

**UNIVERSIDAD ESCUELA COLOMBIANA DE INGENIERÍA  
JULIO GARAVITO**

**LABORATORIO 3**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**NOMBRES:**

**CHRISTIAN ALFONSO ROMERO MARTINEZ  
ANDERSON FABIAN GARCIA NIETO**

**PROFESORA:  
MARIA IRMA DIAZ ROZO**

**14/03/25**

## CONOCIENDO

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

- a) ¿Cuántos paquetes tiene?

Tiene 2 paquetes, uno llamado domain y el otro presentation

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

El propósito del paquete presentación es presentar la interfaz al usuario, mostrar la simulación del paquete domain

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

El propósito de dominio es definir los actores y el funcionamiento que se va a presentar en el paquete presentación.

2. Revisen el paquete de dominio,

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

Se compone de 4 clases

Con una de ellas abstracta y otra correspondiente a una interfaz se importa en 3 de estas clase java.awt.Color y en una java.util.\*

La clase item, agent, person importan de java.awt.Color Mientras que la clase City importa de java.util.\*

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

### 4 clases

Para este caso Implica que tenemos 2 tipos de objetos, debido a que la clase abstracta y la interfaz no pueden instanciar objetos

### Clase abstracta

Nos quiere dar una plantilla para que una clase hija pueda hacer sobre escritura en un método y hacer una llamada para ejecución más simple.

### Clase interfaz

Muy similar a la clase abstracta nos permite instanciar métodos sobre los que hacemos overriding, en este caso todos son abstractos y públicos finales estáticos.

### java.awt.Color

Nos da todo lo necesario para hacer representaciones en el sistema, en este caso específicamente con el color.

### java.util

Nos ofrece un marco de colecciones donde incluso podemos instaurar fecha, o generar números aleatorios.

3. Revisen el paquete de presentación,

- a) ¿Cuántos componentes tiene?

Este paquete cuenta con 2 clases, pero están codificados en un único script, estas se componen por importar el paquete Dominio, y además javax.swing, java.awt, java.awt.event y java.io

**b) ¿Cuántos métodos públicos propios (no heredados) ofrece?**

Ofrece 4 métodos con esas características por las dos clases, principalmente en la clase CityGUI los métodos que cumplen con las características enunciadas son:

```
-gettheCity()  
-main(String[] args)
```

En cuanto a la clase PhotoCity, los metdos correspondientes son:

```
-PhotoCity(CityGUI gui)  
-paintComponent(Graphics g)
```

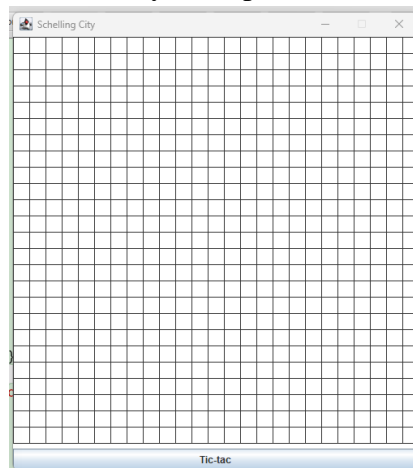
**4. Para ejecutar un programa en java, ¿Qué método se debe ejecutar? ¿En qué clase se encuentra?**

- a. Para ejecutar el código debemos usar el método main que se encuentra en CityGUI, siendo este porque es el método principal, llamando al constructor que está privado de la clase CityGUI.

**5. Ejecuten el programa.**

**a. ¿Qué funcionalidades ofrece?**

- i. Las funcionalidades que ofrece son las de crear un tablero con una cuadrícula y en la parte inferior un botón con Tic-tac



ii.

**b. ¿Qué hace actualmente?**

- i. Actualmente solo se instancia, ganando las características de tamaño, preparar elementos y preparar acciones y se vuelve visible.

**c. ¿Por qué?**

- i. No existe el método `photo.repaint()` y el método `ticTac` en la clase `City` no está definido por lo que al dar click no se hace nada dejándonos solo con la cuadrícula.

## 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.

`Package` tiene como significado “Paquete”, un paquete en Java es una agrupación de componentes lógicos usados en una aplicación, teniendo una relación entre estos, dichos componentes lógicos pueden ser distintas definiciones de clases, interfaces, archivos de texto entre otros componentes.

Su uso se basa en la estructuración y estructuración de clases relacionadas, estos ayudan a evitar conflictos de nombres, proporcionan una estructura de directorios lógica y permiten el control de acceso a clases y miembros.

Para poder hacer uso de los elementos de los paquetes es necesario importar su módulo de código en curso, para esto se usa la palabra `import` con significado al español “importar”

2. Revise el contenido del directorio de trabajo y sus subdirectorios. Describa su contenido. ¿Qué coincidencia hay entre paquetes y directorios?

-Dentro del directorio inicial “schelling-2025-1”, está lo siguiente:

◀ schelling-2025-1 2 elementos

Nombre	Última modificación	Tamaño del a...
 schelling	-	260 KB
 schelling_.asta	12 mar 2025	10 KB

Donde `schelling` es el directorio principal de trabajo, y el archivo `.asta` es el archivo donde se trabajará el diseño del trabajo.

-Dentro del subdirectorio “schelling”, están los siguientes archivos y subdirectorios:

◀ schelling 4 elementos

Nombre	Última modificación	Tamaño del a...
 doc	-	233 KB
 domain	-	13 KB
 presentation	-	14 KB
 package.bluej	12 mar 2025	799 bytes

Donde se encuentran los subdirectorios `doc`, `domain` y `presentation`, aparte de un archivo `.bluej`, siendo este último un proyecto que contiene los archivos que se incluyen en un proyecto de BlueJ.

-Dentro del subdirectorio domain se encuentran los siguientes archivos.

Agent.class	988	520	Archivo CLASS	12/03/2025 11:...	10A9C757
Agent.ctxt	887	323	Archivo CTXT	12/03/2025 11:...	F8043F2D
Agent.java	1.191	412	Archivo JAVA	12/03/2025 11:...	683CC56B
City.class	1.690	910	Archivo CLASS	12/03/2025 11:...	6816580A
City.ctxt	637	256	Archivo CTXT	12/03/2025 11:...	6152B300
City.java	1.407	500	Archivo JAVA	12/03/2025 11:...	C251B50C
Item.class	697	372	Archivo CLASS	12/03/2025 11:...	E9D352DA
Item.ctxt	419	195	Archivo CTXT	12/03/2025 11:...	4F1A80CC
Item.java	519	248	Archivo JAVA	12/03/2025 11:...	C76A9F69
package.bluej	1.317	434	BlueJ Project File	12/03/2025 11:...	70149A45
Person.class	1.145	637	Archivo CLASS	12/03/2025 11:...	5F2315BF
Person.ctxt	848	342	Archivo CTXT	12/03/2025 11:...	678D5AAC
Person.java	1.213	492	Archivo JAVA	12/03/2025 11:...	0C4C09D4

Donde los archivos son:

- .java:** Son archivos de código fuente escritos en el lenguaje de programación Java
- .class:** Son archivos compilados generados a partir de los archivos .java.
- .bluej:** Son archivos de configuración específicos de BlueJ.
- .ctxt:** Son archivos de contexto específicos de BlueJ.

-Dentro de presentation se encuentran los siguientes archivos.

presentation 8 elementos

Nombre	Última modificación	Tamaño del a...
CityGUI\$1.class	12 mar 2025	703 bytes
CityGUI.class	12 mar 2025	2 KB
CityGUI.ctxt	12 mar 2025	486 bytes
CityGUI.java	12 mar 2025	4 KB
package.bluej	12 mar 2025	605 bytes
PhotoAManufacturing.class	29 sept 2022	2 KB
PhotoAutomata.class	10 mar 2022	2 KB
PhotoCity.class	12 mar 2025	2 KB

Donde los archivos son:

- .java:** Son archivos de código fuente escritos en el lenguaje de programación Java
- .class:** Son archivos compilados generados a partir de los archivos .java.
- .bluej:** Son archivos de configuración específicos de BlueJ.
- .ctxt:** Son archivos de contexto específicos de BlueJ.

-Dentro de doc, se encuentran los siguientes subdirectorios y archivos.








aplicacion	31.238	6.517	Carpeta de archivos	6/03/2020 6:52 ...	
domain	62.597	13.257	Carpeta de archivos	12/03/2025 11:...	
presentation	11.602	3.180	Carpeta de archivos	10/04/2019 10:...	
presentation	64.133	15.430	Carpeta de archivos	12/03/2025 11:...	
resources	57	57	Carpeta de archivos	10/04/2019 10:...	
allclasses-frame....	1.233	541	Microsoft Edge HT...	12/03/2025 11:...	87A32137
allclasses-nofra...	1.113	529	Microsoft Edge HT...	12/03/2025 11:...	1D4A8D9B
constant-values....	6.972	1.521	Microsoft Edge HT...	12/03/2025 11:...	B360FE16
help-doc.html	7.854	2.338	Microsoft Edge HT...	12/03/2025 11:...	99FDF218
index.html	2.966	1.098	Microsoft Edge HT...	12/03/2025 11:...	E1610405
index-all.html	15.283	2.341	Microsoft Edge HT...	12/03/2025 11:...	35AF46BB
logfile.txt	4.497	718	Documento de tex...	12/03/2025 11:...	77B6C6C1
overview-frame....	931	506	Microsoft Edge HT...	12/03/2025 11:...	F9FF351A
overview-summ...	4.067	1.183	Microsoft Edge HT...	12/03/2025 11:...	FDE7F4A4
overview-tree.ht...	5.490	1.408	Microsoft Edge HT...	12/03/2025 11:...	E19F1D3C
package-list	20	20	Archivo	12/03/2025 11:...	6B72F79E
script.js	827	352	Archivo JavaScript	12/03/2025 11:...	82FF7475
serialized-form....	4.522	1.231	Microsoft Edge HT...	12/03/2025 11:...	5CF7A556
stylesheet.css	12.842	2.669	Documento de hoj...	12/03/2025 11:...	27695CE1

Donde:




- .html:** Una página HTML que lista todas las clases del proyecto en un marco (frame).
- .txt:** Un archivo de registro (log) que almacena mensajes generados durante la ejecución o generación de la documentación.
- .css:** Un archivo CSS que define el estilo visual de la documentación.
- .js:** Un archivo JavaScript que contiene scripts para mejorar la interactividad de la documentación.

-Dentro de los respectivos subdirectorios se encuentran diferentes archivos html, a excepción del subdirectorio resoruces que contiene un archivo .gif.







Nombre	Última modificación	Tamaño del a...
 Celula.html	27 sept 2013	18 KB
 package-frame.html	27 sept 2013	1 KB
 package-summary.html	27 sept 2013	6 KB
 package-tree.html	27 sept 2013	5 KB

Nombre	Última modificación	Tamaño del a...
 Agent.html	12 mar 2025	13 KB
 City.html	12 mar 2025	11 KB
 Item.html	12 mar 2025	11 KB
 package-frame.html	12 mar 2025	1 KB
 package-summary.html	12 mar 2025	5 KB
 package-tree.html	12 mar 2025	5 KB
 Person.html	12 mar 2025	15 KB


  

Nombre	Última modificación	Tamaño del a...
 package-frame.html	27 sept 2013	898 bytes
 package-summary.html	27 sept 2013	5 KB
 package-tree.html	27 sept 2013	5 KB

Nombre	Última modificación	Tamaño del a...
 CityGUI.html	12 mar 2025	20 KB
 package-frame.html	12 mar 2025	929 bytes
 package-summary.html	12 mar 2025	4 KB
 package-tree.html	12 mar 2025	5 KB
 PhotoAManufacturing.html	29 sept 2022	14 KB
 PhotoCity.html	12 mar 2025	18 KB

Nombre	Última modificación	Tamaño del a...
 inherit.gif	27 sept 2013	57 bytes

Donde:

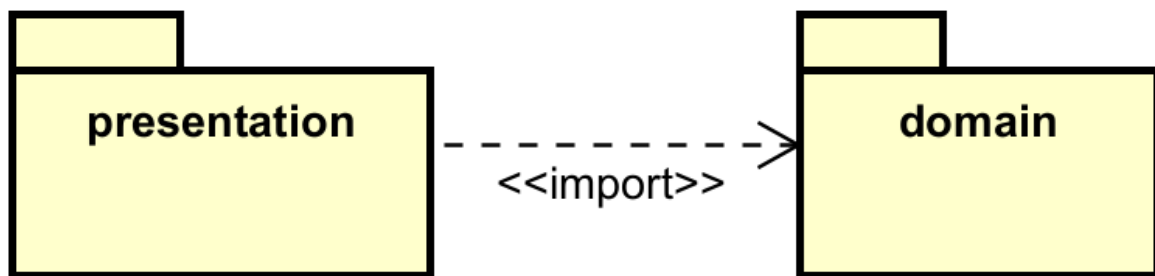
**.html:** Una página HTML que lista todas las clases del proyecto en un marco (frame).

**.gif:** En Java, los archivos de recursos (como imágenes) se suelen cargar en tiempo de ejecución para ser utilizados en la interfaz gráfica o en otras partes

del programa. Aquí te explico cómo podrías usar un archivo .gif en tu proyecto

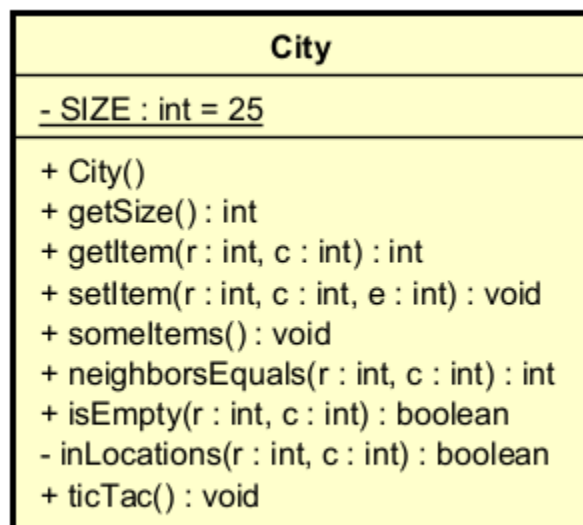
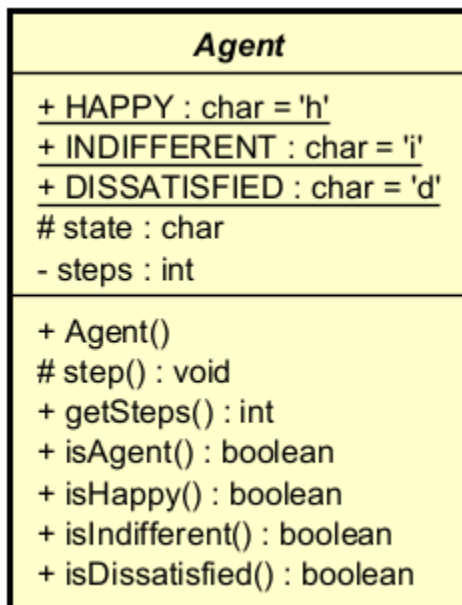
La coincidencia que hay entre los directorios y los paquetes, es que, el paquete presentation contiene a la clase CityGUI, y el paquete domain, contiene 3 clases (Agent, Person, City), donde la clase Agent es de tipo abstracto, y hay una interfaz que es Item.

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.



#### ARQUITECTURA DETALLADA

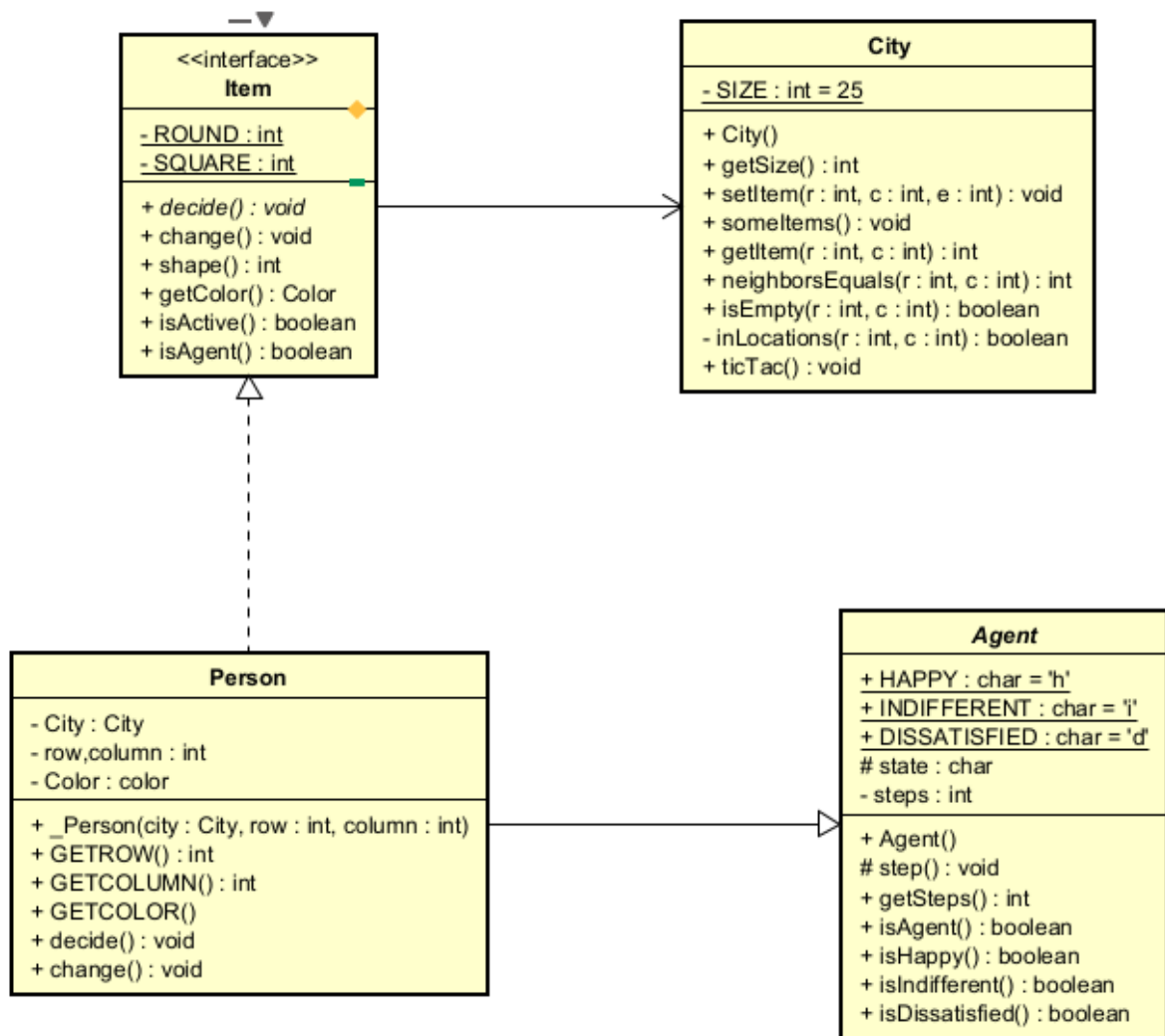
1. Para preparar el proyecto para BDD. Completen el diseño detallado del paquete de dominio. Adicionen el diagrama de clases en el paquete correspondiente.



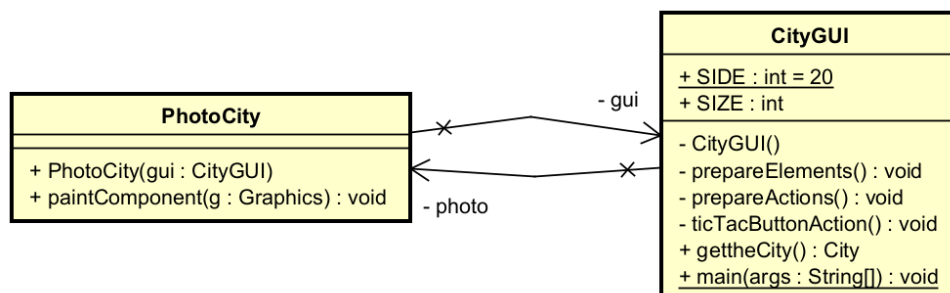
- a. ¿Qué componentes hacen falta?

Hacen falta componentes cruciales como lo son la clase Ítem y la clase Person.





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?

- Porque las 2 clases se escriben en un solo archivo, además en bluej solo se muestran las clases públicas y la clase **PhotoCity** no es pública.

3. Adicione la clase de pruebas unitarias necesaria para BDD en un paquete independiente de test. (No lo adicione al diagrama de clases)

a. ¿Qué paquete debe usar?

- Se debe usar el paquete test

**b. ¿Por qué?**

- i. Las pruebas se venen mantener separadas de el codigo para mantener una estructura limpia, y cumplir con los estándares de java

**c. ¿Asociado a qué clase?**

- i. Lo asociamos a la clase person

**d. ¿Por qué?**

- i. porque es una clase que se encuentra en el paquete de domain y utiliza todas las otras clases del proyecto, por lo que lo podemos intuir cómo clase principal y probar detalles como el registro en ciudad, estado inicial e incluso parte del funcionamiento

## **CICLO 1. INICIANDO CON LAS PERSONAS NORMALES**

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

El tipo de colección que usa para albergar cosas es un arreglo de arreglos, que se representa en el código de esta forma `ITEM[SIZE][SIZE]`

City si puede recibir personas.

Porque las personas están implementadas en la clase ítem/comportamiento de un ítem, donde city agrega nuevos ítems, de esta manera City sí pueden recibir personas.

**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.**

-El código asociado a la clase Person, es la clase Agent, clase padre de Person, el constructor de la clase Agent crea por defecto a las personas con un estado `DISSATISFIED` .

-La interfaz Ítem hay un método `shape()`, como persona tiene el comportamiento de un ítem y no sobreescribe el método, está la persona se va a crear con las características puestas en ese método `shape`, en este caso siendo `round`, haciendo que la persona sea un círculo.

-El tiempo aumenta cuando se llama al método `change()`, dentro de este mismo se llama al método `step` en la clase Agent, y este le aumenta la cantidad de pasos en uno.

-Las clases que definen a la clase persona son 2:

-Agent

-Item

Ya que la clase persona hereda de la clase Agent, y también persona tiene el comportamiento de un ítem

**3. Person por ser un Agent,**

**a. ¿Qué atributos tiene?**

- i. Los atributos extra que tiene person por ser un agente son los heredados, como:
  - protected char **state**
  - private int **steps**.

Los atributos propios de la clase Person son:

- private City **city**
- private int **row,column**;
- protected Color **color**;

**b. ¿Qué puede hacer (métodos)?**

- i. protected void step()
  - 1. Se encarga de incrementar los pasos.
- ii. public final int getSteps()
  - 1. Retorna el número de pasos, pero este no es modificable por que es final, además que tampoco es abstracto
- iii. isHappy(), isIndifferent(), isDissatisfied()
  - 1. Estos métodos se implementan en agente, y son los metodos de estado.
- iv. public final boolean isAgent()
  - 1. Este método devuelve true ya que la unica forma que se pueda acceder a este método, es que se sea un Agente

**c. ¿Qué decide hacer distinto?**

- i. Decide() es el método que lo diferencia puesto que le permite cambiar su estado entre Happy, Indifferent y Dissatisfied, además es propio

**d. ¿Qué no puede hacer distinto a todos los agentes?**

- i. Una persona no puede hacer diferente a los agentes las siguientes cosas:
  - El estado inicial que todas las personas se instancian con estado DISSATISFIED
  - El incremento de los de los pasos, debido a que el método está en Agent
  - Los métodos para consultar los estados, debido a que son finales, entonces no pueden ser sobreescritos.
  - Estar con un estado diferente a HAPPY, INDIFERENT, DISSATISFIED

**e. ¿Qué debe aprender a hacer? Justifique sus respuestas.**

- i. Debe aprender a decidir por sí misma, ya que como le enseña/hereda la clase no es suficiente para su comportamiento, por eso es que se sobreescribe, para que Person aprenda a decidir por si misma.

**4. Por comportarse como un Ítem,**

- a. **¿Qué sabe hacer?**
  - i. Person por comportarse con un Ítem, adquiere los comportamientos propuestos en la interfaz Ítem, por esto mismo puede hacer de nuevo los siguientes métodos:
    - Cambiar su estado, por medio del método decide()
    - Incrementar sus pasos, change()
    - Devolver su color, getColor()
    - Devolver su forma, shape()
    - Indicar si está activo, isActive()
    - Indicar si es un agente, isAgente()
- b. **¿Qué decide hacer distinto?**
  - i. La Person sobreescribe varios métodos, lo que decide hacer diferente es:
    - decide() que decide cambiar su estado en función de los pasos
    - getColor() que devuelve el color específico
    - change() que incrementar sus pasos al cambiar
- c. **¿Qué no puede hacer distinto?**
  - i. No altera la implementación de shape(), se queda con el valor por defecto de es activo en el método isActive() y no se considera un agente con el metodo isAgent(), todo esto porque en la clase Person no se sobreescriben estos metodos.
- d. **¿Qué debe aprender a hacer? Justifique sus respuestas.**
  - i. Debe aprender decide() en este caso, un método abstracto definido en la interfaz, por lo que la clase Person se encarga de implementarlo y redefinirlo. Por otro lado, el método change() en la clase Item no está implementado, mientras que en Person sí lo está, y su implementación consiste en llamar al método step()

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

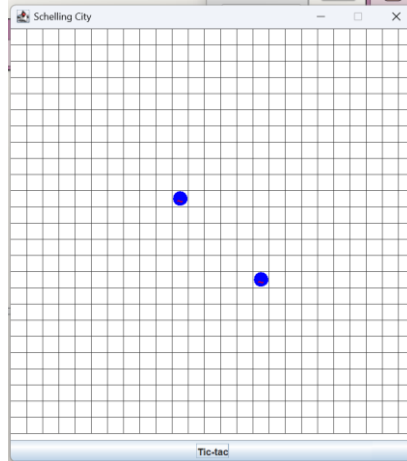
- a. Principalmente la persona cambia la cantidad de pasos dados y después decide que hacer, esto debido a que debe analizar a sus vecinos, entonces cuando se mueve y ya ha decidido a que estado cambiarse, se puede reiniciar el ciclo, a través de los ticks.

En relación con el código, primero el usuario da click en el botón Tick, después la ciudad llama al método tick(), que hace que todas las personas actúen, por último cuando cada persona está actuando, se llaman a los métodos change() y decide() que incrementan los pasos y cambian su estado en función de sus pasos, y por último se actualiza la interfaz gráfica

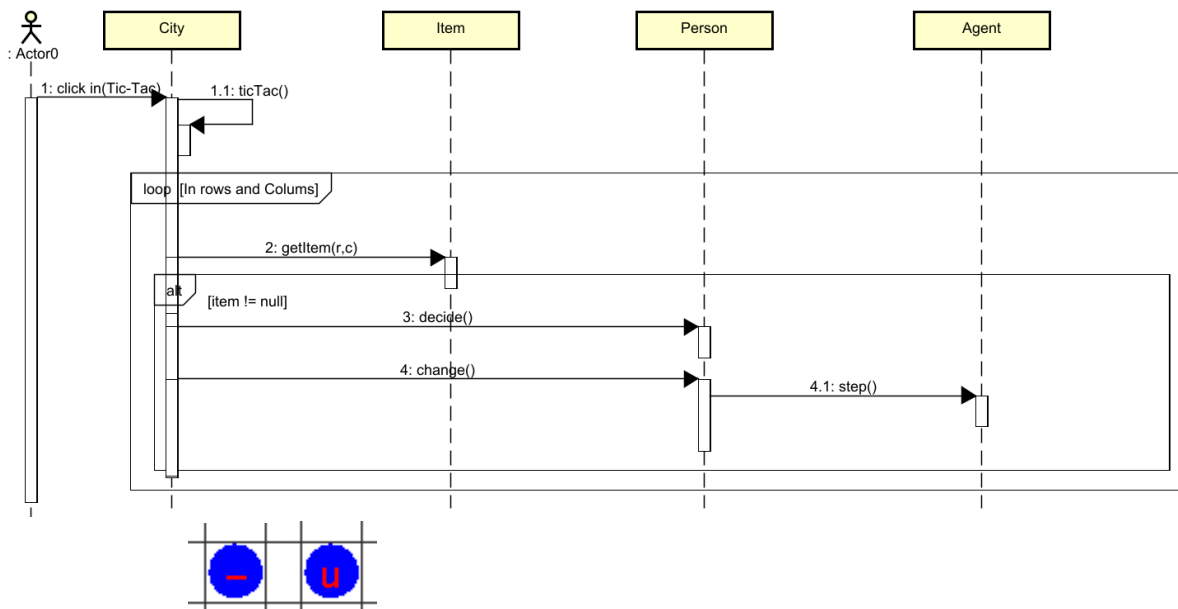
## 6. Ahora vamos a crear dos personas en diferentes posiciones (10,10) (15,15) llámelas adán y eva usando el método someItems() . Ejecuten el programa,

- a. **¿Qué pasa con las personas?**

- i. Aparecen las personas como insatisfechas a pesar que step es 0 y  $0\%3=0$  que es feliz
- ii. **¿Por qué?**
  1. Esto es porque el agente se inicia insatisfecho.
- iii. **Capturen una pantalla significativa.**








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



Al dar clic en Tic-Tac llamamos a decide() y change() por lo que podemos evidenciar alteraciones en su estado y en pasos.

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

- a. 1 =  Insatisfecho

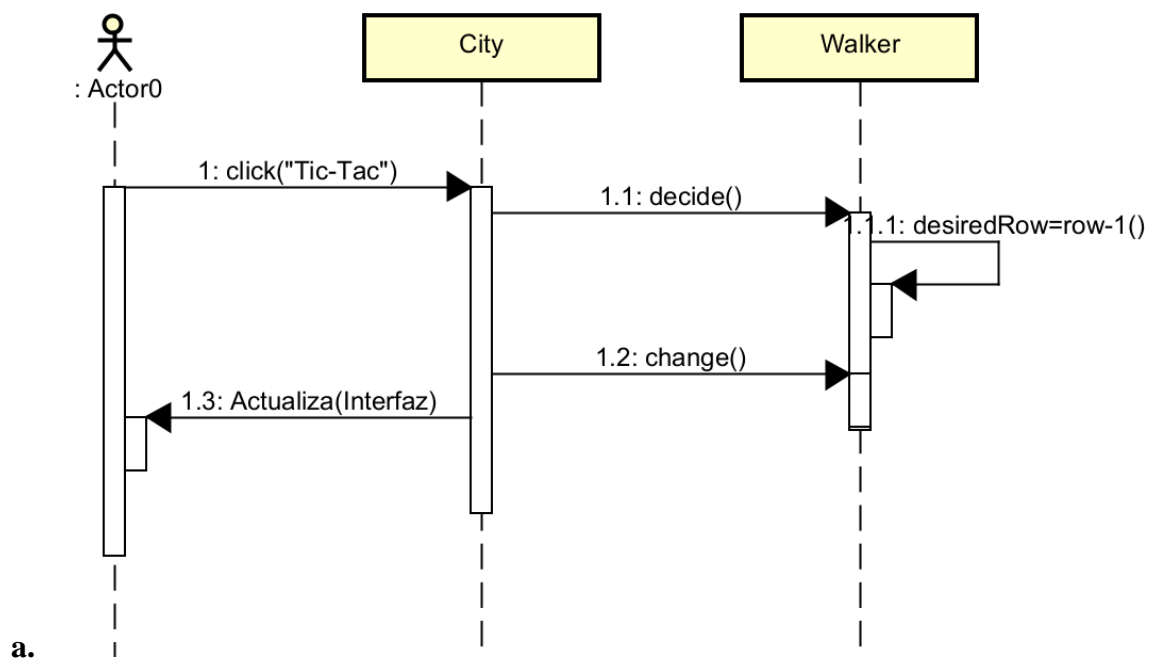
- b. 2 =  Happy
- c. 3 =  Indiferente
- d. 4 =  Insatisfecho
- e. 5 =  Happy
- f. ¿Es correcto?
- No del todo puesto que toma la decisión sin una validación clara del entorno y de igual forma cambia su estado a pesar de no tener cambios en su posición ni vecinos

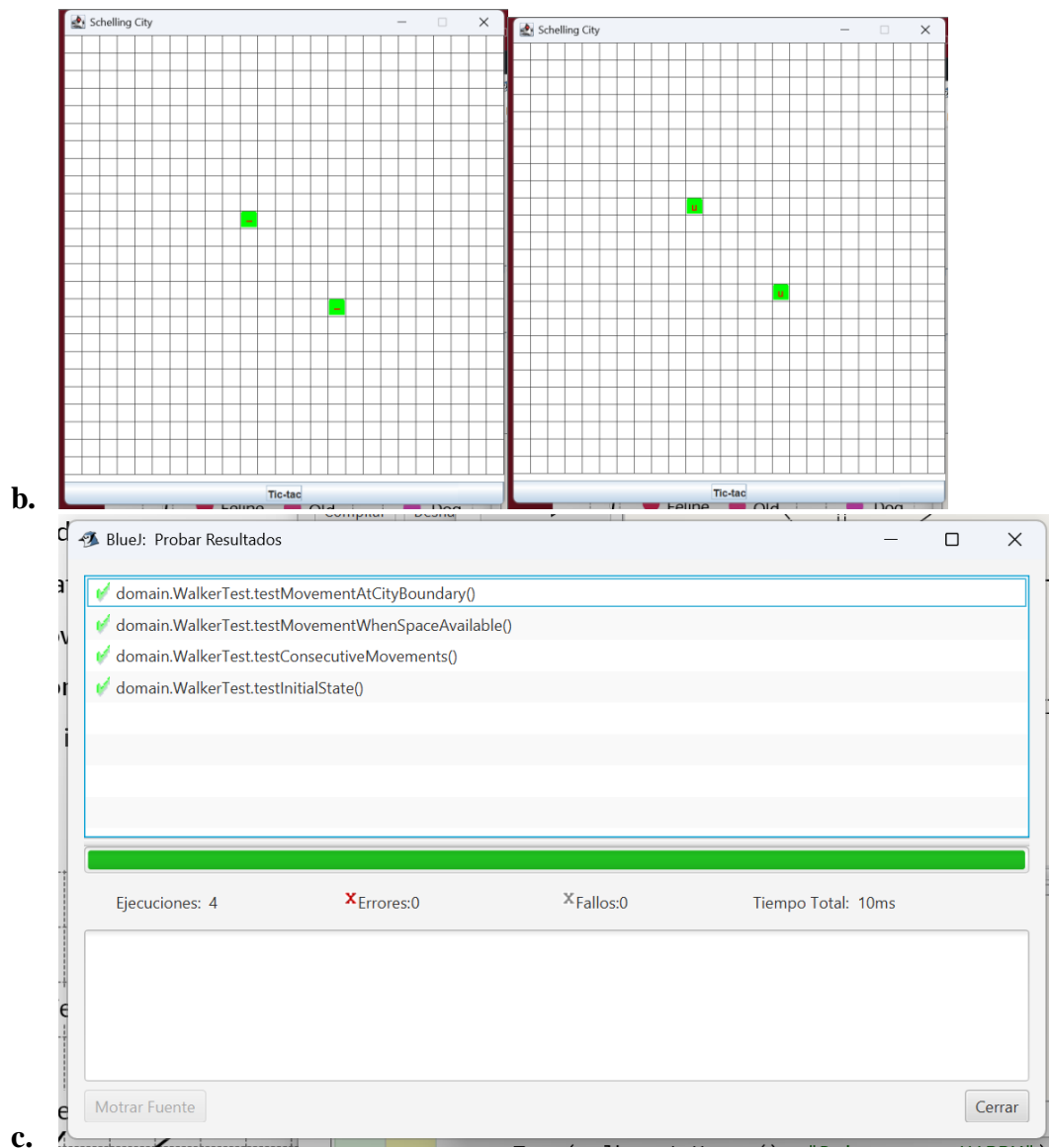
## CICLO 2. INCLUYENDO A LOS CAMINANTES

### 1. Para implementar esta nueva persona Walker ¿Cuáles métodos se sobrescriben (overriding)?

- decide()
  - Para poder determinar si se mueve y actualizar su estado
- change()
  - Para validar si al moverse se encuentra la posición disponible
- shape()
  - Para tomar la figura de rectángulo
- getColor()
  - Para ser visible con el color verde

### 2. Diseñen, construyan y prueben esta nueva clase. (Mínimo dos pruebas de unidad)





3. **Adicione una pareja de caminantes, llámelas messner y kukuczka,**  
(a) **¿Cómo quedarían después de tres Tic-tac? Ejecuten el programa y hagan tres clics en el botón. Capturen una pantalla significativa.**



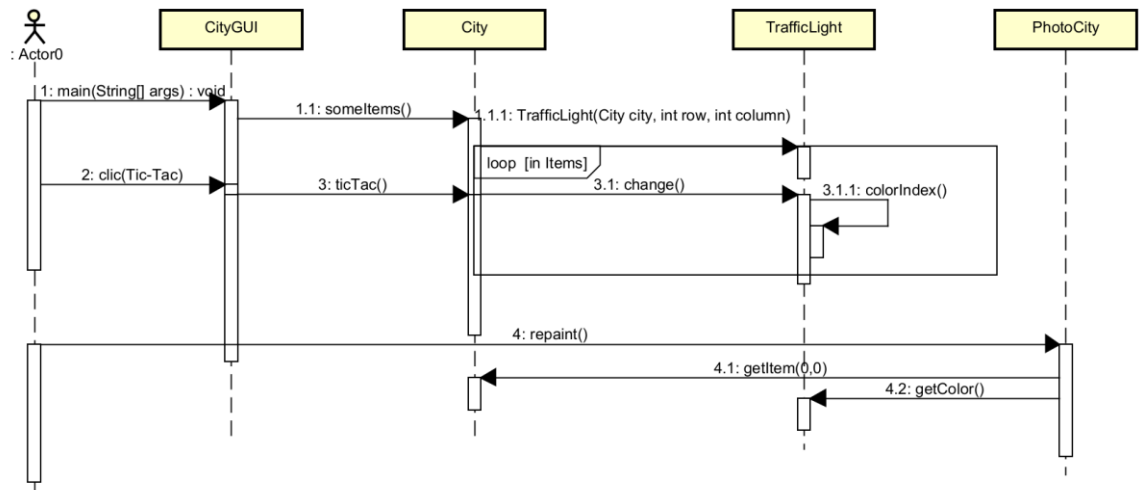
**(b) ¿Es correcto?**

Funciona correctamente, puesto que van moviéndose hacia arriba como lo previsto y además cambió su estado luego del primer click.

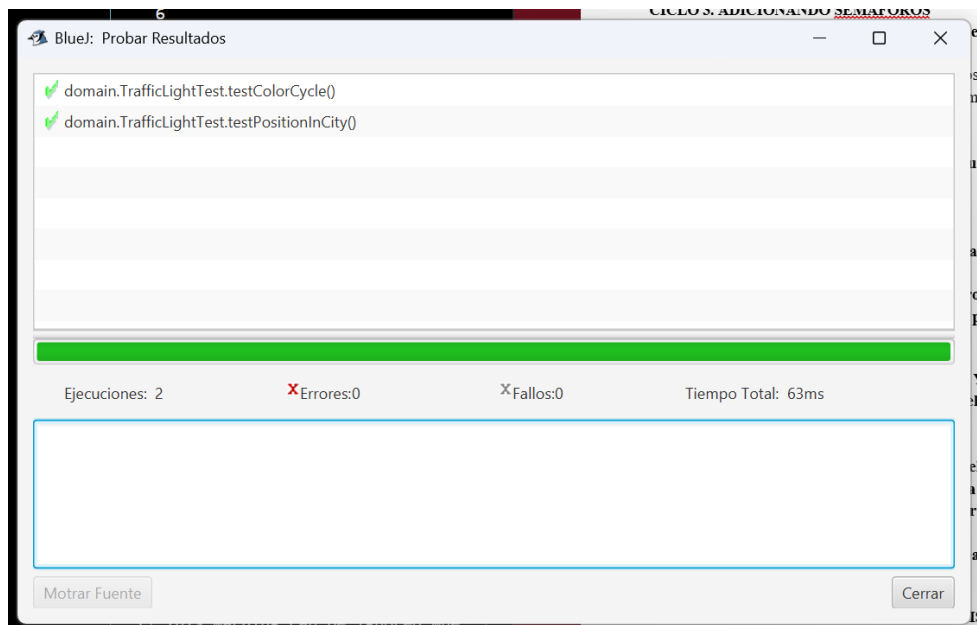
### **CICLO 3. ADICIONANDO SEMÁFOROS**

- 1. Para poder adicionar semáforos, ¿Debe cambiar el código de City en algo? ¿Por qué?**
  - a. En la idea de solución no debemos hacer cambios en City, esto porque hicimos una clase de semáforo que implementa la clase Ítem, y al tener el comportamiento de un ítem pueden entrar los semáforos a la ciudad
- 2. Diseñen , construyan y prueben esta nueva clase. (Mínimo dos pruebas de unidad)**





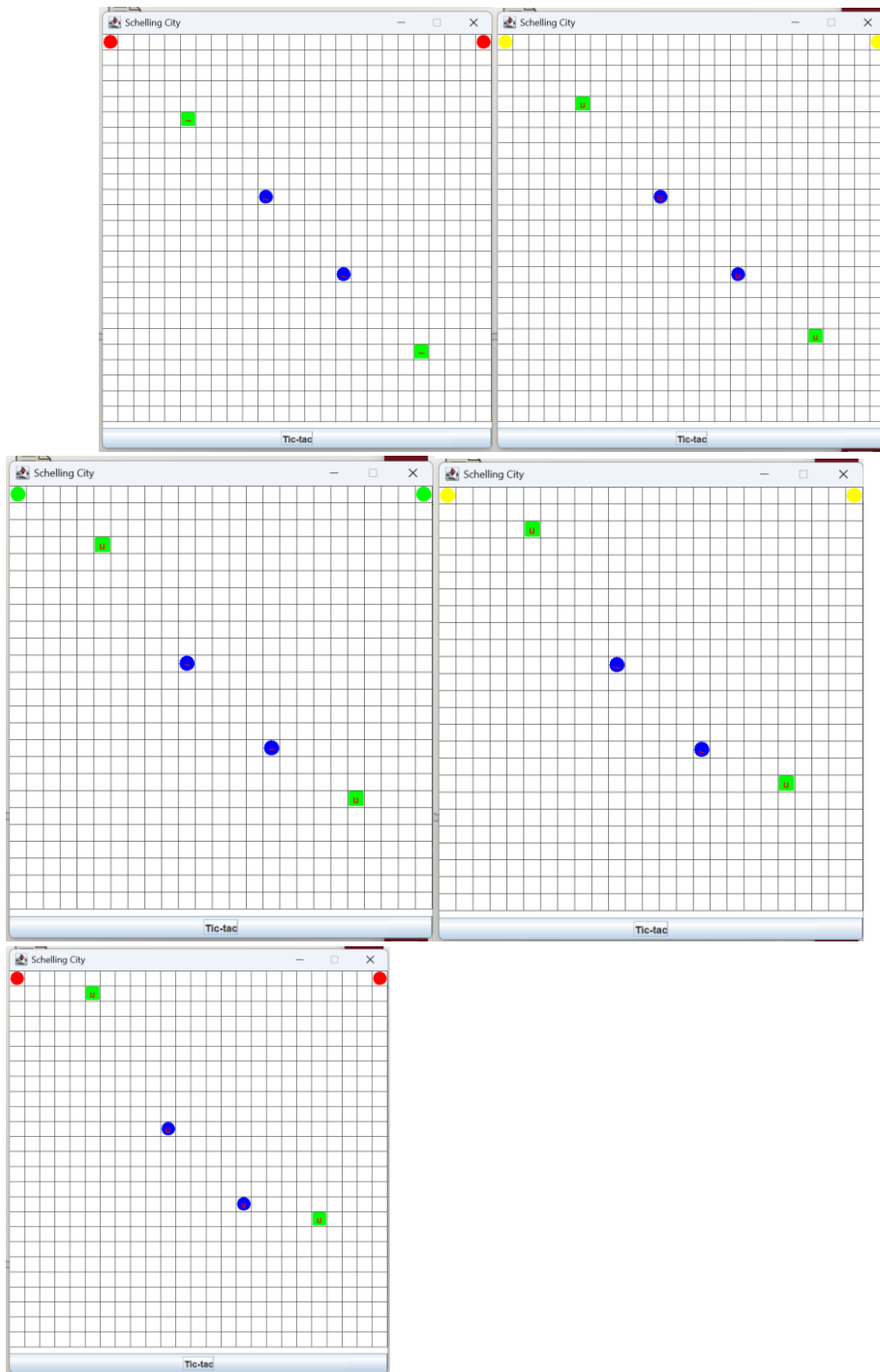
a.



b.

3. Adicionen dos semaforos en las esquinas superiores de la ciudad, llámenlos alarm y alert

(a) ¿Cómo quedarían después de cuatro Tic-tac? Ejecuten el programa y hagan cuatro clics en el botón. Capturen una pantalla significativa.



**(b) ¿Es correcto?**

Efectivamente es correcto, el semáforo cambia con cada clic los colores, como debería.

#### **CICLO 4. NUEVA PERSONA: PROPONIENDO Y DISEÑANDO**

##### **1. Propongan, describan e implementen el nuevo tipo de persona. (Mínimo dos pruebas de unidad)**

- Se propone crear una CHEERFUL que tiene como característica ser una persona alegre que genera alegría en su entorno, esta característica promueve la felicidad del vecindario, permitiendo que todos sus vecinos esten felices.

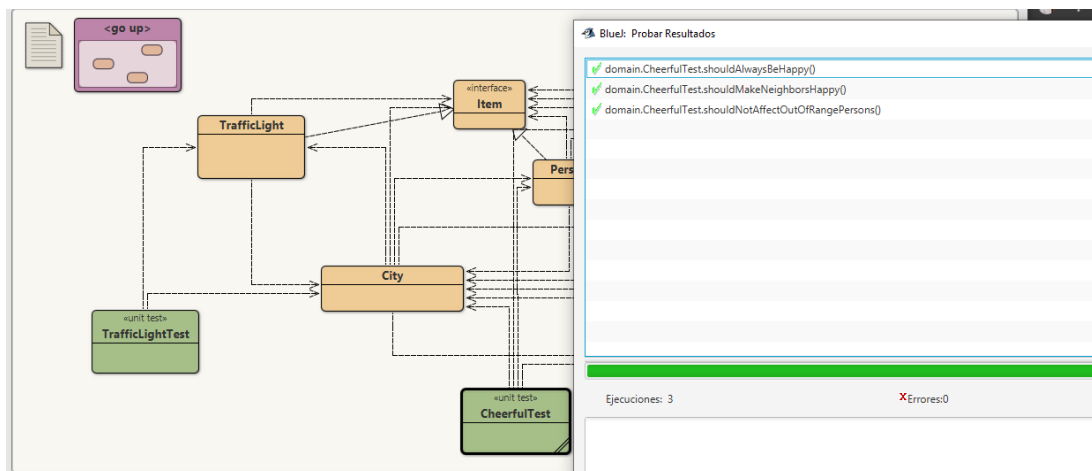
- b. Las dos pruebas de unidad que se pensaron es que cheerful person siempre este feliz, y también que haga feliz a sus vecinos

**2. Considerando una pareja de ellas con el apellido de ustedes.**

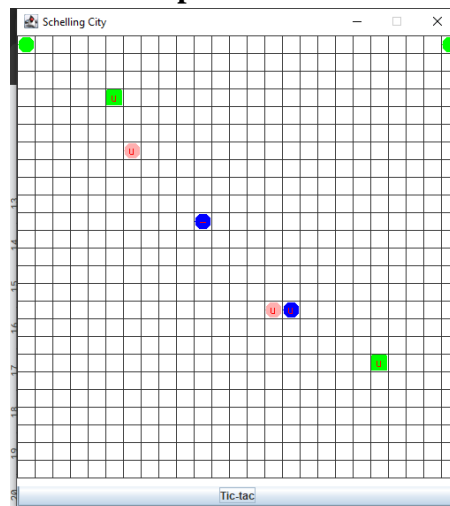
**(a) Piensen en otra prueba significativa y expliquen la intención.**

La nueva prueba de unidad tiene la intención de verificar que Cheerful solo afecta a los vecinos dentro de su rango de 3x3 y asegurar que no cambia el estado de personas que están fuera de su rango de radiación, garantizando que su influencia se limite únicamente a las posiciones adyacentes.

**(b) Codifiquen la prueba de unidad correspondiente y capturen la pantalla de resultados de ejecución de la prueba.**



**c) Ejecuten el programa con esa prueba como prueba de aceptación y capturen las pantallas correspondientes.**



**CICLO 5. NUEVO ÍTEM: PROPONIENDO Y DISEÑANDO**

**1. Propongan, describan e implementen el nuevo tipo de ítem. (Mínimo dos pruebas de unidad)**

El nuevo ítem `StreetLight` es un poste de luz que simula el comportamiento de una luz que se enciende y apaga cada tres turnos, utilizando dos colores para representar sus estados: `Color.DARK\_GRAY` para el estado "apagado" y `Color.ORANGE` para el estado "encendido". Hereda de `TrafficLight` y reutiliza su lógica básica, pero

personaliza el comportamiento de cambio de estado (`change()`) y los colores (`getColor()`). Cada tres turnos, el poste de luz alterna entre encendido y apagado. Este ítem no es un agente (`isAgent()` devuelve `false`) y siempre está activo (`isActive()` devuelve `true`).

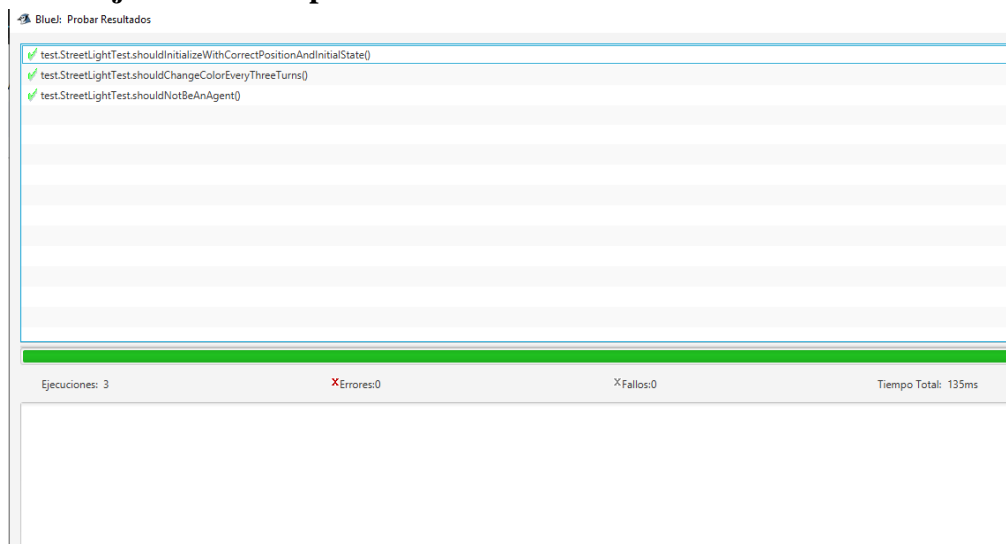
Para el caso de las pruebas, la primera se hizo con la intención de verificar que el `streetLight` se prendiera y apagara cada 3 turnos, y en cuanto a la segunda para verificar que no es un agente.

## 2. Considerando un par de ellos con el nombre de ustedes.

### (a) Piensen en otra prueba significativa y expliquen la intención.

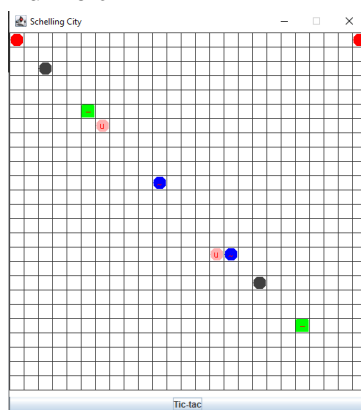
La otra prueba se pensó para probar que el `streetlight` se inicializa correctamente, también para que se cree en estado inactivo, con esto nos referimos a que empiece con la luz apagada.

### (b) Codifiquen la prueba de unidad correspondiente y capturen la pantalla de resultados de ejecución de la prueba.

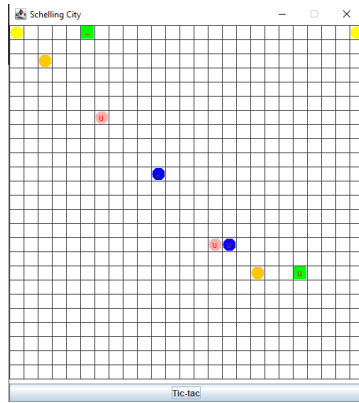


### (c) Ejecuten el programa con esa prueba como prueba de aceptación y capturen las pantallas correspondientes.

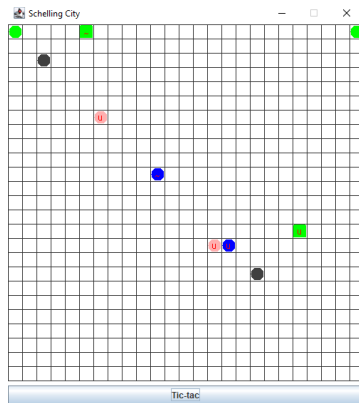
#### Turno 0



#### Turno 3



### Turno 6



### CICLO 6. BONO.

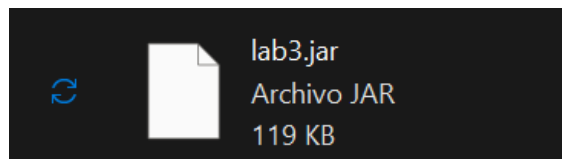
**La persona Schelling se rigue por las siguientes reglas:**

- Las personas Schelling sólo se mueven si están insatisfechos con su vecindad.
- Cuando están bien donde están, las personas Schelling no se mueven.
- Están indiferentes si todos los vecinos son como ellos o no tienen vecinos.
- Están insatisfechos si menos de 1/3 de los vecinos no son como ellos.
- Están satisfechos si más de 1/3 de los vecinos son como ellos y no todos son como ellos.

### EMPAQUETANDO LA VERSIÓN FINAL PARA EL USUARIO

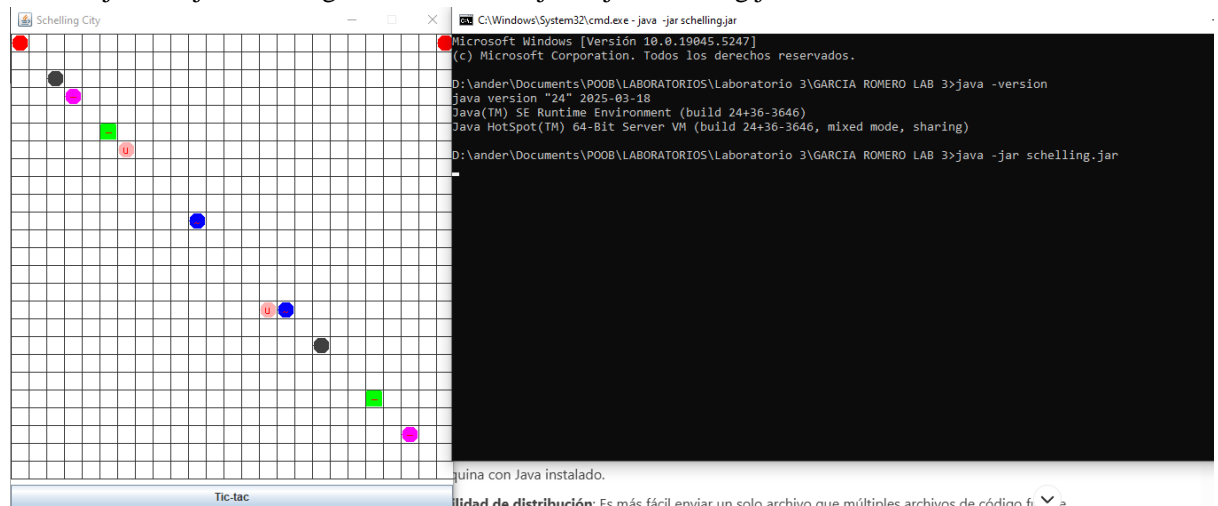
1. **Revise las opciones de BlueJ para empaquetar su programa entregable en un archivo .jar. Genere el archivo correspondiente.**

Usamos la opción Crear Archivo Jar, allí podemos definir la clase principal, e incluso incluir fuente y archivos de proyecto bluej



2. **Consulte el comando java para ejecutar un archivo jar. ejecutenlo ¿Qué pasa?**

Lo primero que se hizo fue instalar java, después de eso en el directorio donde se ubicaba el archivo jar, se ejecutó el siguiente comando: `java -jar schelling.jar`



### 3. ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.

**Portabilidad:** Un archivo .jar incluye todos los archivos necesarios para ejecutar el programa en cualquier computadora que tenga Java instalado, sin necesidad de configuraciones adicionales.

**Facilidad de distribución:** En lugar de compartir múltiples archivos de código fuente y dependencias, se puede distribuir un solo archivo .jar, lo que simplifica la instalación y el uso del programa.

**Protección del código fuente:** El .jar solo contiene archivos compilados (.class), lo que impide que el código original en Java sea visible o modificado fácilmente.

## DE BLUEJ A CONSOLA

En esta sección del laboratorio vamos a aprender a usar java desde consola. Para esto se va a trabajar con el proyecto del punto anterior.

## COMANDOS BÁSICOS DEL SISTEMA OPERATIVO

1. Investiguen los comandos para moverse en la estructura de directorios: **crear, borrar, listar su contenido y copiar o eliminar un archivo**[2].

Función	Comando	Estructura
Crear un directorio	mkdir	mkdir NombreDelNuevoDirectorio
Borrar un directorio vacío	rmdir	rmdir NombreDelDirectorio

<b>Borrar un directorio con archivos/subdirectorios</b>	rm -r	rm -r NombreDelDirectorio
<b>Listar contenido de un directorio</b>	dir	dir (Lista el contenido del directorio actual)
<b>Copiar directorios y los archivos que tengan</b>	cp -r	cp -r Directorio_copiar Nuevo_directorio
<b>Copiar archivos</b>	copy	copy Archivo_origen Archivo_Destino
<b>Eliminar un archivo</b>	del	del Nombre_archivo del Nombre_directorio
<b>Acceder a un directorio</b>	cd	cd Nombre_Directorio
<b>Copiar todos los archivos del directorio con un mismo tipo</b>	xcopy	xcopy *.tipo Directorio_copiar Nuevo_directorio

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**

The image shows a Windows command prompt window and a File Explorer window. The command prompt shows the following commands and their outputs:

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3>mkdir Nuevo_Directorio
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3>rmdir Nuevo_Directorio
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3>rmdir schelling
El sistema no puede encontrar el archivo especificado.
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3>mkdir schelling
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3>cd schelling
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling>mkdir src
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling>cd src
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling\src>mkdir domain
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling\src>mkdir presentation
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling\src>mkdir test
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling\src>

```

The File Explorer window shows the directory structure:

```

Este equipo > Disco local (D:) > ander > Documentos > POOB > LABORATORIOS > Laboratorio 3 > GARCIA ROMERO LAB 3 > schelling > src

```

Nombre	Fecha de modificación	Tipo	Tamaño
domain	22/03/2025 4:38 p. m.	Carpeta de archivos	
presentation	22/03/2025 4:39 p. m.	Carpeta de archivos	
test	22/03/2025 4:39 p. m.	Carpeta de archivos	

3. **En el directorio copien únicamente los archivos \*.java del paquete de aplicación. Consulte y capture el contenido de src/domain**

El paquete de la aplicación se compone del domain que es la lógica de esta, y presentaciones que es la interfaz

Domain

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling-hastaciclo6-copy\domain>xcopy /Y "*.java" "D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling\src\domain"
D:Agent.java
D:Cheerful.java
D:City.java
D:Item.java
D:Person.java
D:SchellingPerson.java
D:StreetLight.java
D:TrafficLight.java
D:Walker.java
9 archivo(s) copiado(s)

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling-hastaciclo6-copy\domain>_
```

Este equipo > Disco local (D:) > ander > Documentos > POOB > LABORATORIOS > Laboratorio 3 > GARCIA ROMERO LAB 3 > schelling > src > domain

Nombre	Fecha de modificación	Tipo	Tamaño
Agent.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
Cheerful.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
City.java	22/03/2025 3:22 p. m.	Archivo JAVA	5 KB
Item.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
Person.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
SchellingPerson.java	22/03/2025 3:22 p. m.	Archivo JAVA	5 KB
StreetLight.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
TrafficLight.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
Walker.java	22/03/2025 3:22 p. m.	Archivo JAVA	4 KB

Presentation

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling-hastaciclo6-copy\presentation>xcopy /Y "*.java" "D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling\src\presentation"
D:CityGUI.java
1 archivo(s) copiado(s)

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling-hastaciclo6-copy\presentation>_
```

Documentos > POOB > LABORATORIOS > Laboratorio 3 > GARCIA ROMERO LAB 3 > schelling > src > presentation

Nombre	Fecha de modificación	Tipo	Tamaño
CityGUI.java	22/03/2025 3:22 p. m.	Archivo JAVA	6 KB

Estructura de proyectos java

En java los proyectos se estructuran considerando tres directorios básicos.

schelling

src



**bin**

**docs**

**1. Investiguen los archivos que deben quedar en cada una de esas carpetas y la organización interna de cada una de ellas.**

**SCR(Fuente):** En este se guarda el código/archivos fuente como lo son los archivos .java,

src/domain/

src/presentation/

src/test/

**bin(binarios o compilados):** Se almacenan los archivos compilados, por ejemplo los .class.

Después de compilar el proyecto Java, los archivos .class (bytecode) resultantes se almacenan en este directorio.

bin/domain/

bin/presentation/

bin/test/

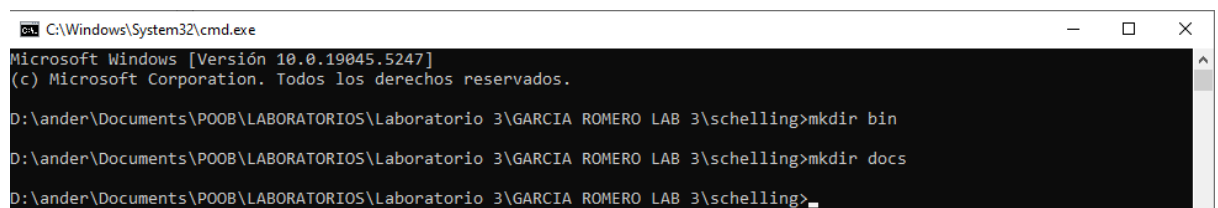
**docs(documentación):** En este se guarda la documentación, manuales, diagramas entre otros, con relación al proyecto.

También puede contener la documentación generada automáticamente, como la documentación JavaDoc.

**2. ¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué?**

**Cópielos y consulte y capture el contenido del directorio que modificó.**

Como aún no se tiene el directorio bin y docs, se crean:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling>mkdir bin
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling>mkdir docs
D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling>_
```

Los archivos que se deberían copiar del proyecto original al directorio bin, son los archivos .class. Los .ctxt no se incluyen porque no son parte de la convención estándar de estructura de proyectos Java.

**Domain**

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling-hastaciclo6-copy\domain>xcopy /Y "*.class" "D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling\bin\domain"
D:Agent.class
D:Cheerful.class
D:City.class
D:Item.class
D:Person.class
D:SchellingPerson.class
D:StreetLight.class
D:TrafficLight.class
D:Walker.class
9 archivo(s) copiado(s)

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling-hastaciclo6-copy\domain>
```

Este equipo > Disco local (D:) > ander > Documentos > POOB > LABORATORIOS > Laboratorio 3 > GARCIA ROMERO LAB 3 > schelling > bin > domain				
	Nombre	Fecha de modificación	Tipo	Tamaño
★	Agent.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB
★	Cheerful.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB
★	City.class	22/03/2025 3:22 p. m.	Archivo CLASS	3 KB
★	Item.class	22/03/2025 3:22 p. m.	Archivo CLASS	1 KB
★	Person.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB
	SchellingPerson.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB
DL	StreetLight.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB
	TrafficLight.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB
	Walker.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB

## Comandos de java

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

#### javac

Este comando es el compilador de java En esencia hace la conversión de archivos .java(archivo fuente) en .class(bytecode)

#### java

Ejecuta las aplicaciones java usa un .class(bytecode) para dar inicio al programa.

#### javadoc

Genera la documentación en HTML, tomando en cuenta los comentarios en el proyecto

#### jar

Crea, edita y extrae archivos de tipo jar

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

Comandos para consultar son: java -help y javac -help

```
C:\Users\Chris>javac -help
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
  -endoredsdirs <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
```

```
C:\Users\Chris>java -help
Usage: java [options] <mainclass> [args...]
       (to execute a class)
or java [options] -jar <jarfile> [args...]
       (to execute a jar file)
or java [options] -m <module>/<mainclass> [args...]
       java [options] --module <module>/<mainclass> [args...]
       (to execute the main class in a module)
or java [options] <sourcefile> [args]
       (to execute a single source-file program)

Arguments following the main class, source file, -jar <jarfile>,
-m or --module <module>/<mainclass> are passed as the arguments to
main class.

where options include:

  -cp <class search path of directories and zip/jar files>
  -classpath <class search path of directories and zip/jar files>
  --class-path <class search path of directories and zip/jar files>
        A ; separated list of directories, JAR archives,
        and ZIP archives to search for class files.
  -p <module path>
  --module-path <module path>...
        A ; separated list of elements, each element is a file path
        to a module or a directory containing modules. Each module is either
        a modular JAR or an exploded-module directory.
  --upgrade-module-path <module path>...
        A ; separated list of elements, each element is a file path
        to a module or a directory containing modules to replace
        upgradeable modules in the runtime image. Each module is either
        a modular JAR or an exploded-module directory.
  --add-modules <module name>[,<module name>,...]
        root modules to resolve in addition to the initial module.
        <module name> can also be ALL-DEFAULT, ALL-SYSTEM,
        ALL-MODULE-PATH.
  --enable-native-access <module name>[,<module name>...]
```

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

Comandos para conocer la version que corresponden a cada uno son: javac -version y java -version

```
C:\Users\Chris>javac -version
javac 21.0.6

C:\Users\Chris>java -version
java version "21.0.6" 2025-01-21 LTS
Java(TM) SE Runtime Environment (build 21.0.6+8-LTS-188)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.6+8-LTS-188, mixed mode, sharing)
```

Las versiones encontradas fueron:

- javac 21.0.6
- java 21.0.6

## Compilando

1. Utilizando el comando javac, desde el directorio raiz (desde schelling con una sólo instrucción), compile el proyecto. ¿Qué instrucción completa tuvo que dar a la consola para compilar TODO el proyecto? Tenga presente que se pide un único comando y que los archivos compilados deben quedar en los directorios respectivos.

El comando que funciono fue el siguiente javac -d bin src/domain/\*.java  
src/presentation/\*.java

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling>javac -d bin src/domain/Agent.java src/
domain/Cheerful.java src/domain/City.java src/domain/Item.java src/domain/Person.java src/domain/SchellingPerson.java sr
c/domain/StreetLight.java src/domain/TrafficLight.java src/domain/Walker.java src/presentation/CityGUI.java

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling>javac -d bin src/domain/*.java src/pres
entation/*.java

D:\ander\Documents\POOB\LABORATORIOS\Laboratorio 3\GARCIA ROMERO LAB 3\schelling>_
```

En la imagen salen dos diferentes formas de compilar el proyecto, la forma larga, archivo por archivo de tipo .java y la última que es la forma general.

Se usa -d bin para que los archivos .class generados se guardan en los directorios respectivos y no junto a los directorios que contienen a los archivos .java

## 2. Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles nuevos archivos aparecen ahora y dónde se ubican?

### Domain

Documentos > POOB > LABORATORIOS > Laboratorio 3 > GARCIA ROMERO LAB 3 > schelling > bin > domain

Nombre	Fecha de modificación	Tipo	Tamaño
Agent.class	22/03/2025 7:20 p. m.	Archivo CLASS	1 KB
Cheerful.class	22/03/2025 7:20 p. m.	Archivo CLASS	2 KB
City.class	22/03/2025 7:20 p. m.	Archivo CLASS	2 KB
Item.class	22/03/2025 7:20 p. m.	Archivo CLASS	1 KB
Person.class	22/03/2025 7:20 p. m.	Archivo CLASS	2 KB
SchellingPerson.class	22/03/2025 7:20 p. m.	Archivo CLASS	2 KB
StreetLight.class	22/03/2025 7:20 p. m.	Archivo CLASS	1 KB
TrafficLight.class	22/03/2025 7:20 p. m.	Archivo CLASS	2 KB
Walker.class	22/03/2025 7:20 p. m.	Archivo CLASS	2 KB

### Presentation

Documentos > POOB > LABORATORIOS > Laboratorio 3 > GARCIA ROMERO LAB 3 > schelling > bin > presentation

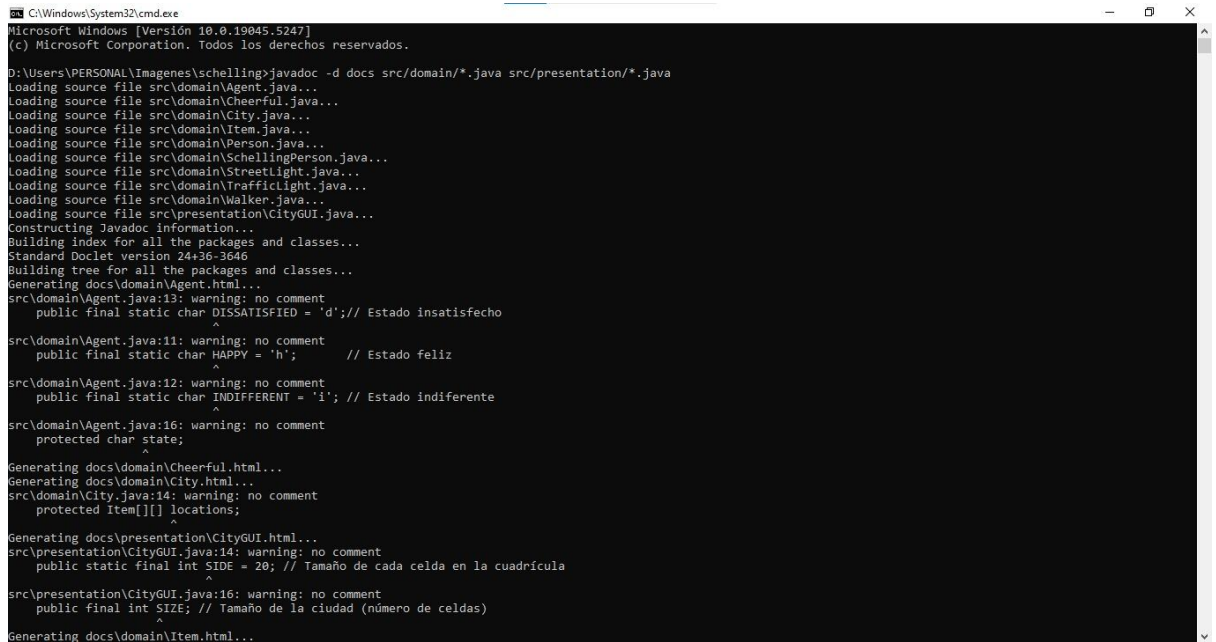
Nombre	Fecha de modificación	Tipo	Tamaño
CityGUI\$1.class	22/03/2025 7:20 p. m.	Archivo CLASS	1 KB
CityGUI.class	22/03/2025 7:20 p. m.	Archivo CLASS	2 KB
PhotoAManufacturing.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB
PhotoAutomata.class	22/03/2025 3:22 p. m.	Archivo CLASS	2 KB
PhotoCity.class	22/03/2025 7:20 p. m.	Archivo CLASS	3 KB

El comando lo que hizo fue añadir los archivos .class al directorio presentation, también en domain, del directorio test no se le pidió que se agregara, porque después nos piden que hagamos eso.

## DOCUMENTANDO

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

javadoc -d docs src/domain/\*.java src/presentation/\*.java

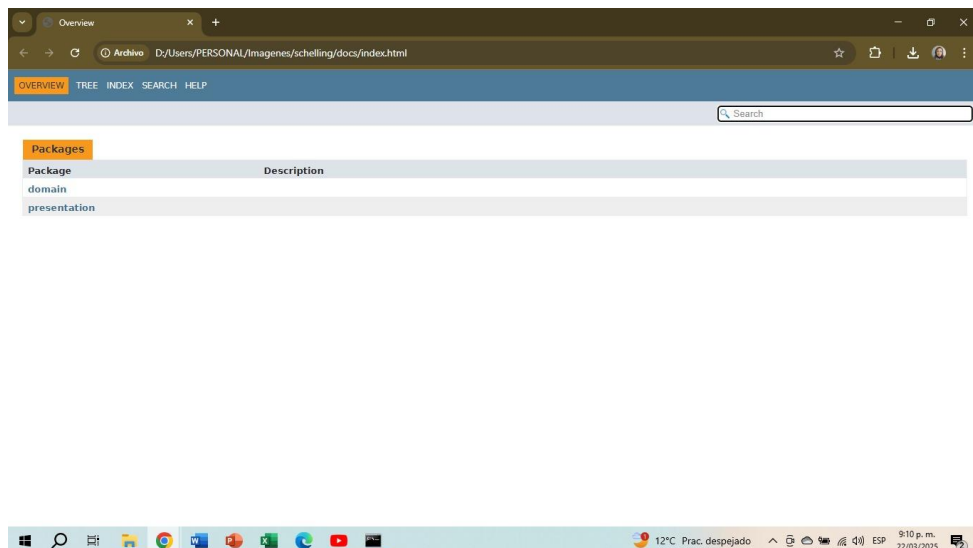


```
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Users\PERSONAL\Imágenes\schelling>javadoc -d docs src/domain/*.java src/presentation/*.java
Loading source file src\domain\Agent.java...
Loading source file src\domain\Cheerful.java...
Loading source file src\domain\City.java...
Loading source file src\domain\Item.java...
Loading source file src\domain\Person.java...
Loading source file src\domain\SchellingPerson.java...
Loading source file src\domain\StreetLight.java...
Loading source file src\domain\TrafficLight.java...
Loading source file src\domain\Walker.java...
Loading source file src\presentation\CityGUI.java...
Constructing Javadoc information...
Building index for all the packages and classes...
Standard Doclet version 24.36-3646
Building tree for all the packages and classes...
Generating docs\domain\Agent.html...
src\domain\Agent.java:13: warning: no comment
    public final static char DISSATISFIED = 'd'; // Estado insatisfecho
                        ^
src\domain\Agent.java:11: warning: no comment
    public final static char HAPPY = 'h'; // Estado feliz
                        ^
src\domain\Agent.java:12: warning: no comment
    public final static char INDIFFERENT = 'i'; // Estado indiferente
                        ^
src\domain\Agent.java:16: warning: no comment
    protected char state;
                        ^
Generating docs\domain\Cheerful.html...
Generating docs\domain\City.html...
src\domain\City.java:14: warning: no comment
    protected Item[][] locations;
                        ^
Generating docs\presentation\CityGUI.html...
src\presentation\CityGUI.java:14: warning: no comment
    public static final int SIDE = 20; // Tamaño de cada celda en la cuadrícula
                        ^
src\presentation\CityGUI.java:16: warning: no comment
    public final int SIZE; // Tamaño de la ciudad (número de celdas)
                        ^
Generating docs\domain\Item.html...
```

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

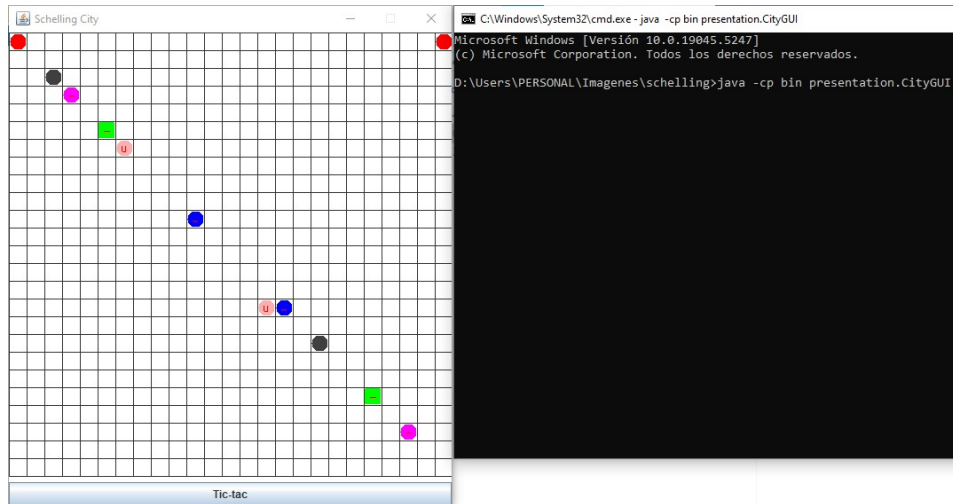
El archivo que hay que abrir para empezar a navegar por la documentación es index.html. Este archivo está en el directorio raíz de la documentación generada por Javadoc.



## EJECUTANDO

1. Empleando el comando java, desde el directorio raíz, ejecute el programa. ¿Cómo utilizó este comando?

java -cp bin presentation.GardenGUI



## PROBANDO

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

### Adicion Directorio Pruebas



equipo > Disco local (D:) > Usuarios > PERSONAL > Imágenes > schelling > src > test			
Nombre	Fecha de modificación	Tipo	Tamaño
CheerfulTest.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
PersonTest.java	22/03/2025 3:22 p. m.	Archivo JAVA	5 KB
SchellingPersonTest.java	22/03/2025 3:22 p. m.	Archivo JAVA	2 KB
StreetLightTest.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
TrafficLightTest.java	22/03/2025 3:22 p. m.	Archivo JAVA	3 KB
WalkerTest.java	22/03/2025 3:22 p. m.	Archivo JAVA	4 KB



Para poder incluir un paquete de tipo .jar en este caso las librerías que se usaron, se busco en internet el archivo, se descargo y se le menciona al código la librería que se usa.

**Código para implementar la librería:** "lib/\*;out"

**Instrucción completa:** javac -d out -cp "lib/\*;out" src/test/\*.java

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Users\PERSONAL\Imagenes\schelling>javac -d out -cp "lib/*;out" src/test/*.java

D:\Users\PERSONAL\Imagenes\schelling>
```

**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**

**El comando utilizado fue:** java -cp "lib/\*;bin;test" org.junit.platform.console.ConsoleLauncher --scan-classpath

**3. Pegue en su documento el resultado de las pruebas**

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Users\PERSONAL\Imagenes\schelling>java -cp "lib/*;bin;test" org.junit.platform.console.ConsoleLauncher --scan-classpath

Thanks for using JUnit! Support its development at https://junit.org/sponsoring

+ [36m.+ [0m
+ [36m+-- [0m + [36mJUnit Jupiter+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [36mStreetLightTest+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mshouldInitializeWithCorrectPositionAndInitialState()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mshouldChangeColorEveryThreeTurns()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mshouldNotBeAnAgent()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [36mWalkerTest+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestMovementAtCityBoundary()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestMovementWhenSpaceAvailable()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestConsecutiveMovements()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestInitialState()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [36mSchellingPersonTest+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mshouldBeDissatisfiedWhenFewSimilarNeighbors()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mshouldBeHappyWhenEnoughSimilarNeighbors()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [36mPersonTest+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestDecide_StateAfterZeroSteps()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestPersonAtBoundaryPosition()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestPersonIgnoresNeighbors()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestDecide_StateAfterTwoSteps()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestInitialState_Dissatisfied()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestInitialColor_Blue()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestInitialShape_Round()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestPositionRegisteredInCity()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestDecide_StateAfterOneStep()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestStepsIncrement_AfterChange()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [36mTrafficLightTest+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestColorCycle()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mtestPositionInCity()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [36mCheerfulTest+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mshouldAlwaysBeHappy()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mshouldMakeNeighborsHappy()+ [0m + [32m[OK]+ [0m
+ [36m| +-- [0m + [34mshouldNotAffectOutOfRangePersons()+ [0m + [32m[OK]+ [0m
+ [36m+-- [0m + [36mJUnit Vintage+ [0m + [32m[OK]+ [0m
+ [36m+-- [0m + [36mJUnit Platform Suite+ [0m + [32m[OK]+ [0m

Test run finished after 163 ms
[ 9 containers found ]
```

```
Test run finished after 163 ms
[      9 containers found      ]
[      0 containers skipped    ]
[      9 containers started    ]
[      0 containers aborted    ]
[      9 containers successful ]
[      0 containers failed     ]
[     24 tests found          ]
[      0 tests skipped         ]
[     24 tests started         ]
[      0 tests aborted         ]
[     24 tests successful      ]
[      0 tests failed          ]
```

## Empaquetando

1. Consulte cómo 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 ?

Dirigiéndose al directorio raíz, se compilan los paquetes domain y presentation, test no porque se nos pide que esas no.

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

Se crea el archivo manifest que especifica la clase principal que se debe ejecutar cuando se inicia el archivo JAR

```
echo Main-Class: presentation.CityGUI > manifest.txt
```

Se abrió el archivo de texto y se le agregaron las librerías correspondientes:

```
Class-Path: libs/
```

Y luego con el siguiente comando se crea el archivo .jar sin las fuentes ni las clases de prueba:

```
jar cvfm schelling.jar manifest.txt -C bin .
```



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5247]
(c) Microsoft Corporation. Todos los derechos reservados.

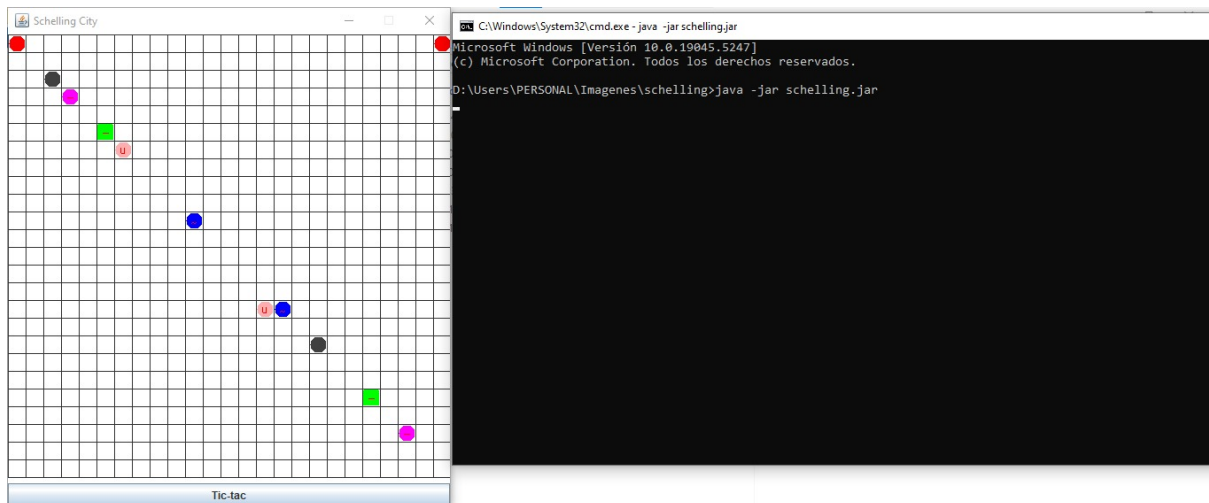
D:\Users\PERSONAL\Imagenes\schelling>javac -d bin src/domain/*.java src/presentation/*.java

D:\Users\PERSONAL\Imagenes\schelling>echo Main-Class: presentation.CityGUI > manifest.txt

D:\Users\PERSONAL\Imagenes\schelling>jar cvfm schelling.jar manifest.txt -C bin .
added manifest
adding: domain/(in = 0) (out= 0)(stored 0%)
adding: domain/Agent.class(in = 883) (out= 509)(deflated 42%)
adding: domain/Cheerful.class(in = 1042) (out= 690)(deflated 33%)
adding: domain/City.class(in = 1851) (out= 1150)(deflated 37%)
adding: domain/Item.class(in = 595) (out= 349)(deflated 41%)
adding: domain/Person.class(in = 1033) (out= 633)(deflated 38%)
adding: domain/SchellingPerson.class(in = 1416) (out= 944)(deflated 33%)
adding: domain/StreetLight.class(in = 874) (out= 535)(deflated 38%)
adding: domain/TrafficLight.class(in = 1056) (out= 620)(deflated 41%)
adding: domain/Walker.class(in = 1314) (out= 833)(deflated 36%)
adding: presentation/(in = 0) (out= 0)(stored 0%)
adding: presentation/CityGUI$1.class(in = 614) (out= 407)(deflated 33%)
adding: presentation/CityGUI.class(in = 1888) (out= 1100)(deflated 41%)
adding: presentation/PhotoAManufacturing.class(in = 1984) (out= 1057)(deflated 46%)
adding: presentation/PhotoAutomata.class(in = 1917) (out= 1038)(deflated 45%)
adding: presentation/PhotoCity.class(in = 2056) (out= 1172)(deflated 42%)
adding: test/(in = 0) (out= 0)(stored 0%)
adding: test/CheerfulTest.class(in = 1473) (out= 819)(deflated 44%)
adding: test/PersonTest.class(in = 3333) (out= 1585)(deflated 52%)
adding: test/SchellingPersonTest.class(in = 1210) (out= 692)(deflated 42%)
adding: test/StreetLightTest.class(in = 1475) (out= 831)(deflated 43%)
adding: test/TrafficLightTest.class(in = 1577) (out= 886)(deflated 43%)
adding: test/WalkerTest.class(in = 2372) (out= 1282)(deflated 45%)

```

## 2. ¿Cómo se ejecuta el proyecto empaquetado?



## RETROSPECTIVA

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

Anderson Fabian Garcia Nieto: 19

Christian Alfonso Romero Martinez: 19

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

El laboratorio se culminó, principalmente gracias a la persistencia y al apoyo mutuo en dudas y dificultades.

Todos los métodos se completaron

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

Lo consideramos muy importante, pues logramos solventar varios errores gracias al programar en equipo, por otro lado también nos guió el desarrollo el implementar primero las pruebas, puesto que nos dio un objetivo claro en el desarrollo.

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

Si fue un logro, porque la configuración de las máquinas no impuso un gran reto, en un caso particular, uno de nosotros tiene configuraciones específicas en el computador para otros desarrollos, que con el desarrollo por consola generó algunos conflictos, sin embargo al ser trabajo en equipo lo logramos solventar

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

Para mejorar los resultados consideramos importante entender los problemas e investigar dinámicas de solución para tomar soluciones más óptimas, al igual que consultar las dificultades, pues se pudieron solucionar más rápido permitiendo un mejor resultado

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

Microsoft. (2023). Windows Commands. Microsoft Docs.

Oracle. (2023). Java Development Kit (JDK) Documentation.

GeeksforGeeks. (s. f.). Abstract Classes in Java

Baeldung. (s. f.). Abstract Class vs Interface in Java

Barker, J. (2005). Beginning Java Objects: From Concepts to Code (2nd ed.). Apress.

Oracle. (n.d.). Java Platform, Standard Edition Documentation. Oracle

Copilot. (2025). Respuesta a consulta sobre implementación de arrays en Java. Microsoft.

Charte, F. (2021, February 9). Paquetes en Java: qué son, para qué se utilizan, y cómo se usan (con vídeo). campusMVP.es.

<https://www.campusmvp.es/recursos/post/paquetes-en-java-que-son-para-que-se-utilizan-y-como-se-usan.aspx>