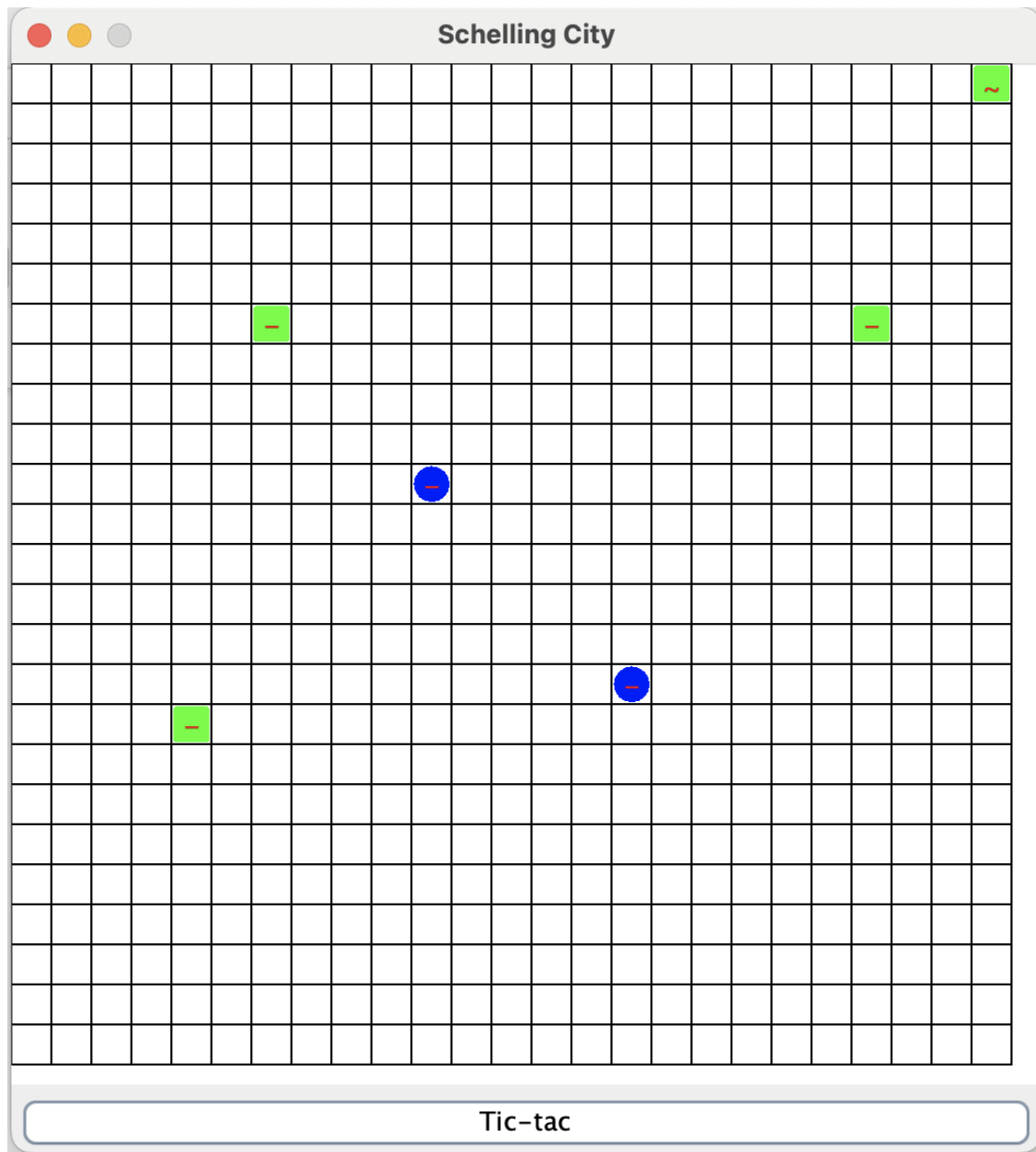
1. Ciclo 2)

Para implementar esta nueva persona Walker ¿cuáles métodos se sobre-escriben (overriding)?

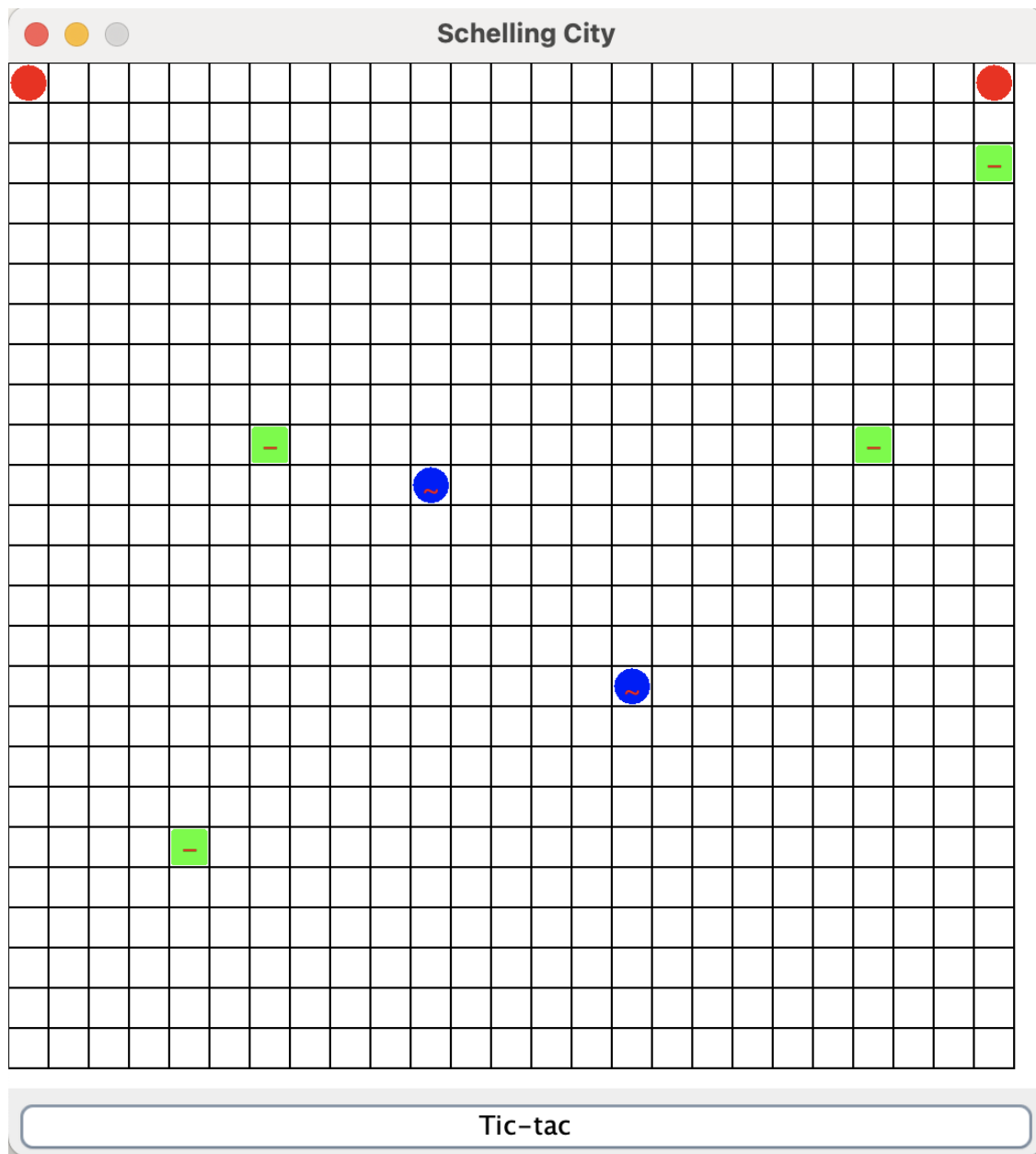
Sobre escribimos Shape decide y change



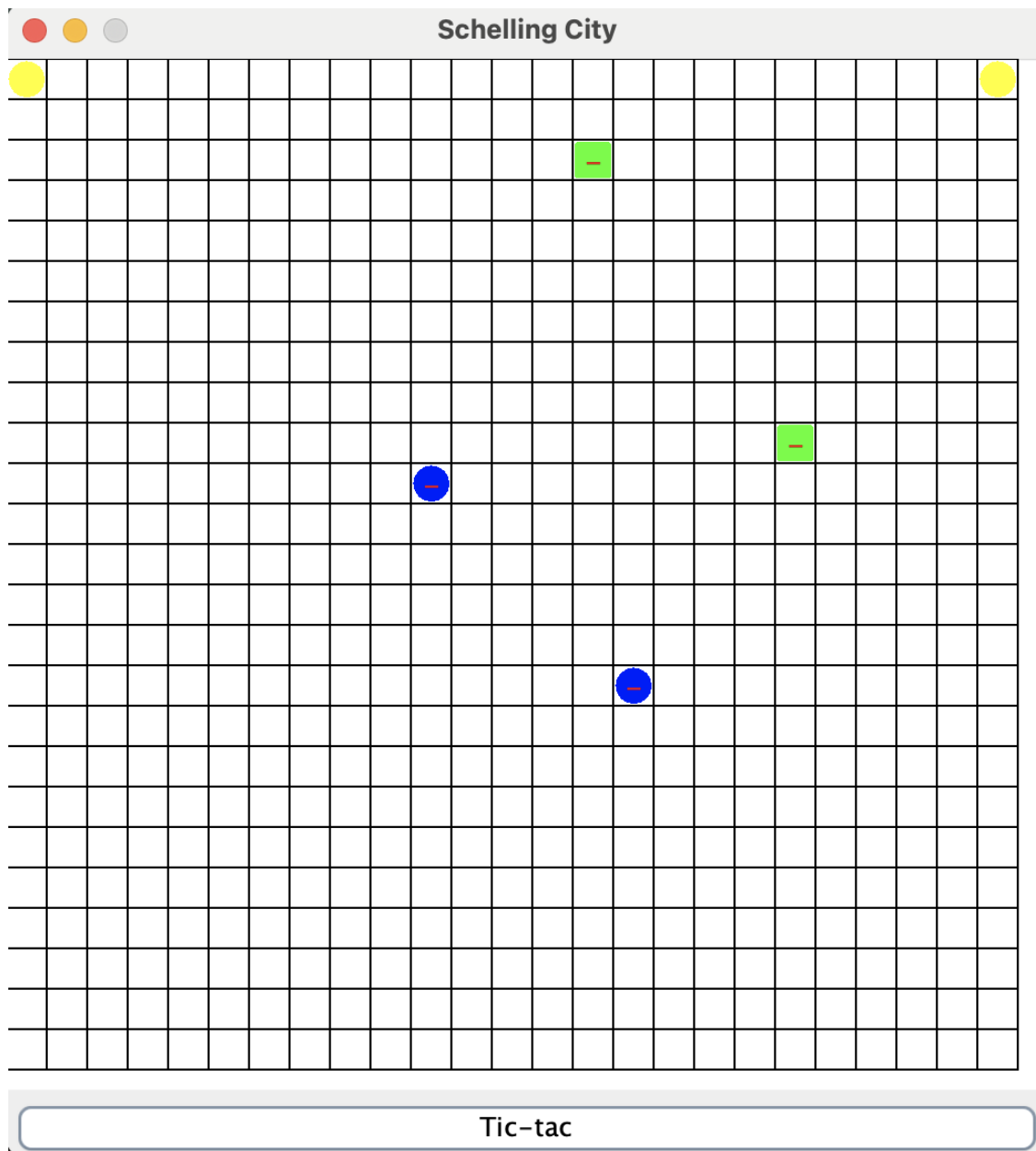
Despues de los tics:

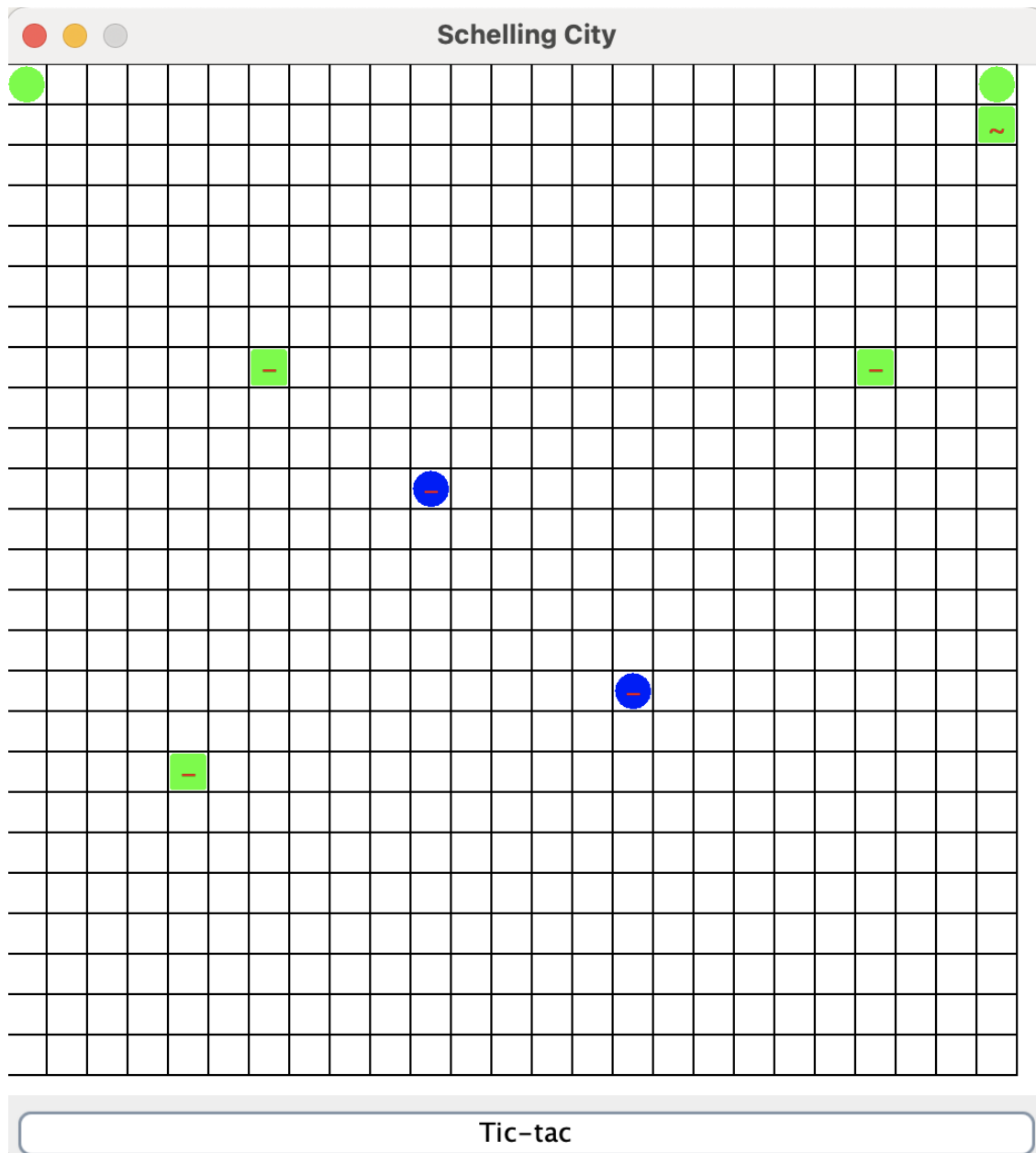


Ciclo 3)

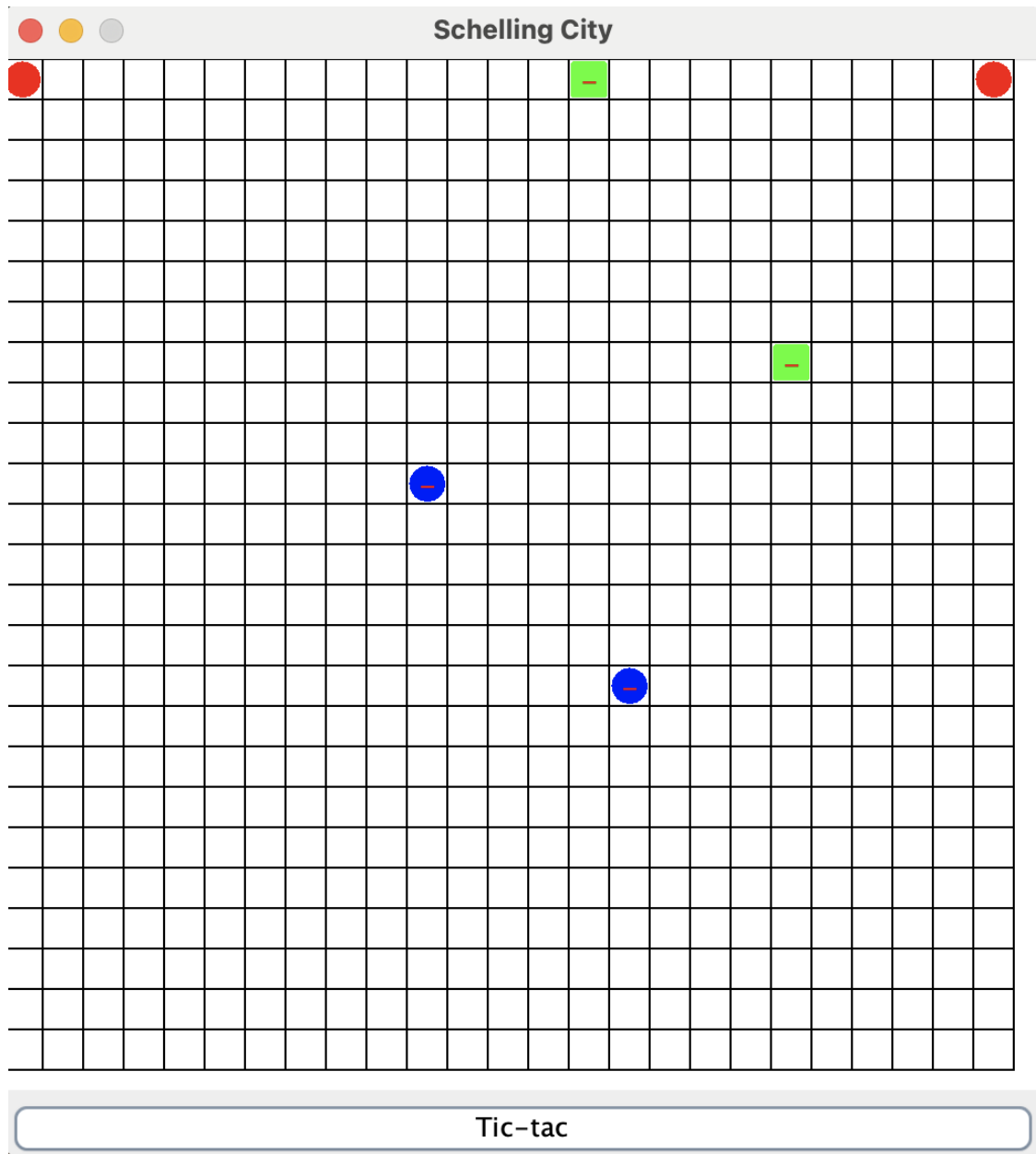


Despues de los tics requeridos





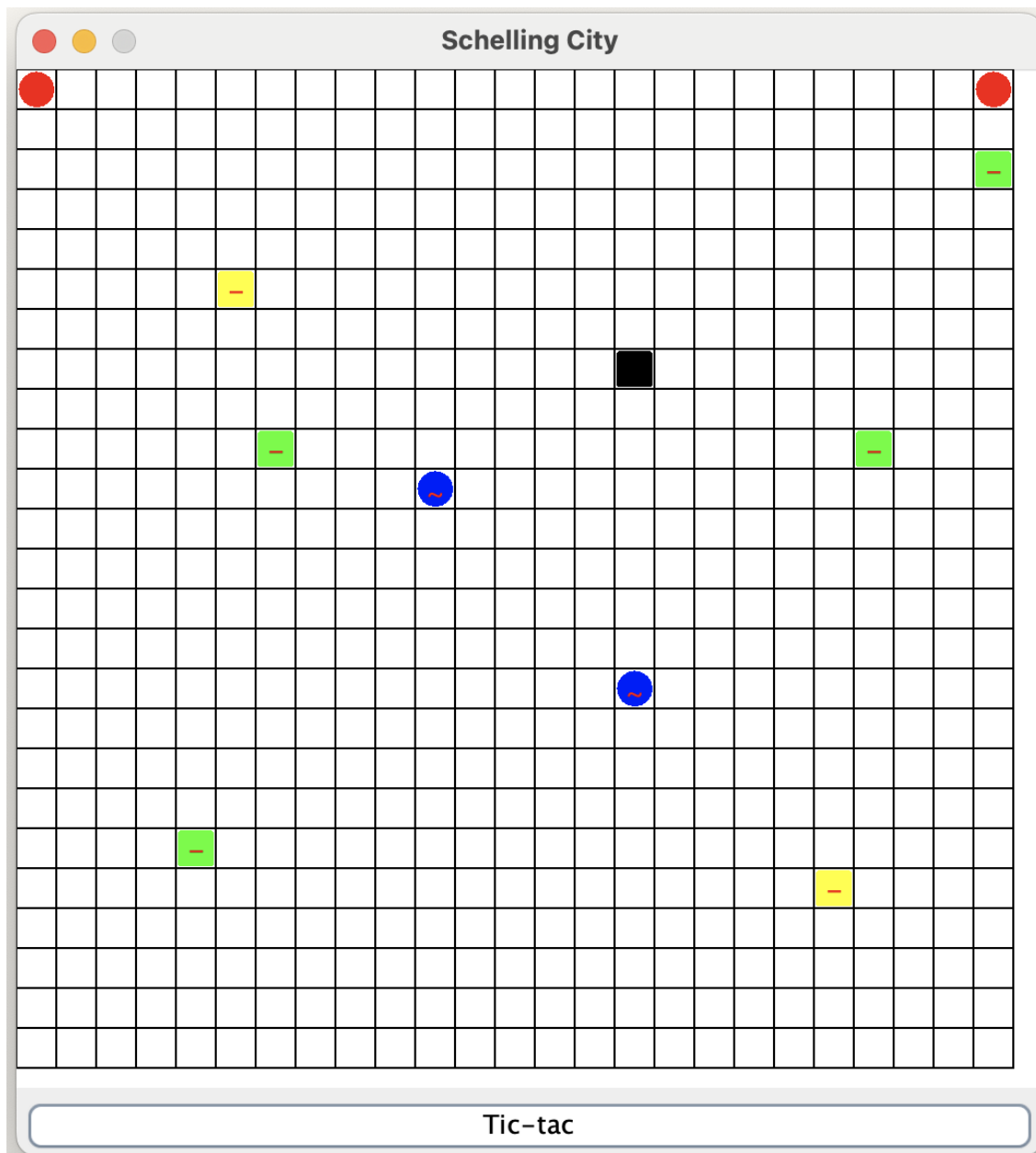
Por ultimo:



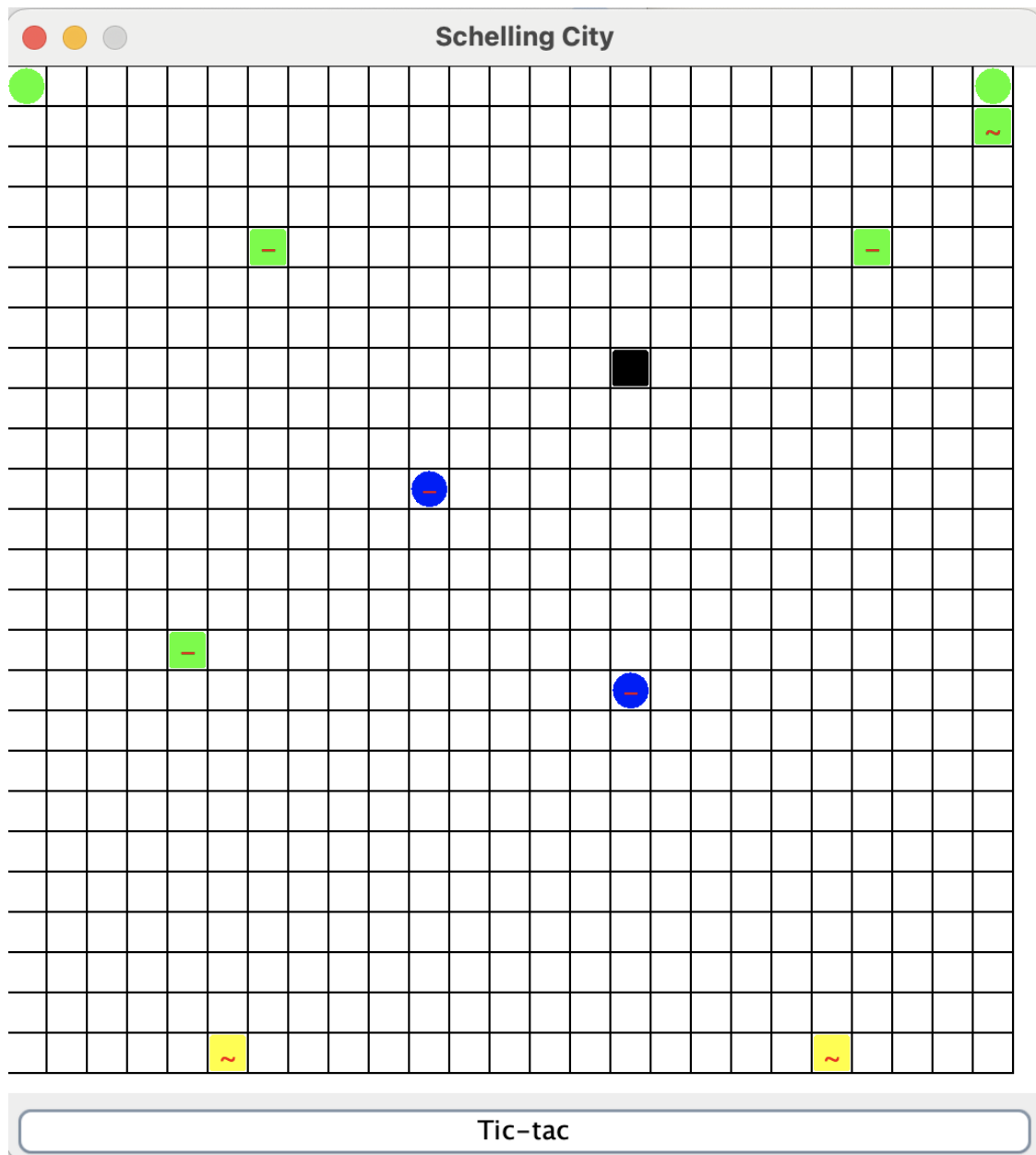
Lo cual es correcto ya que el semáforo funciona correctamente.

No debimos modificar nada en city ademas de Agergar los Items en Someltems en la clase City.

Ciclo 4)

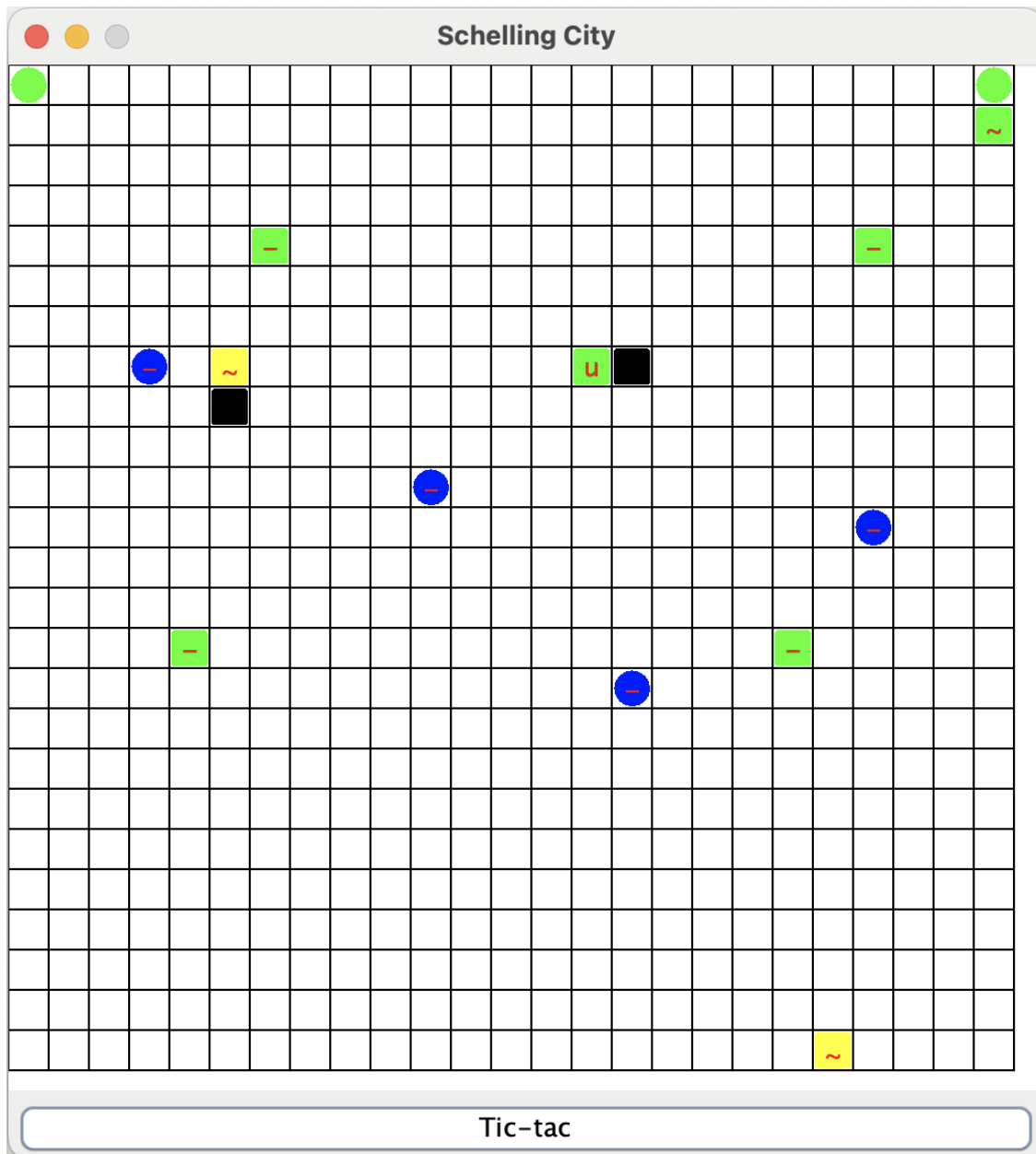


Despues de los tics

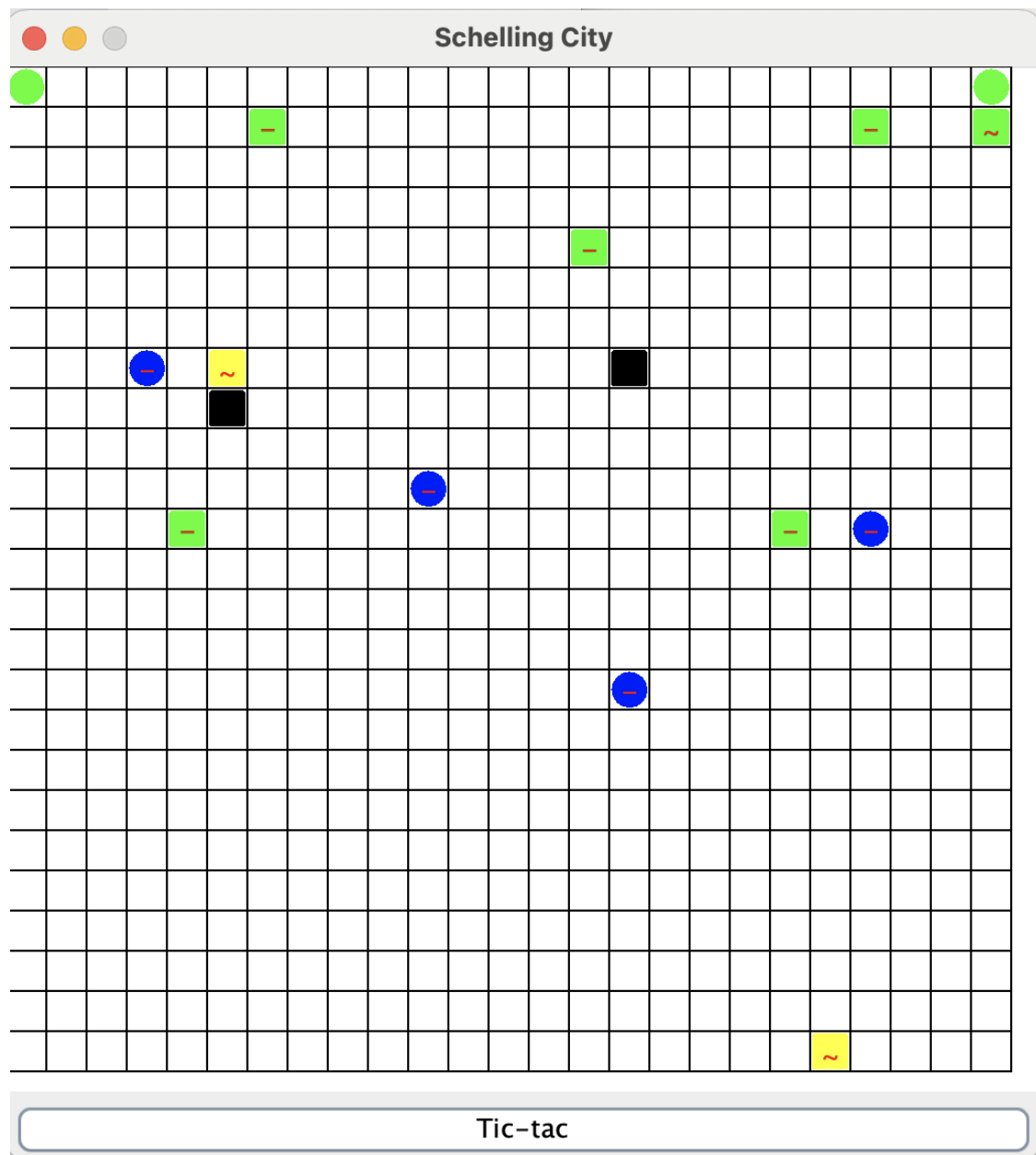


Decidimos crear una partícula que se comporta de manera extraña, en contra via a las demas siendo imposible de satisfacer, no le gusta está acompañada ni que le nieguen el poder seguir su camino.

Ciclo 5)

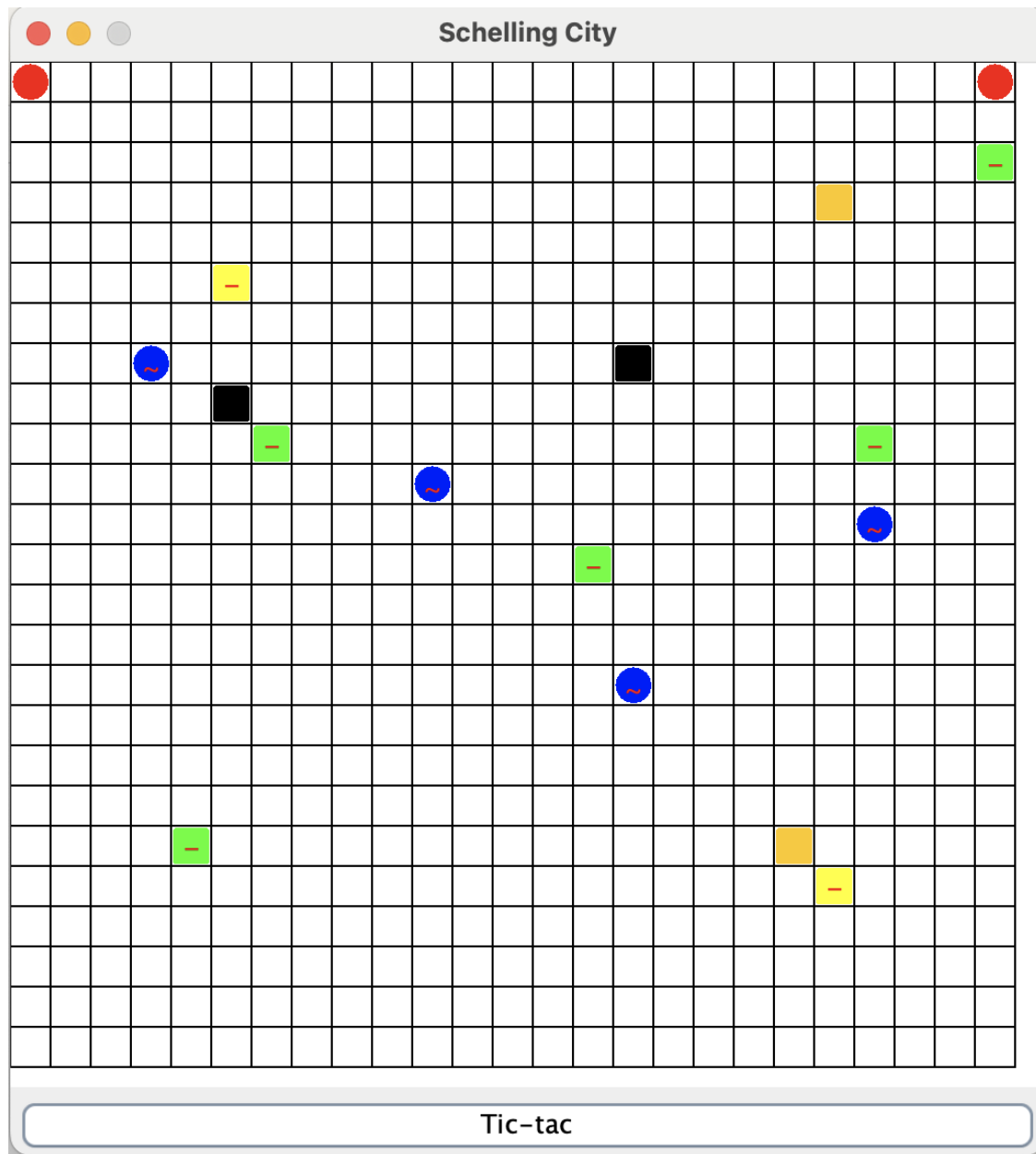


Creamos el item Building, para crear los edificios que representa los cuadrados negros en la ciudad.



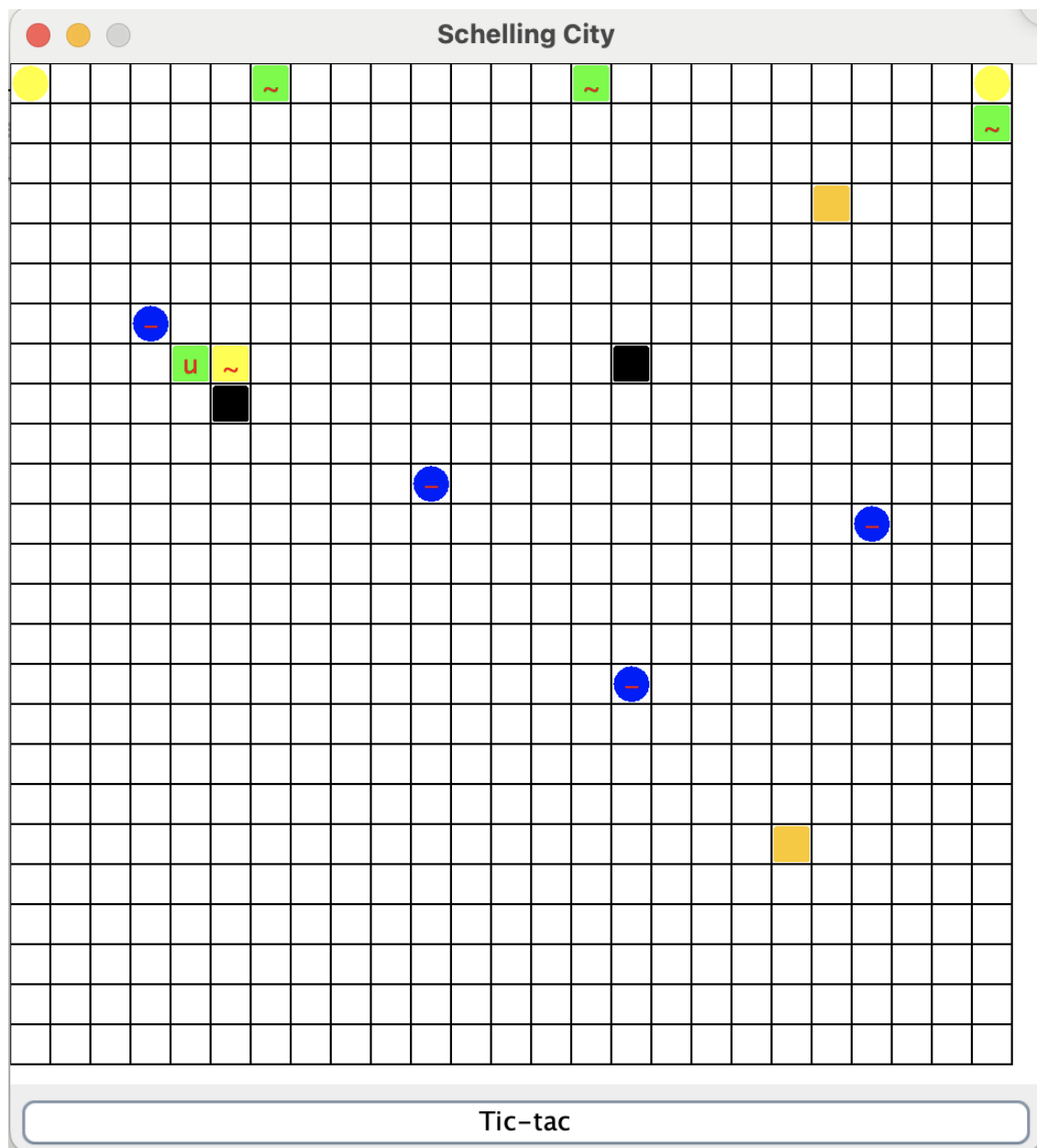
Ciclo 6)

Agregamos los Beast

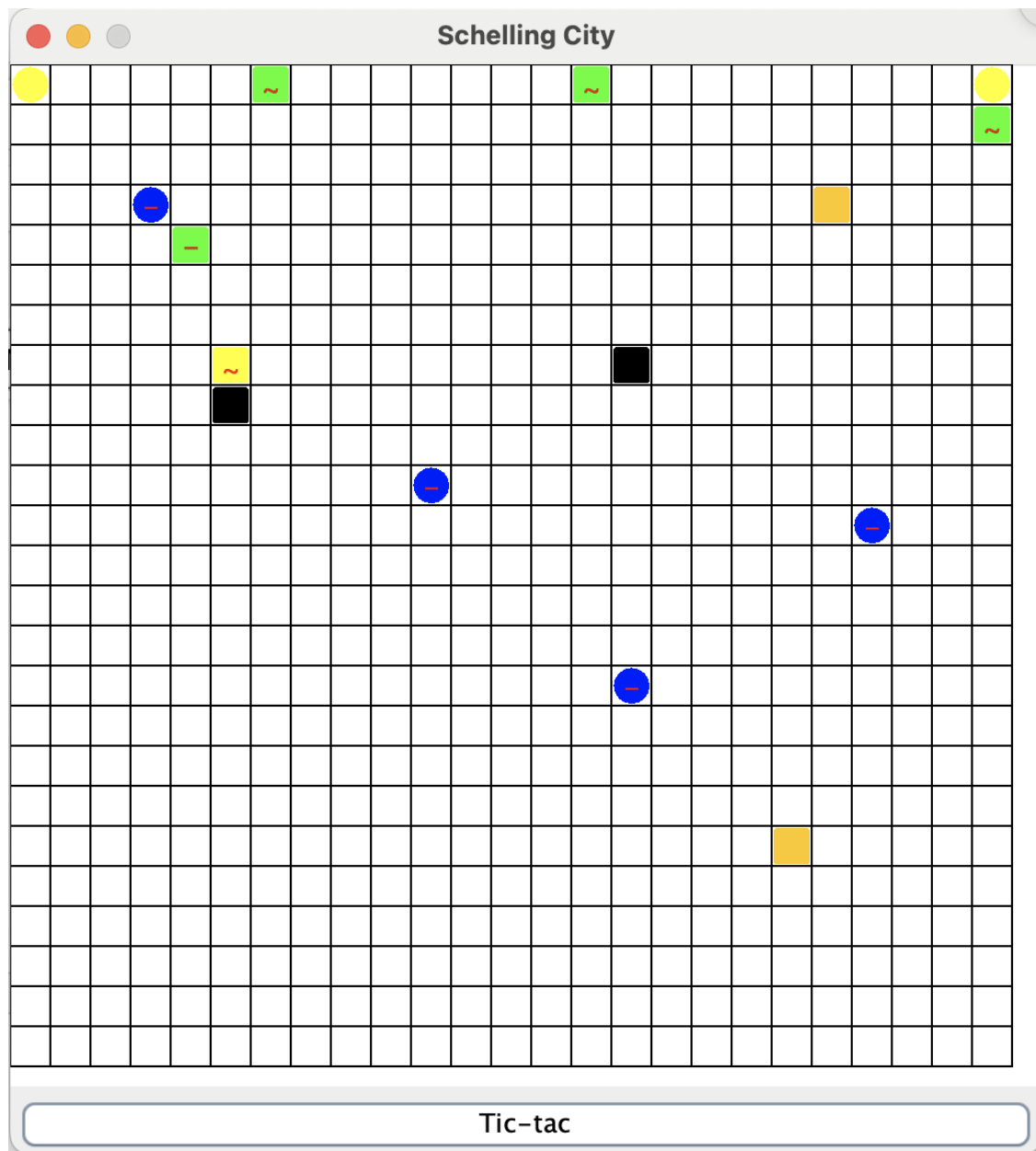


Despues de unos tics pueden desaparecer otras Person

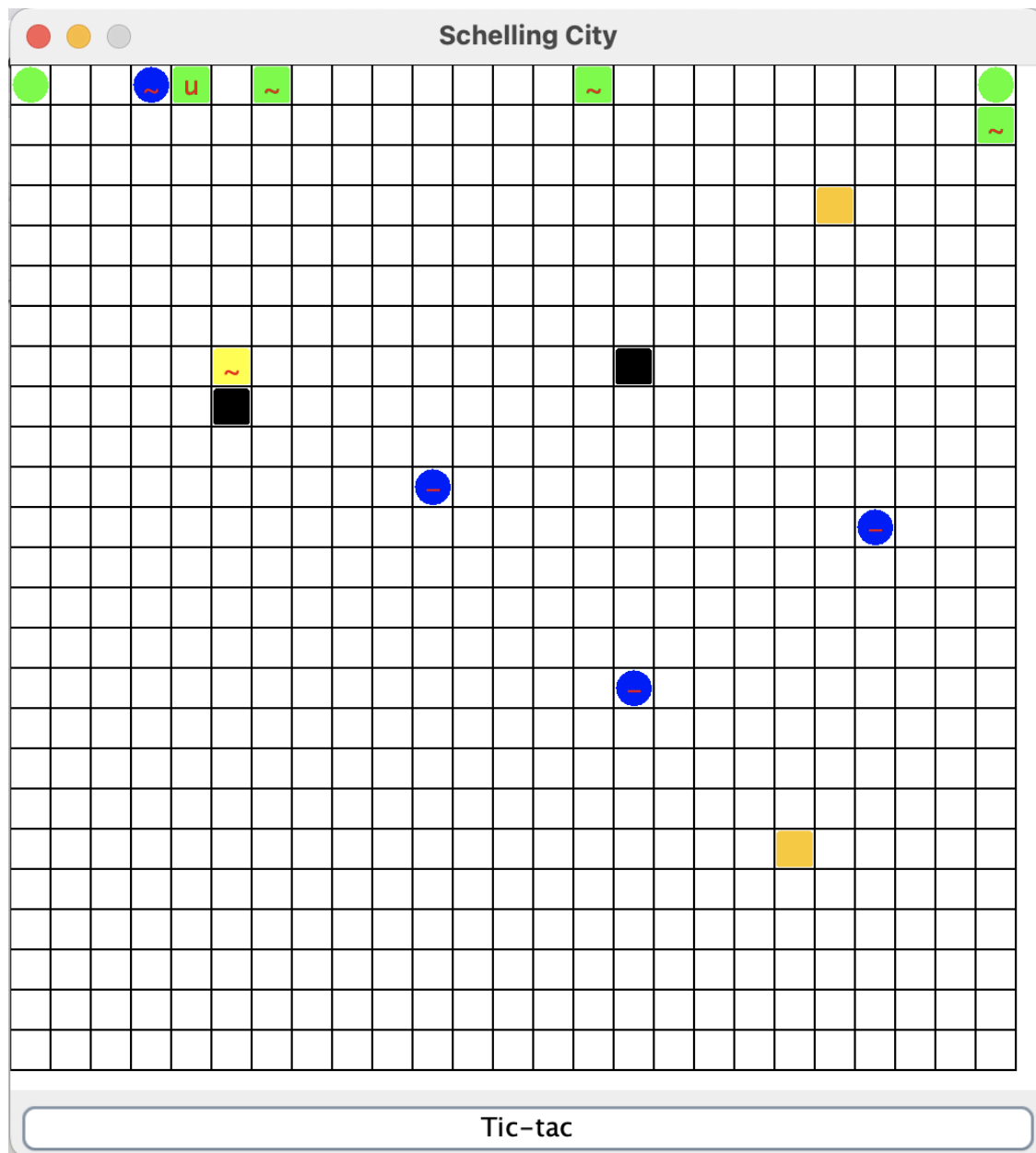
Las Perdona Schelling:



Se mueve de acuerdo a lo especificado en el problema.



Todos interactuan entre si.



DE BLUEJ A CONSOLA

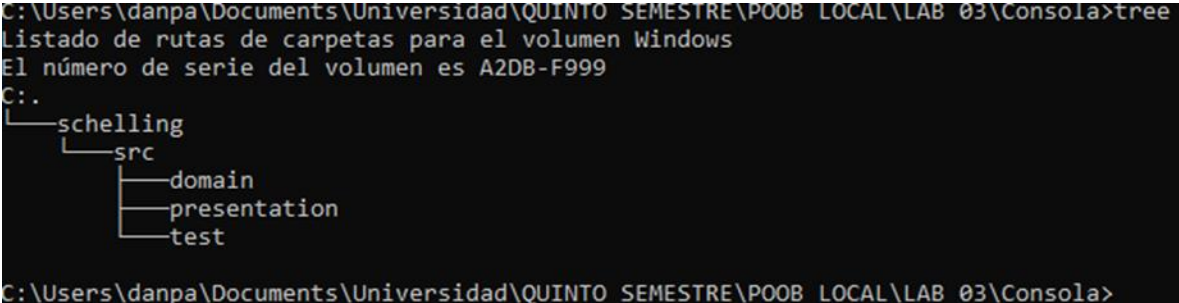
Comandos básicos del sistema operativo [En lab03.doc]

es de iniciar debemos repasar los comandos básicos del manejo de la consola.

1. Investiguen los comandos para moverse en la estructura de directorios: crear, borrar, listar su contenido y copiar o eliminar un archivo.
 - Crear un archivo desde la consola: Para crear un archivo en consola se pueden ejecutar los siguientes comandos:
 - o `echo Información del archivo > nombre.tipoDeArchivo`
 - o Para agregar más información `echo Informacion >> nombre.tipoDeArchivo.`
 - o `copy con archivo.tipoDeArchivo,` luego se le da ENTER y podemos empezar a escribir, para guardar la información del archivo le damos en CTRL +Z y ENTER.
 - o `type nul > archivo.tipoDeArchivo`
 - o con ubicación específica: `echo información del archivo > C:\Users\Usuario\Desktop\archivo.txt`
 - Listar contenido de un archivo:
 - o `type archivo. tipoDeArchivo:` este comando mostrara todo el contenido del archivo por líneas.
 - o `more archivo. tipoDeArchivo:` este comando se utiliza para cuando el archivo es muy extenso, se dividirá por páginas y para pasar de pagina se utiliza ENTER o la BARRA ESPACIADORA.
 - o `notepad archivo.tipoDeArchivo:` Lo que hace esta función es abrirlo desde el bloc de notas.
 - o `find "palabra" archivo.tipoDeArchivo:` Lo que hace este comando es buscar por líneas una palabra en específico.
 - o `findstr "palabra1 palabra2" archivo.tipoDeArchivo:` Lo que hace este comando es buscar múltiples palabra en un archivo.
 - o `type "C:\Users\TuUsuario\Desktop\archivo.txt":` Si se necesita buscar la información de alguna otra carpeta.
 - Copiar un archivo:
 - o `copy direccionOrigen direccionDestino:` Se utiliza para copiar un archivo de una parte a otra.
 - o `xcopy archivo.tipoDeArchivo C:\ Destino:` Ofrece mas opciones para copiar un archivo
 - o `xcopy C:\Origen\archivo.txt C:\Destino\ /I /Y:` Se utiliza para copiar archivos manteniendo la estructura de subcarpetas.
 - o `xcopy C:\Origen C:\Destino /E /I /Y:` Se utiliza para copiar toda la carpeta con todos los archivos
 - o `robocopy C:\Origen C:\Destino archivo.tipoDeArchivo:` copia los archivos de una carpeta de manera avanzada, este a comparación del xcopy es mas potente y permite copiar carpetas mas grandes.
 - o `robocopy C:\Origen C:\Destino /E:` Copia todos los archivos de una carpeta.

- Eliminar un archivo:
 - o del archivo.tipoDeArchivo: Elimina el archivo de la carpeta actual.
 - o del C:\RutaDelArchivo\archivo.tipoDeArchivo: Elimina el archivo en esa ubicación.
 - o Del /Q archivo.tipoDeArchivo: Elimina el archivo evitando la confirmación.
 - o del /S archivo.tipoDeArchivo: Elimina el archivo de varias subcarpetas, lo que quiere decir que eliminara los archivos con ese nombre dentro de la carpeta y subcarpetas.
 - o erase archivo.tipoDeArchivo: Elimina el archivo de la carpeta actual.
 - o del C:\RutaDelArchivo\archivo.tipoDeArchivo: elimina la carpeta con archivos dentro.
 - o rd /S /Q C:\RutaDelArchivo: Forma alternativa de eliminar la carpeta con archivos dentro.
2. Organicen un nuevo directorio con la estructura propuesta para probar desde allí su habilidad con los comandos de consola. Consulten y capturen el contenido de su directorio.

Para crear los directorios se utilizo el comando mkdir 'Nombre del directorio' y para ubicarse dentro del directorio se utiliza cd C:\Ubicacion del archivo.



```

C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03\Consola>tree
Listado de rutas de carpetas para el volumen Windows
El número de serie del volumen es A2DB-F999
C:
├──schelling
│   └──src
│       ├──domain
│       ├──presentation
│       └──test
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03\Consola>
  
```

3. En el directorio copien únicamente los archivos *.java del paquete de aplicación. Consulte y capture el contenido de src/domain
- Para copiar los archivos .java utilizamos el siguiente comando:
 - o cd C:\UbicacionDelDomain\schelling\domain: Para ubicarnos en el directorio donde copiaremos los archivos.
 - o copy *.java "C:\UbicacionDelArchivo\schelling\src\domain": copiamos todos los archivos .java y los pegamos en la dirección deseada, en nuestro caso es esa.

Como primer paso identificamos los archivos .class de nuestro proyecto original

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\schelling\domain>dir *.class
El volumen de la unidad C es Windows
El número de serie del volumen es: A2DB-F999

Directorio de C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\schelling\domain
22/03/2025  12:14 a. m.          1.003 Agent.class
22/03/2025  06:45 p. m.          1.364 Beast.class
22/03/2025  06:45 p. m.           830 Building.class
22/03/2025  06:45 p. m.          3.783 City.class
22/03/2025  04:11 p. m.           697 Item.class
22/03/2025  06:45 p. m.          1.206 Person.class
22/03/2025  06:45 p. m.          1.036 TrafficLight.class
22/03/2025  06:45 p. m.          1.113 Walker.class
22/03/2025  06:45 p. m.          1.126 Wallflower.class
          9 archivos          12.158 bytes
          0 dirs  57.937.379.328 bytes libres

C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\schelling\domain>
```

Seguido a eso procedemos a crear los dos subdirectorios faltantes:

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03\Consola\schelling>mkdir bin
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03\Consola\schelling>mkdir docs
```

Ya creados los subdirectorios procedemos a copiar los archivos .class a nuestro nuevo directorio bin

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\schelling\domain>xcopy *.class "C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\Consola\schelling\bin"
C:\Agent.class
C:\Beast.class
C:\Building.class
C:\City.class
C:\Item.class
C:\Person.class
C:\TrafficLight.class
C\Walker.class
C\Wallflower.class
9 archivo(s) copiado(s)

C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\schelling\domain>
```

También copiamos los .class del subdirectorio original de presentation:

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\schelling\presentation>xcopy *.class "C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03\Consola\schelling\bin"
C:\CityGUI$1.class
C:\CityGUI.class
C\PhotoManufacturing.class
C\PhotoAutomata.class
C\PhotoCity.class
9 archivo(s) copiado(s)

C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\schelling\presentation>
```

Este es el resultado final:

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\Consola\schelling\bin>dir *.class
El volumen de la unidad C es Windows
El número de serie del volumen es: A2DB-F999

Directorio de C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\Consola\schelling\bin
22/03/2025  12:14 a. m.          1.003 Agent.class
22/03/2025  06:45 p. m.          1.364 Beast.class
22/03/2025  06:45 p. m.           830 Building.class
22/03/2025  06:45 p. m.          3.783 City.class
22/03/2025  11:00 a. m.           760 CityGUI$1.class
22/03/2025  11:00 a. m.          2.093 CityGUI.class
22/03/2025  04:11 p. m.           697 Item.class
22/03/2025  06:45 p. m.          1.206 Person.class
21/03/2025  07:00 p. m.          1.984 PhotoAManufacturing.class
21/03/2025  07:00 p. m.          1.917 PhotoAutomata.class
22/03/2025  11:00 a. m.          2.277 PhotoCity.class
22/03/2025  06:45 p. m.          1.036 TrafficLight.class
22/03/2025  06:45 p. m.          1.113 Walker.class
22/03/2025  06:45 p. m.          1.126 Wallflower.class
         14 archivos         21.189 bytes
          0 dirs  57.928.269.824 bytes libres
```


Comandos de java [En lab03.doc]

1. Consulte para qué sirven cada uno de los siguientes comandos:

- `javac`: `javac` (Java Compiler) compila los archivos fuentes `.java` y genera archivos `.class`.
- `java`: `java` (Java Runtime) ejecuta un programa Java a partir de un archivo `.class`.
- `javadoc`: Genera una documentación HTML a partir de los comentarios en el código fuente.
- `jar`: `jar` (Java Archive) Empaqueta múltiples archivos `.class` en un único `.jar`.

2. Cree una sesión de consola y consulte en línea las opciones de los comandos `java` y `javac`. Capture las pantallas.

Cuando se ejecuta el comando `java` aparece el siguiente menú:

```

C:\>java
Sintaxis: java [-options] class [args...]
          (para ejecutar una clase)
    o  java [-options] -jar jarfile [args...]
        (para ejecutar un archivo jar)
donde las opciones incluyen:
    -d32      usar un modelo de datos de 32 bits, si está disponible
    -d64      usar un modelo de datos de 64 bits, si está disponible
    -server   para seleccionar la VM "server"
              La VM por defecto es server.

    -cp <ruta de acceso de búsqueda de clases de los directorios y los archivos zip/jar>
    -classpath <ruta de acceso de búsqueda de clases de los directorios y los archivos zip/jar>
                Lista separada por ; de directorios, archivos JAR
                y archivos ZIP para buscar archivos de clase.
    -D<nombre>=<valor>
                definir una propiedad del sistema
    -verbose[:<class>|<gc>|<jni>]
                activar la salida verbose
    -version   imprimir la versión del producto y salir
    -version:<valor>
                Advertencia: Esta función está en desuso y se eliminará
                en una versión futura.
                es necesario que se ejecute la versión especificada
    -showversion
                imprimir la versión del producto y continuar
    -jre-restrict-search | -no-jre-restrict-search
                Advertencia: Esta función está en desuso y se eliminará
                en una versión futura.
                incluir/excluir JRE privados de usuario en la búsqueda de versión
    -? -help   imprimir este mensaje de ayuda
    -X         imprimir la ayuda sobre las opciones que no sean estándar
    -ea[:<nombre paquete>...[:<nombre clase>]]
    -enableassertions[:<nombre paquete>...[:<nombre clase>]]
                activar afirmaciones con la granularidad especificada
    -da[:<nombre paquete>...[:<nombre clase>]]
    -disableassertions[:<nombre paquete>...[:<nombre clase>]]
                desactivar afirmaciones con la granularidad especificada
    -esa | -enablesystemassertions
                activar afirmaciones del sistema
    -dsa | -disablesystemassertions
                desactivar afirmaciones del sistema
    -agentlib:<nombre bib>[=<opciones>]
                cargar la biblioteca de agente nativa <nombre bib>, como -agentlib:hprof
                véase también -agentlib:jwp=help y -agentlib:hprof=help
    -agentpath:<nombre ruta acceso>[=<opciones>]
                cargar biblioteca de agente nativa con el nombre de la ruta de acceso completa
    -javaagent:<ruta acceso jar>[=<opciones>]
                cargar agente de lenguaje de programación Java, véase java.lang.instrument
    -splash:<ruta acceso imagen>
                mostrar una pantalla de presentación con la imagen especificada
Consulte http://www.oracle.com/technetwork/java/javase/documentation/index.html para obtener más información.
C:\>

```

Y cuando se ejecuta el comando javac aparece el siguiente menú:

```

PS C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\Java arenas> javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>           Read options and filenames from file
  -Akey[=value]         Options to pass to annotation processors
  --add-modules <module>(<module>)*
                        Root modules to resolve in addition to the initial modules,
                        or all modules on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, -bootclasspath <path>
                        Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
                        Specify where to find user class files and annotation processors
  -d <directory>        Specify where to place generated class files
  -deprecation
                        Output source locations where deprecated APIs are used
  --enable-preview
                        Enable preview language features.
                        To be used in conjunction with either -source or --release.
  -encoding <encoding>  Specify character encoding used by source files
  -endorseddirs <dirs>  Override location of endorsed standards path
  -extdirs <dirs>       Override location of installed extensions
  -g
                        Generate all debugging info
  -g:{lines,vars,source}
                        Generate only some debugging info
  -g:none
                        Generate no debugging info
  -h <directory>
                        Specify where to place generated native header files
  --help, -help, -?     Print this help message
  --help-extra, -X      Print help on extra options
  -implicit:{none,class}
                        Specify whether to generate class files for implicitly referenced files
  -J<flag>               Pass <flag> directly to the runtime system
  --limit-modules <module>(<module>)*
                        Limit the universe of observable modules
  --module <module>(<module>)*, -m <module>(<module>)*
                        Compile only the specified module(s), check timestamps
  --module-path <path>, -p <path>
                        Specify where to find application modules
  --module-source-path <module-source-path>
                        Specify where to find input source files for multiple modules
  --module-version <version>
                        Specify version of modules that are being compiled
  -nowarn
                        Generate no warnings
  -parameters
                        Generate metadata for reflection on method parameters
  -proc:{none,only,full}
                        Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...]
                        Names of the annotation processors to run;
                        bypasses default discovery process
  --processor-module-path <path>
                        Specify a module path where to find annotation processors
  --processor-path <path>, -processorpath <path>
                        Specify where to find annotation processors
  -profile <profile>

```

```

-profile <profile>
    Check that API used is available in the specified profile.
    This option is deprecated and may be removed in a future release.
--release <release>
    Compile for the specified Java SE release.
    Supported releases:
        8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
-s <directory>
    Specify where to place generated source files
--source <release>, -source <release>
    Provide source compatibility with the specified Java SE release.
    Supported releases:
        8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
--source-path <path>, -sourcepath <path>
    Specify where to find input source files
--system <jdk>|none
    Override location of system modules
--target <release>, -target <release>
    Generate class files suitable for the specified Java SE release.
    Supported releases:
        8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
--upgrade-module-path <path>
    Override location of upgradeable modules
-verbose
    Output messages about what the compiler is doing
--version, -version
    Version information
-Werror
    Terminate compilation if warnings occur
PS C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\Java arenas>

```

3. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.

En java:

```

C:\>java --version
java 24 2025-03-18
Java(TM) SE Runtime Environment (build 24+36-3646)
Java HotSpot(TM) 64-Bit Server VM (build 24+36-3646, mixed mode, sharing)

```

Como se puede evidenciar en la captura, la versión de java usada es la 24.

En javac:

```

PS C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\Java arenas>javac -version
javac 24

```

Como se evidencia en la captura, la versión usada del Java Compiler es la 24.

Compilando [En lab03.doc]

1. Utilizando el comando javac, desde el directorio raíz (desde schelling con una sola instrucción), compile el proyecto.

¿Qué instrucción completa tuvo que dar a la consola para compilar TODO el proyecto?










Para compilar todo el proyecto se utilizó el siguiente comando

```
javac -d bin -sourcepath src src/presentation/ *.java
```




Y quedo de la siguiente manera:

 domain	22/03/2025 7:07 p. m.	Carpeta de archivos
 presentation	22/03/2025 7:07 p. m.	Carpeta de archivos

Archivos de domain:

 Agent.class	22/03/2025 7:07 p. m.	Archivo CLASS	1 KB
 Beast.class	22/03/2025 7:07 p. m.	Archivo CLASS	2 KB
 Building.class	22/03/2025 7:07 p. m.	Archivo CLASS	1 KB
 City.class	22/03/2025 7:07 p. m.	Archivo CLASS	3 KB
 Item.class	22/03/2025 7:07 p. m.	Archivo CLASS	1 KB
 Person.class	22/03/2025 7:07 p. m.	Archivo CLASS	1 KB
 TrafficLight.class	22/03/2025 7:07 p. m.	Archivo CLASS	1 KB
 Walker.class	22/03/2025 7:07 p. m.	Archivo CLASS	1 KB
 Wallflower.class	22/03/2025 7:07 p. m.	Archivo CLASS	1 KB

Archivos de presentation:

 CityGUI\$1.class	22/03/2025 7:07 p. m.	Archivo CLASS	1 KB
 CityGUI.class	22/03/2025 7:07 p. m.	Archivo CLASS	2 KB
 PhotoCity.class	22/03/2025 7:07 p. m.	Archivo CLASS	3 KB

Los archivos que se crearon se ubicaron en la carpeta bin.

Documentando [En lab03.doc]

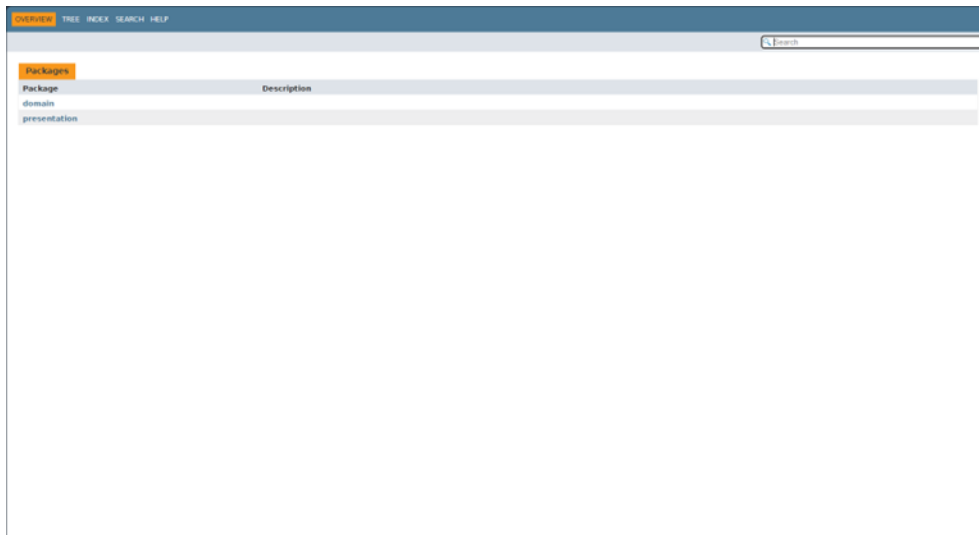
1. Utilizando el comando javadoc, desde el directorio raiz, genere la documentación (API) en formato html, en este directorio. ¿cuál es el comando completo para generar esta documentación?

El comando usado para generar la documentación fue el siguiente:

```
javadoc -d docs -cp src -subpackages presentation domain
```

2. ¿Cuál archivo hay que abrir para empezar a navegar por la documentación? Ábralo y capture la pantalla.

El archivo por abrir es el index.html ubicado en schelling\docs\index.html



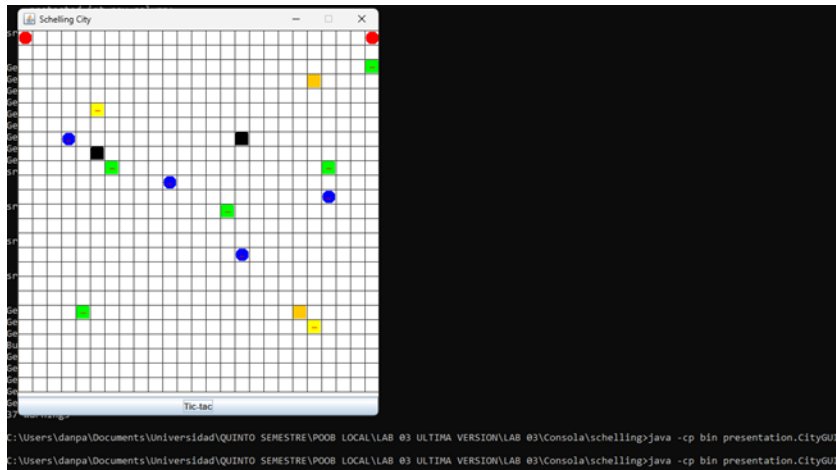
Ejecutando [En lab03.doc]

4. Empleando el comando java, desde el directorio raiz, ejecute el programa. ¿Cómo utilizó este comando?

Se empleo el siguiente comando:

```
java -cp bin presentation.CityGUI
```

donde ubicamos el CityGUI ubicado en presentación.









Probando [En lab03.doc]

1. Adicione ahora los archivos del directorio pruebas y trate de compilar nuevamente el programa. Tenga en cuenta que estas clases requieren la librería junit 4.8.

¿Cómo se incluye un paquete para compilar?

Se descargan las librerías necesarias y se agregan a un subdirectorio dentro de schelling que lo llamaremos “lib” y luego compilamos los paquetes y los test en una misma línea como se vera en el siguiente punto.

 apiguardian-api-1.1.2	22/03/2025 7:31 p. m.	Executable Jar File	7 KB
 junit-jupiter-api-5.9.2	22/03/2025 7:31 p. m.	Executable Jar File	203 KB
 junit-jupiter-engine-5.9.2	22/03/2025 7:31 p. m.	Executable Jar File	241 KB
 junit-platform-commons-1.9.2	22/03/2025 7:31 p. m.	Executable Jar File	101 KB
 junit-platform-console-standalone-1.9.2	22/03/2025 7:32 p. m.	Executable Jar File	2.554 KB
 junit-platform-engine-1.9.2	22/03/2025 7:31 p. m.	Executable Jar File	185 KB

¿Qué instrucción completa tuvo que dar a la consola para compilar?

Utilizamos la siguiente instrucción:

```
javac -d bin -cp "bin;libs/*" src/Test/*.java
```

2. Ejecute desde consola las pruebas.

¿Cómo utilizó este comando? Puede ver ejemplos de cómo ejecutar el “test runner” en How to run JUnit test cases from the command line

Se utilizó el comando de la siguiente manera:

Ubicados desde el Directorio raíz ejecutamos el siguiente comando, donde se entra en la carpeta donde guardamos las librerías (en este caso es el subdirectorio “libs”) y colocamos la versión del junit instalado, seguido a eso los siguientes comandos los cuales ejecutarán todas las pruebas.

```
java -jar libs/junit-platform-console-standalone-1.9.2.jar --classpath bin --scan-classpath
```

3. Pegue en su documento el resultado de las pruebas

```
C:\Users\danpa\Documents\Universidad\QUINTO SEMESTRE\POOB LOCAL\LAB 03 ULTIMA VERSION\LAB 03\consola\schelling>java -jar libs/junit-platform-console-standalone-1.9.2.jar --classpath bin --scan-classpath
Thanks for using JUnit! Support its development at https://junit.org/sponsoring

[36m--[0m +[36mJUnit Jupiter-[0m +[32m[OK]-[0m
[36m---[0m +[36mBeastTest-[0m +[32m[OK]-[0m
[36m|---[0m +[36mShouldCreateBeast-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestInitialColor-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTrafficLightTest-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestChangeColorYellow-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestColorCycle-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestInitialColor-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestChangeColorGreen-[0m +[32m[OK]-[0m
[36m|---[0m +[36mBuildingTest-[0m +[32m[OK]-[0m
[36m|---[0m +[36mShouldCreateBuilding-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestInitialColor-[0m +[32m[OK]-[0m
[36m---[0m +[36mJUnit Vintage-[0m +[32m[OK]-[0m
[36m---[0m +[36mWalkerTest-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWalkerDecideHappy-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWalkerChangeDissatisfied-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWalkerDecideIndifferent-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWalkerInitialization-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWalkerMove-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWalkerShape-[0m +[32m[OK]-[0m
[36m|---[0m +[36mWallflowerTest-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWallflowerChangeDissatisfied-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWallflowerShape-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWallflowerDecideDissatisfied-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWallflowerInitialization-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWallflowerMove-[0m +[32m[OK]-[0m
[36m|---[0m +[36mTestWallflowerDecideHappy-[0m +[32m[OK]-[0m
[36m---[0m +[36mJUnit Platform Suite-[0m +[32m[OK]-[0m

Test run finished after 124 ms
  8 containers found
  0 containers skipped
  8 containers started
  0 containers aborted
  8 containers successful
  0 containers failed
 20 tests found
  0 tests skipped
 20 tests started
  0 tests aborted
 20 tests successful
  0 tests failed
```

Empaquetando [En lab03.doc]

1. Consulte como utilizar desde consola el comando jar para empaquetar su programa entregable en un archivo .jar, que contenga los archivos bytecode necesarios (no las fuentes ni las clases de prueba), y que se pueda ejecutar al instalarlo en cualquier directorio, con solo tener la máquina virtual de java y su entorno de ejecución (JRE).

¿Cómo empaquetó jar?

El jar lo empaquetamos con el siguiente comando:

```
jar cfe schelling.jar presentation.CityGUI -C bin .
```

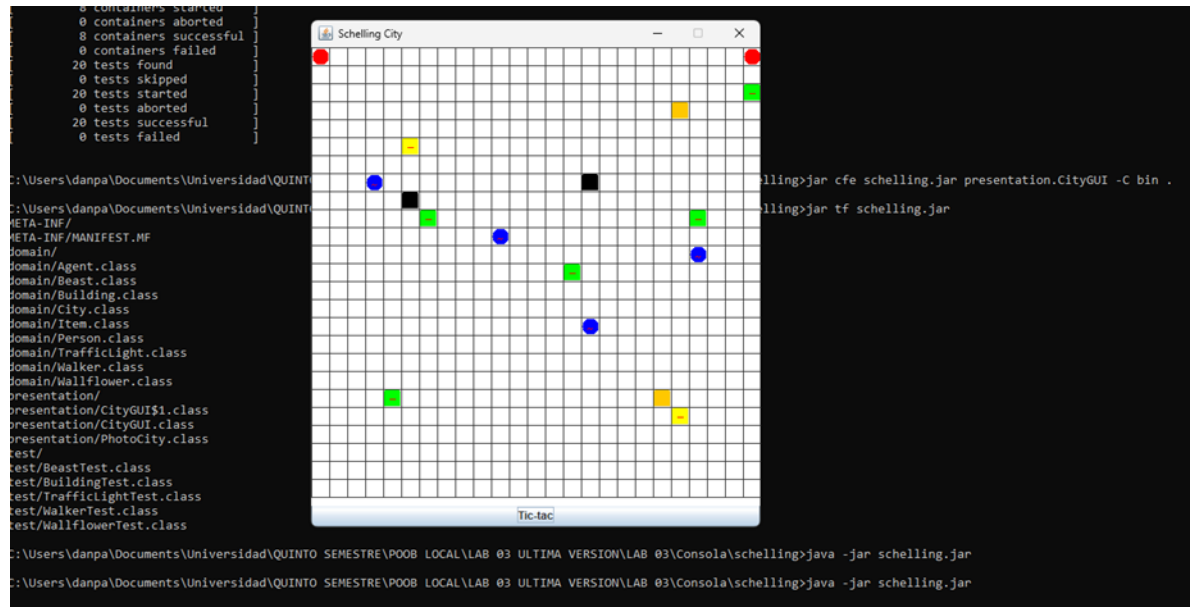
Estando previamente desde el directorio raíz.

2. ¿Cómo se ejecuta el proyecto empaquetado?

Desde el directorio raíz ejecutamos el siguiente comando:

```
java -jar schelling.jar
```

Aquí se nos abre el proyecto:



RETROSPECTIVA

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

El tiempo total invertido por cada uno fue de aproximadamente 16 horas por cada uno de nosotros.

2. ¿Cuál es el estado actual de laboratorio? ¿Por qué? (Para cada método incluya su estado):

El estado actual del laboratorio es completado ya que se realizó cada uno de los puntos asignados en su totalidad y consideramos que no quedo faltando nada en ninguno.

3. Considerando las prácticas XP del laboratorio de hoy ¿por qué consideran que son Importante?

Code must written to agreed standard: Para este laboratorio fue importante realizar para mantener los estándares de java, dejando una estructura de código clara, limpia y legible, donde es posible extensibilidad del código si se quieren agregar cosas nuevas en un futuro.

Code the unit test first: Para este laboratorio fue importante realizar primero las pruebas con el fin de tener una idea más clara de lo que queríamos hacer en cada ciclo, así tuvimos en cuenta los requisitos de cada ciclo y la funcionalidad del código, también para saber lo que, si tenía que funcionar y lo que no, lo que nos llevó al éxito del laboratorio.

4. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?

Nuestro mayor logro quizás fue la fluidez en la que trabajamos el laboratorio, puesto que sentimos que tuvimos más claridad en nuestros objetivos y llegamos más rápido al resultado.

Nuestro mayor problema quizás fue el manejo de java en el cmd, al ser algo nuevo, no teníamos quizás muy claros algunos comandos y conceptos por lo que tuvimos que investigar bastante sobre su funcionamiento.

5. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los

Resultados?

Como equipo, siempre tratamos de dividirnos el trabajo de manera equitativa donde podamos aprender los dos de manera igualada, también solemos trabajar en pair programming lo que nos ayuda a tener un mejor rendimiento a la hora de hacer el código, nos comprometemos a seguir mejorando la calidad de los laboratorios para que cada vez sea mejor la entrega de estos.

6. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares

Adecuados.

Utilizamos múltiples referencias las cuales las citaremos aquí abajo, la más útil fue la ia en la parte de pasar de bluej a consola puesto que al haber mucha información en las páginas no sabíamos cuál de todos los comandos utilizar por lo que esta nos ayudaba a tener más claro los comandos a utilizar.

Referencias bibliográficas:

Aprender a programar. (s.f.). *Compilar y ejecutar un programa Java: Uso de la consola DOS o CMD de Windows, invocar javac*. Recuperado el 16 de marzo de 2025, de https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=395

Abril Code. (s.f.). *Compilar código en Java desde la terminal*. Recuperado el 16 de marzo de 2025, de <https://programacion.abrilcode.com/doku.php?id=bloque1:compilarterminal>

WikiHow. (s.f.). *Cómo crear y eliminar archivos o carpetas desde el intérprete de comandos de Windows*. Recuperado el 16 de marzo de 2025, de <https://es.wikihow.com/crear-y-eliminar-archivos-o-carpetas-desde-el-int%C3%A9rprete-de-comandos-de-Windows>

El Guillemola. (s.f.). *Compilar código Java desde la línea de comandos*. Recuperado el 16 de marzo de 2025, de <https://www.elguillemola.com/compilar-codigo-java-desde-la-linea-de-comandos/>

ChatGPT. (2025). *Asistencia en generación de referencias bibliográficas en formato APA 7 y compilación de código Java desde la línea de comandos*. OpenAI.

