Escuela Colombiana de Ingeniería Julio Garavito

Andrés Cardozo

Tulio Riaño

Laboratorio 03

Programacion Orientada a Objetos

FECHA:

14/03/2025

**Conociendo**

1. En el directorio descarguen los archivos contenidos en schelling.zip. Revisen el código: a) ¿Cuántos paquetes tiene? d) ¿Cuál es el propósito del paquete presentación? e) ¿Cuál es el propósito del paquete dominio?

Cuando abrimos el documento nos encontramos con dos paquetes: presentacion y domain.

El paquete presentación se encarga de mostrarnos la ciudad.

El paquete dominio nos está implementando el contexto del modelo de Schelling, encontramos la ciudad, donde se colocan items, de los cuales pueden ser personas.

1. Revisen el paquete de dominio, a)¿Cuáles son los diferentes tipos de componentes de este paquete? b) ¿Qué implica cada uno de estos tipos de componentes?

Encontramos 2 clases concretas, una interfaz y una clase abstracta, así como por las relaciones notamos que la clase person implementa la interfaz y extiende la clase abstracta Agent.

Además, podemos ver que persona es de una ciudad, como la ciudad tiene como locations de tipo de la interfaz.

1. Revisen el paquete de presentación, a) ¿Cuántos componentes tiene? b) ¿Cuántos métodos públicos propios (no heredados) ofrece?

Tiene una clase concreta llamada cityGUI , y esta clase tiene la clase CityGUI que extiende JFrame y PhotoCity que extiende JPanel, esto se ve asi ya que PhotoCity es como una clase auxiliar para CityGUI. Los metodos que no heredan son: PhotoCity(CityGUI gui), main(String[] args), gettheCity(), CityGUI()-Constructor, prepareElements(), prepareActions(), ticTacButtonAction().

///todos menos paintComponent(Graphics g)

(con ayuda de referencia 2)

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

Se debe ejecutar el método main que se encuentra en la clase CityGUI, ya que es el paquete de presentación.

1. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué? (Deben ejecutar la aplicación java, no crear un objeto como lo veníamos haciendo)

Principalmente al ejecutar el programa se nos abre una pestaña en la que podemos observar una cuadricula en la que se presenta la funcionalidad de tic-tac, esta última no deja hacer nada adicional porque no se ha implementado al 100% el correspondiente método.

**Arquitectura general.**

1. Consulte el significado de las palabras package e import de java.

¿Qué es un paquete? ¿Para qué sirve?

La palabra clave package en Java se utiliza para agrupar clases, interfaces y subpaquetes relacionados. Ayuda a organizar el código y a evitar conflictos de nombres. Los paquetes proporcionan protección de acceso y gestión del espacio de nombres. (tomado de 3)

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

La palabra clave import en Java se utiliza para hacer visibles otras clases o paquetes enteros dentro de tu clase actual. Esto te permite utilizar clases de otros paquetes sin tener que hacer referencia a sus nombres completos.

La declaración import se coloca al principio de un archivo fuente Java, después de la declaración del paquete (si existe) y antes de la declaración de la clase. Ayuda a que el código sea más legible y fácil de mantener al permitir el uso de nombres de clase simples en lugar de nombres totalmente cualificados.

En este laboratorio nosotros importamos tanto JFrame como JPanel ya que son clases que se utilizan para crear interfaces gráficas de usuario (GUI):

* JFrame: Crea una ventana principal de la aplicación.
* JPanel: Es un contenedor dentro de la ventana principal para organizar y agrupar otros componentes visuales.

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

contenido. ¿Qué coincidencia hay entre paquetes y directorios?

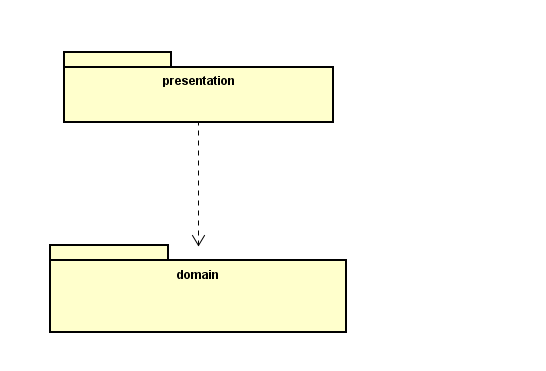
La coincidencia que se presenta entre paquetes y directorios es que este último guarda de la misma manera los paquetes.

El contenido que observamos en el directorio son los respectivos paquetes domain y presentation. En doc se presenta, contenido variado como son varios archivos con extensiones html, algunos java script y carpetas como aplication, domain, presentacion, presentation y resources, cada una de las anteriores tienen archivos que se caracterizan por su extensión html.

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.

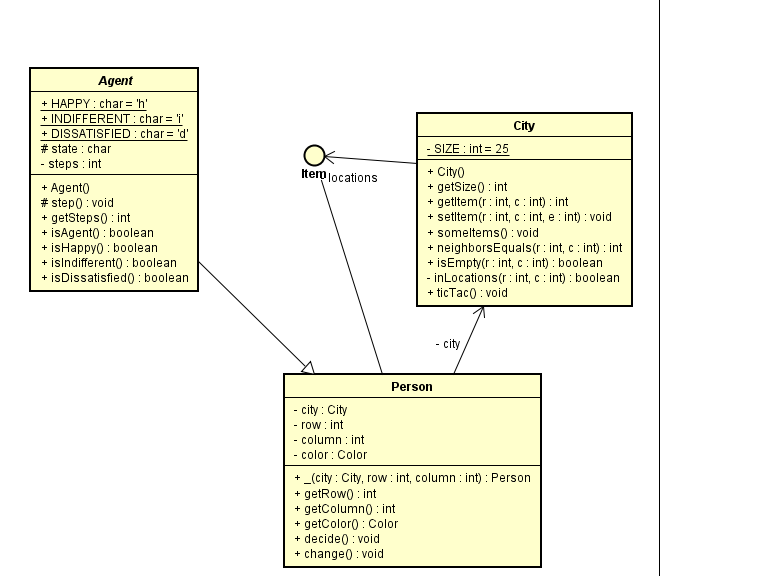
En astah, crear un diagrama de clases (cambiar el nombre por Package Diagram0)



**Arquitectura detallada.**

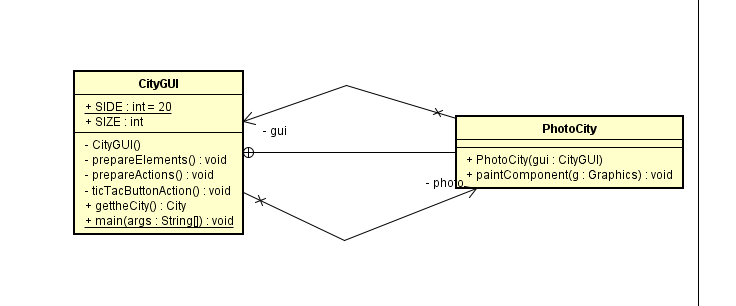
1. Para preparar el poyecto para BDD. Completen el diseño detallado del paquete de dominio. Adicionen el diagrama de clases en el paquete correspondiente. a) ¿Qué componentes hacían falta?

Los componentes que faltaban eran pertenecientes al paquete de dominio, son person e item.



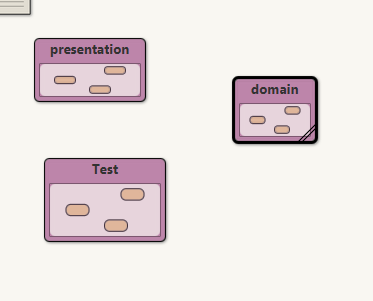
1. Completen el diseño detallado del paquete de presentación. Adicionen el diagrama de clases al paquete correspondiente. a) ¿Por qué hay dos clases y un archivo .java?

Hay dos clases porque una es clase interna (PhotoCity) de la otra (CityGUI). Y un solo archivo Java que es CityGUI.



1. 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?

El paquete que debe usar test es dominio porque es desde donde se va a realizar las pruebas en el back, por otra parte, este va a estar asociado a la clase city, ya que la instanciación de esta clase es con el que vamos a interactuar durante las pruebas.



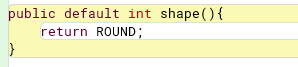
**CICLO 1. Iniciando con las personas normales**

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

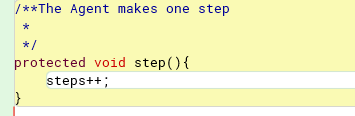
La colección que usa para albergar cosas es un vector de 2D de items, puede recibir personas ya que está implementando la interfaz ítem que se llama locations, por último item esta implementado en personas.

1. 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.

Person se crea con un modificador de acceso público, las clases que definen Person son Agent (Clase Abstracta) y Item (interface), se crea en un estado inicial en una fila, columna, ciudad y un color (Blue).

Un item está definido como square y round, y por defecto es un circulo.

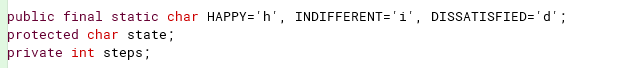
Podria referirse a aumentar su tiempo al metodo: 

1. 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.

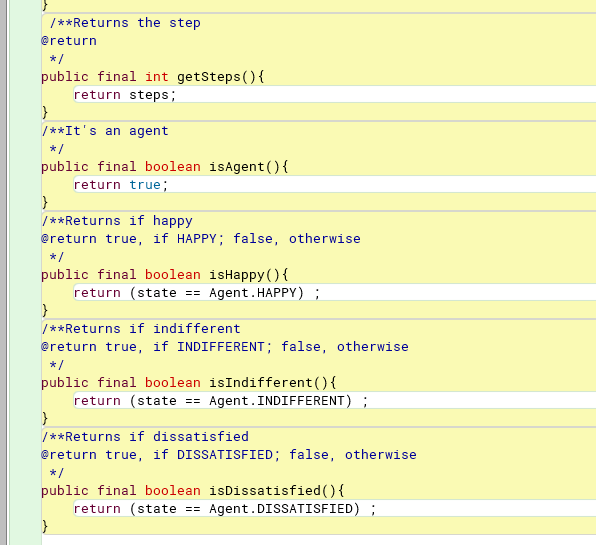
Los atributos que hereda persona por ser un Agent es un char que puede ser:

(HAPPY='h', INDIFFERENT='i', DISSATISFIED='d')

Y otro Char definido como modificador de acceso protegido llamado state, tambien un entero steps.

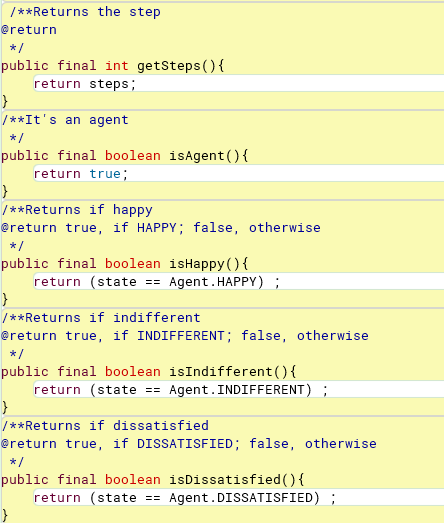


Puede hacer un step, obtener los steps que ha hecho, saber si es un agente, y saber en qué estado esta.



Person por ser un Agent no decide hacer nada distinto a lo que hace un agent.

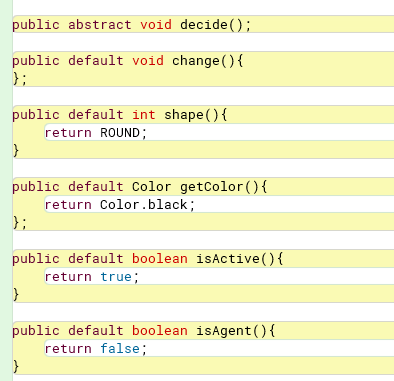
Person por ser un agent no puede hacer nada distinto las cosas que tienen final, como el atributo del estado (HAPPY='h', INDIFFERENT='i', DISSATISFIED='d'), los demas metodos finales:



Person por ser un Agent debe aprender a cambiar su estado y dar un paso para poder usarlo en la carpeta presentacion.

1. 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.

Persona por compartarse como un item sabe retornar el color, isActive, isAgent, shape, change y decide.



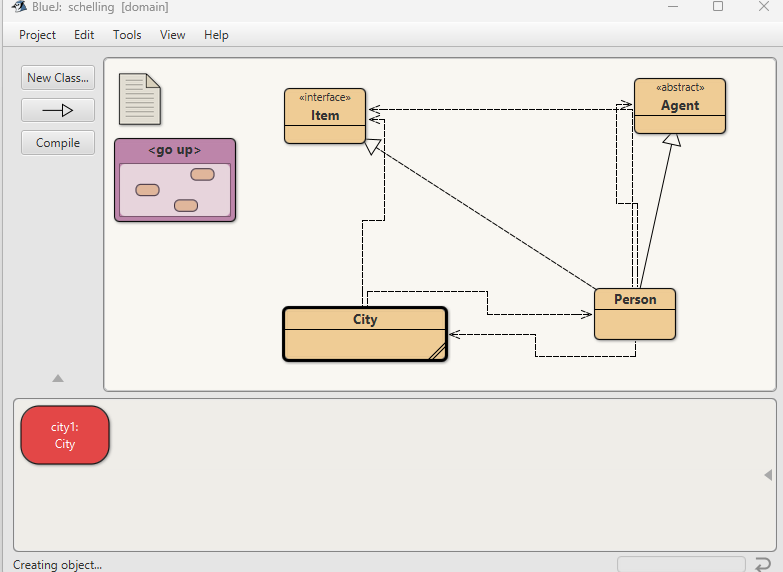
Decide hacer distinto decide y change ya que son metodos que implementamos/ sobreescribimos en persona, y del modo modo debe aprender a hacer estos.

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

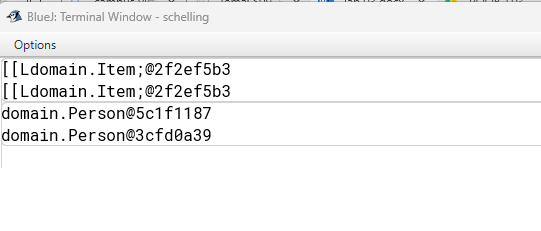
Una persona actua con change y decide, con cada change cambia aumentando un step, y decide como estar de acuerdo con estos steps. Por ejemplo, si steps es multiplo de 3 va a estar HAPPY.

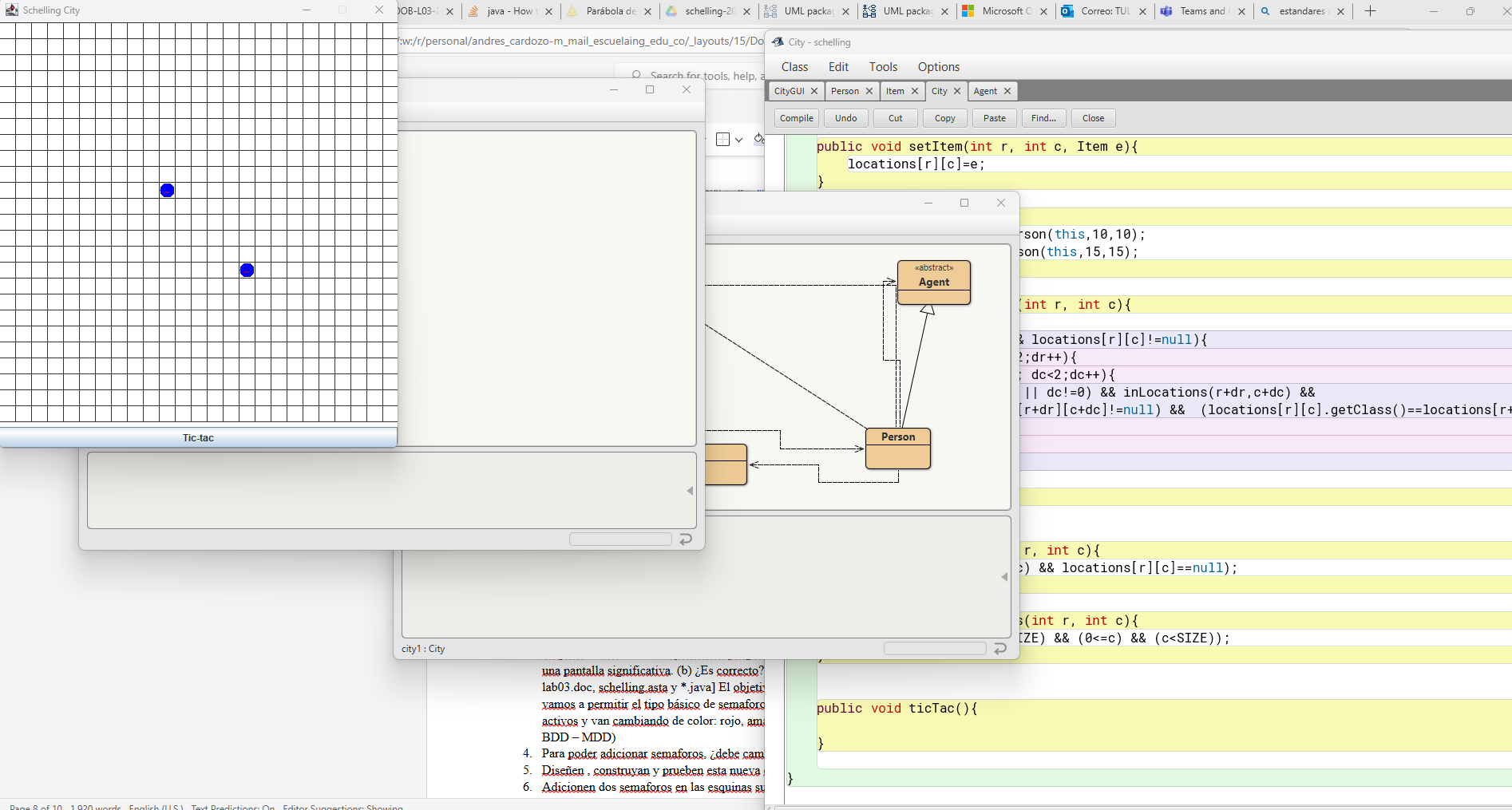
1. Ahora vamos a crear dos personas en diferentes posiciones (10,10) (15,15) llámelas adan y eva usando el método someItems(). Ejecuten el programa, ¿Qué pasa con las personas? ¿Por qué? Capturen una pantalla significativa.

Se crean las personas y se ubican en la posición correspondiente del tablero, pero visualmente no se observa.



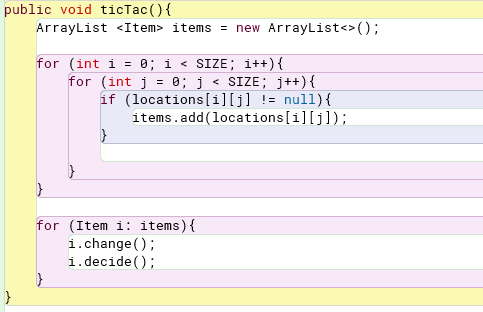
Si imprimimos en el método de setItems podemos ver que si se crean y se asignan en el tablero las personas.

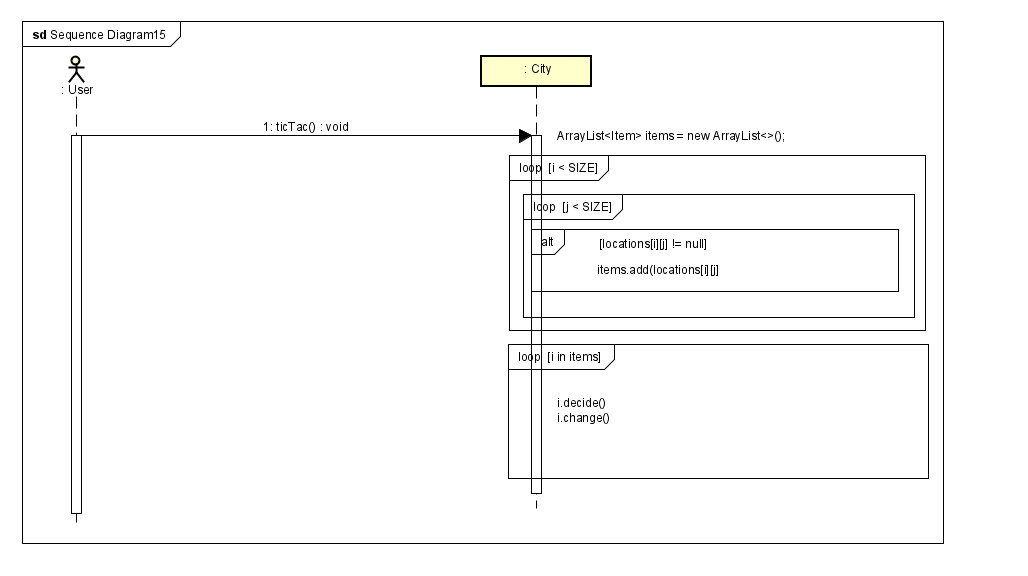


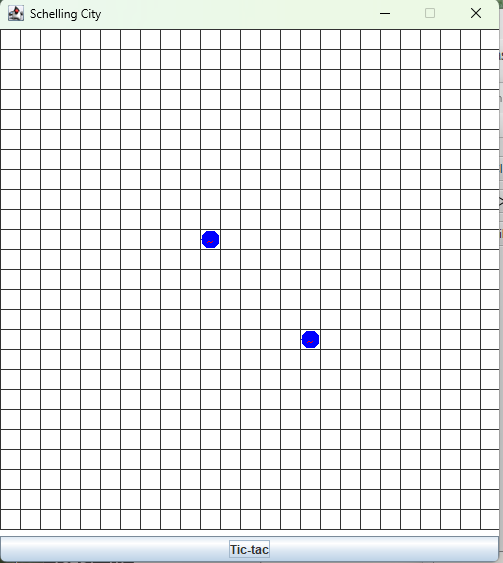
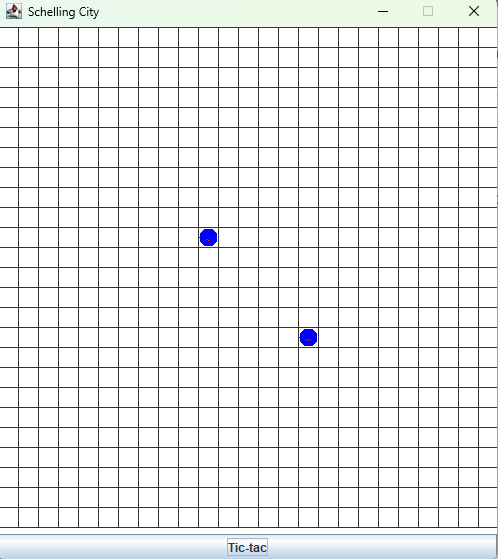
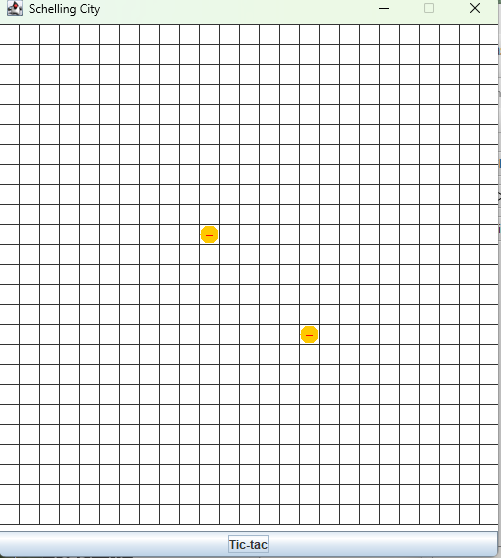
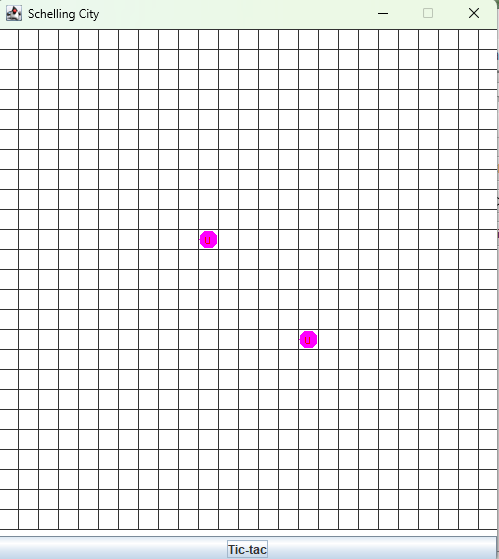


Si usamos el paquete de presentación y ejecutamos se observa en la cuadricula las correspondientes personas, esto debido a que, cuando creamos una persona automaticamente se añade al tablero (ciudad).

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

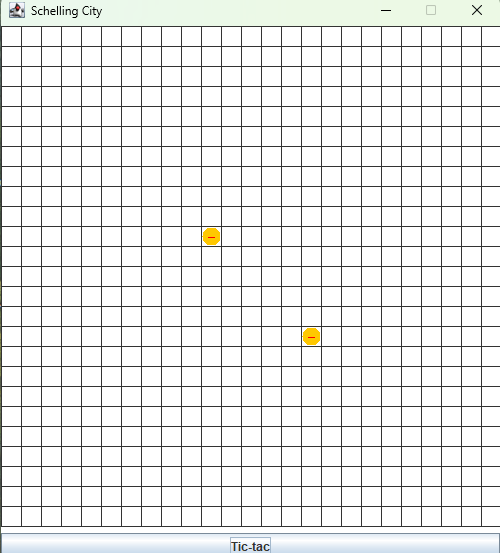




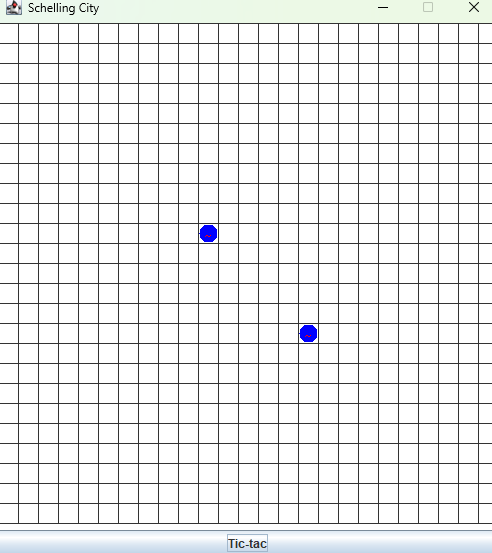
  

1. ¿Cómo quedarían adan y eva después de uno, dos, cuatro y seis Tic-tac? Ejecuten el programa. Capturen pantallas significativas en momentos correspondientes. ¿Es correcto?

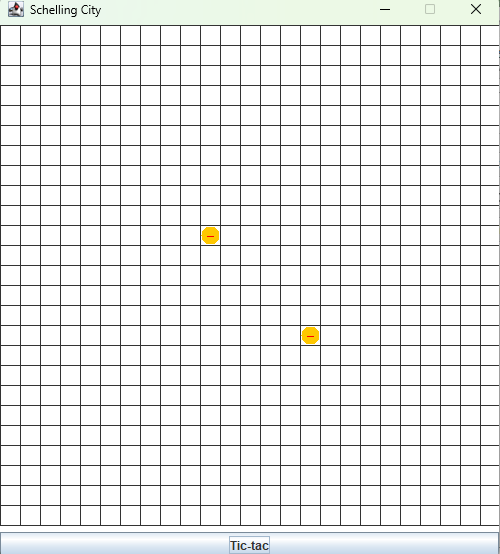
Después de un Tic-tac su estado podemos ver que va a ir cambiando



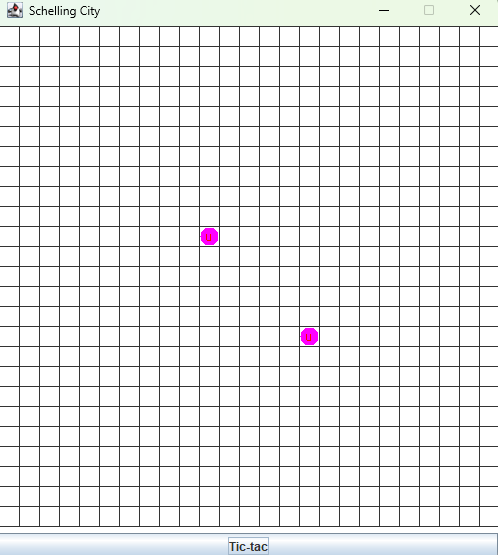
Después de otro Tic-Tac nuevamente cambia



Con otros 2 Tic-Tac para cuatro, vuelve al estado que tenía después del primer tic tac.



Y Por último al sexto Tic-Tac



Esto es correcto ya que sigue la condición de decide().

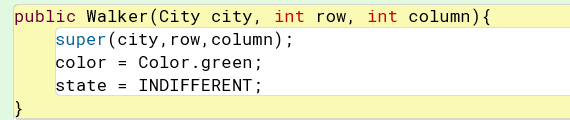
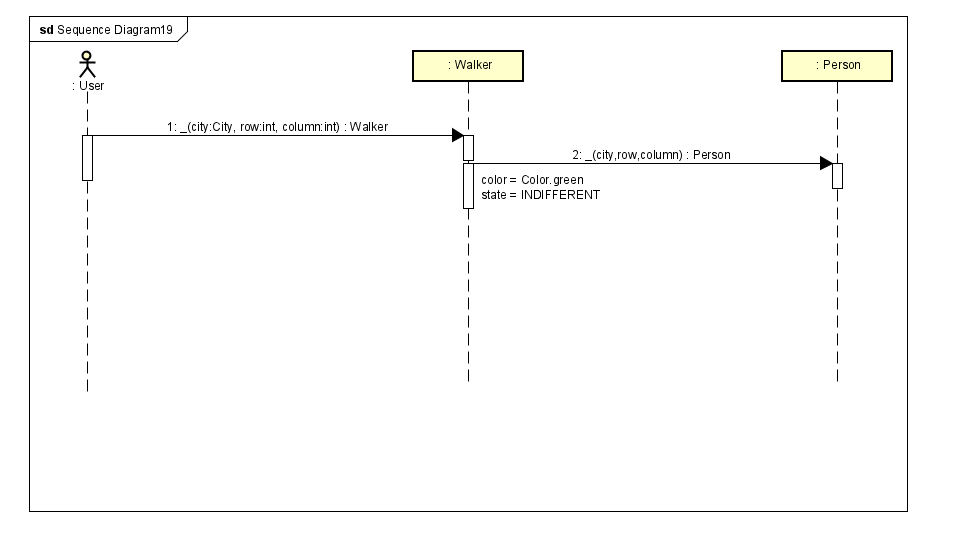
**Ciclo 2.** **Incluyendo a los caminantes**

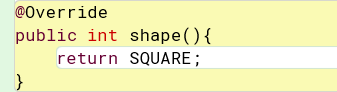
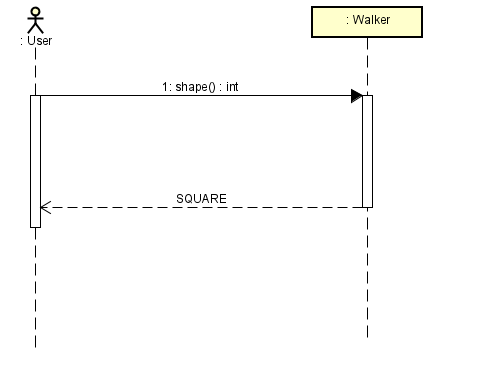
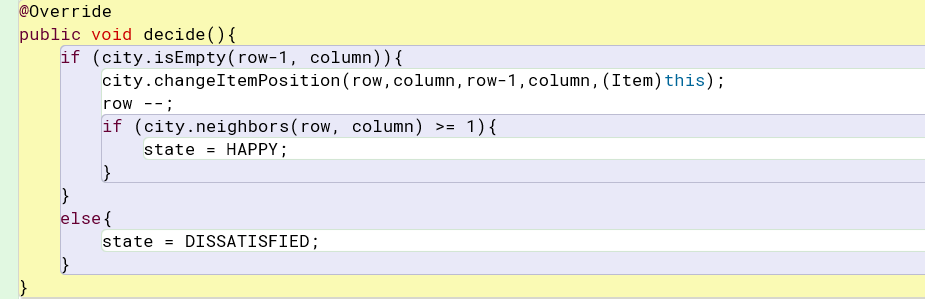
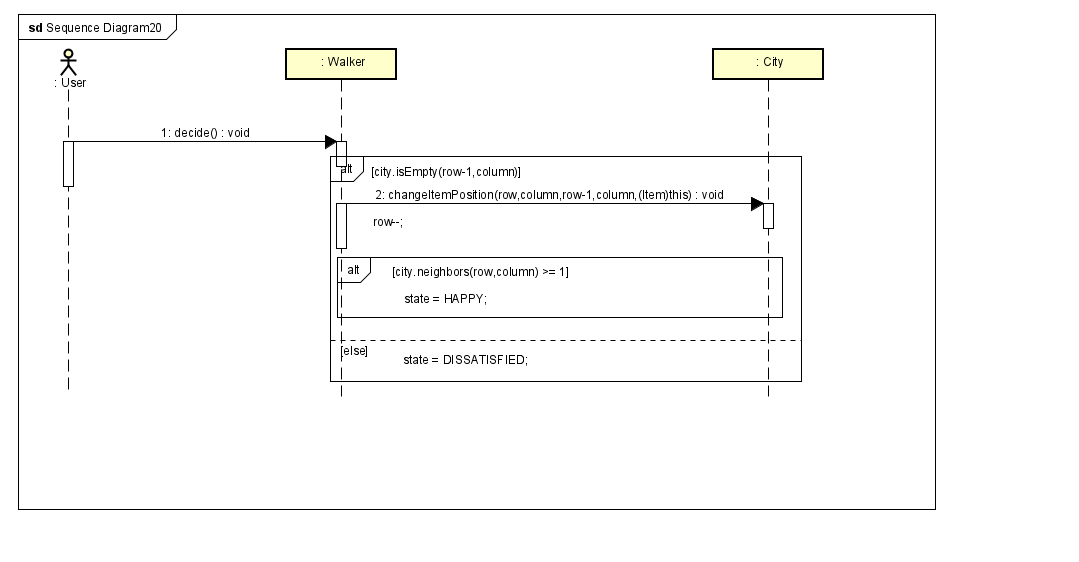
El objetivo de este punto es permitir recibir personas caminantes. Ellas (i) son rectangulos de color verde; (ii) inician indiferentes; (ii) se mueven hacia el norte1, (iii) si quedan vecinos a un item, se ponen felices; (iv) si no logran moverse al sitio que querían, quedan insatisfechos.

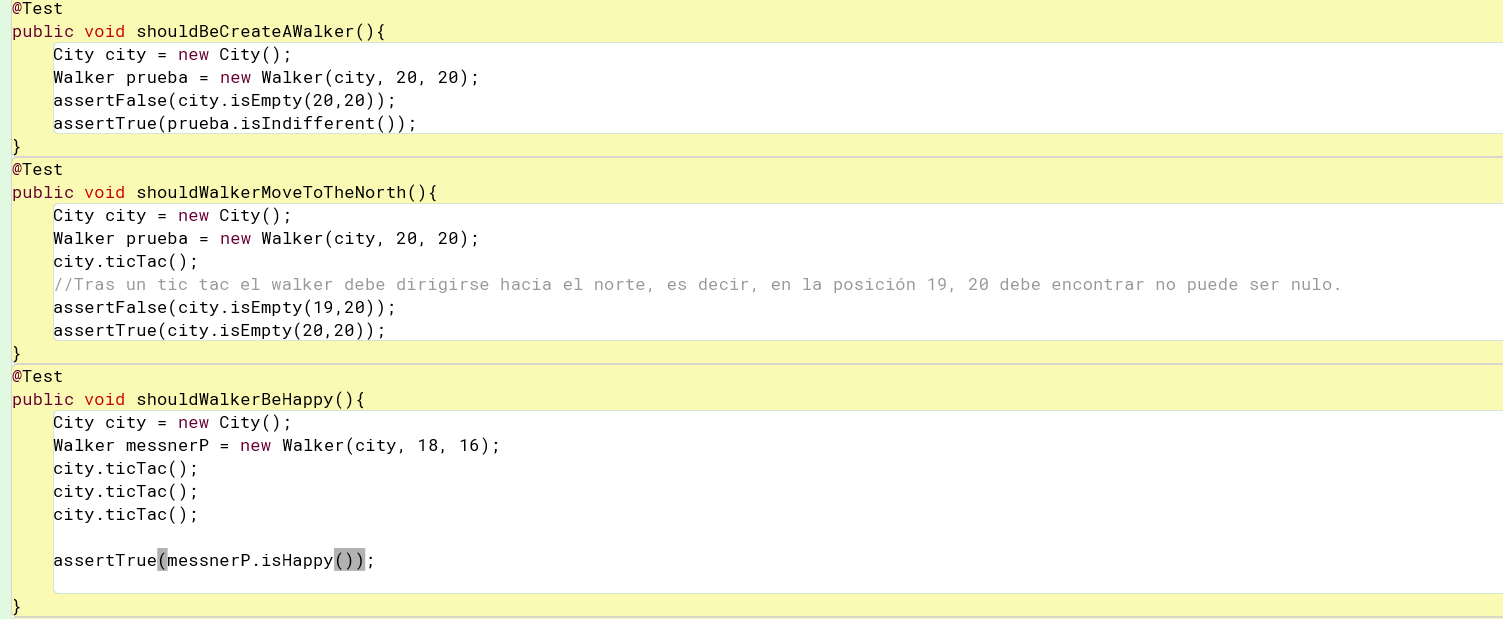
1. Para implementar esta nueva persona Walker ¿cuáles métodos se sobre-escriben (overriding)?

Sobreescribimos los metodos de decidir, de shape y el constructor, ya que tienen un comportamiento diferente a las personas.

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

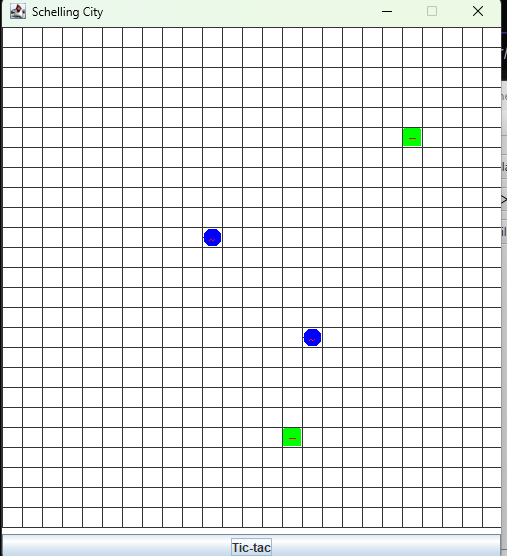




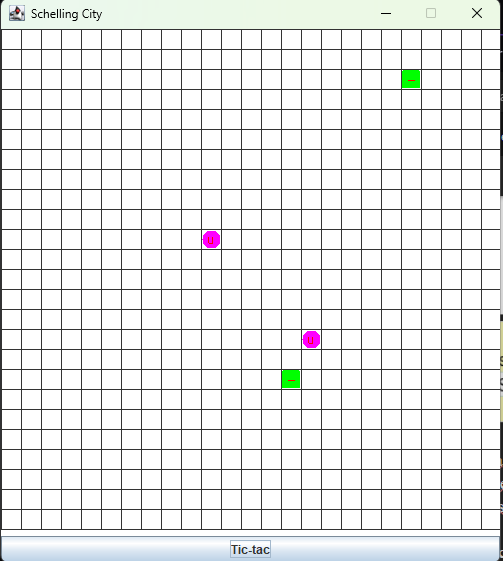


1. Adicione una pareja de caminantes, llámelas messner y kukuczka, (a) ¿Cómo quedarían después de tres Tic-tac? Ejecuten el programa y hagan tres clics en el botón. Capturen una pantalla significativa. (b) ¿Es correcto?

Despues de un tic-tac los caminantes deben subir 1 visualmente en las filas y no deberian de cambiar de estado.

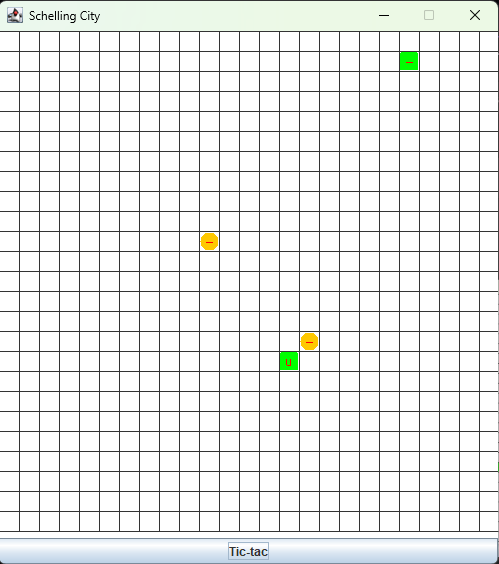


Despues de tres clics se ve asi:

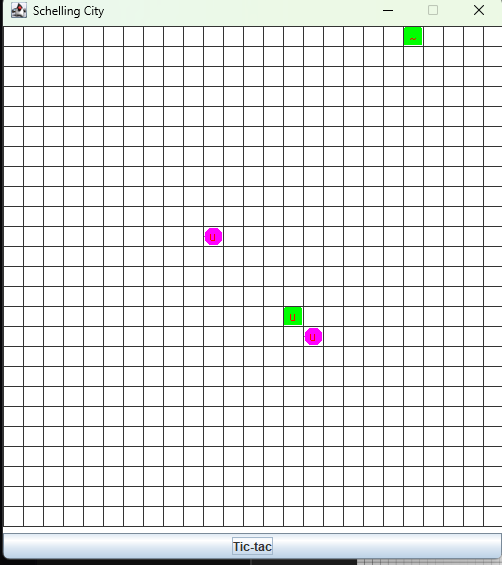


Y esto es correcto ya que cumple con las funciones de walker.

si quedan vecinos a un item, se ponen felices:



Si no logran moverse al sitio que querían, quedan Insatisfechos:



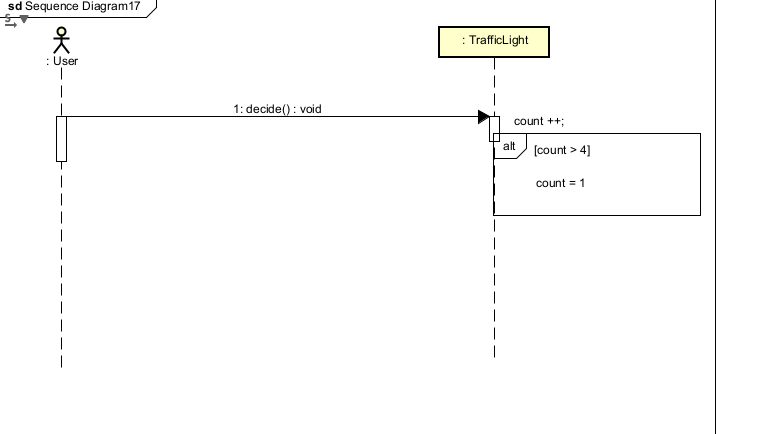
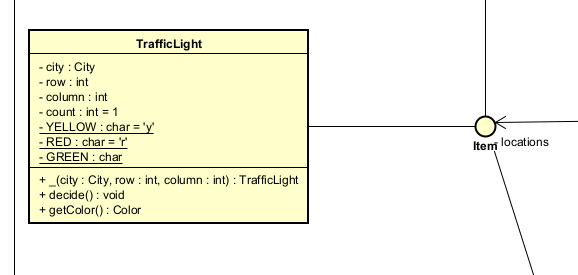
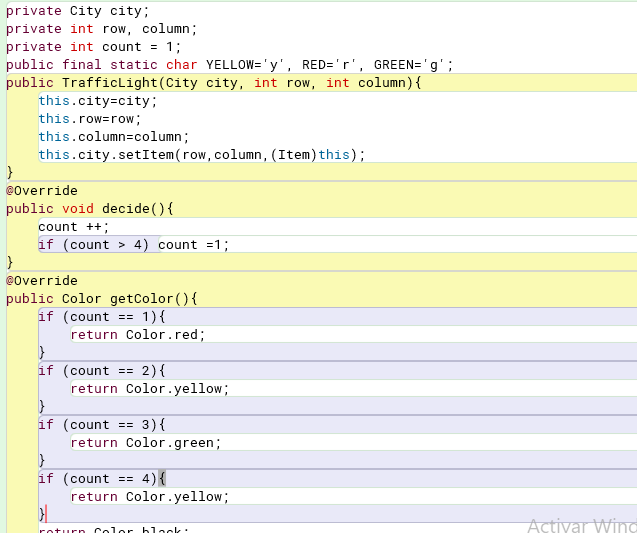
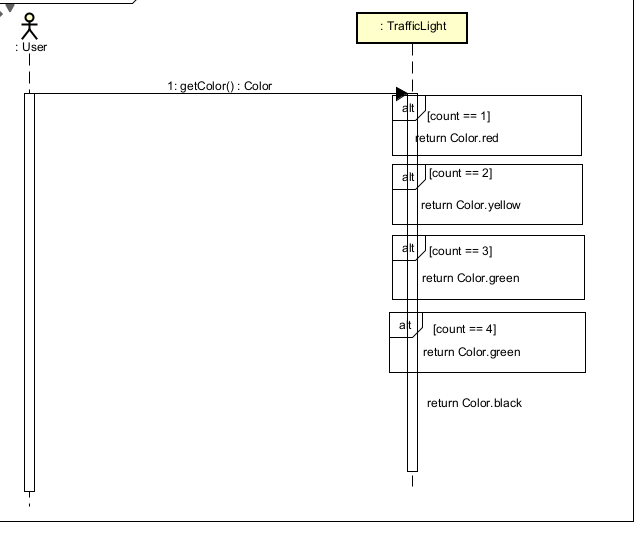
**Ciclo 3. Adicionando semaforos**

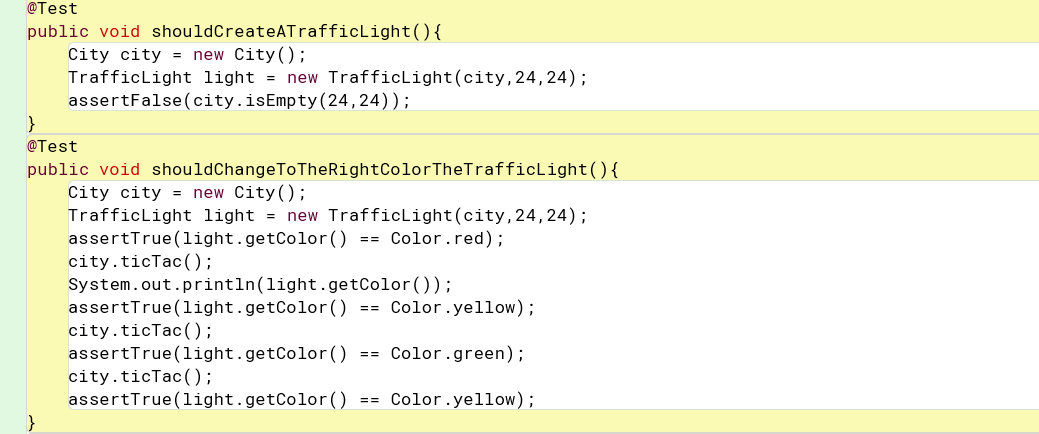
El objetivo de este punto es incluir semaforos (sólo vamos a permitir el tipo básico de semaforos) los semaforos son redondos, siempre activos y van cambiando de color: rojo, amarillo, verde, amarillo, rojo, etc (NO OLVIDE BDD – MDD)

1. Para poder adicionar semaforos, ¿debe cambiar en el código de City en algo? ¿por qué?

No ya que definimos que un semaforo implementa Items, por lo que el codigo de ciudad no tendra problemas al manejarlo.

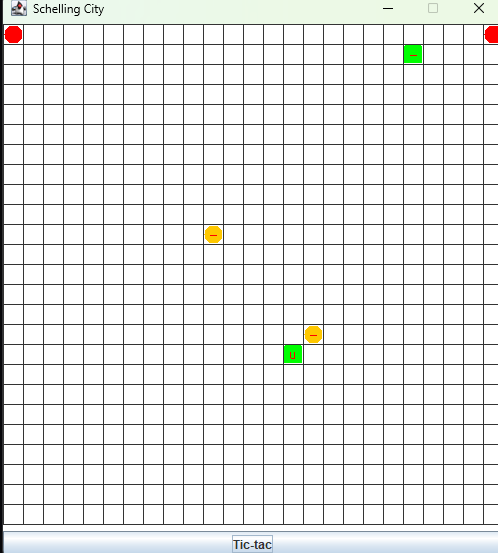
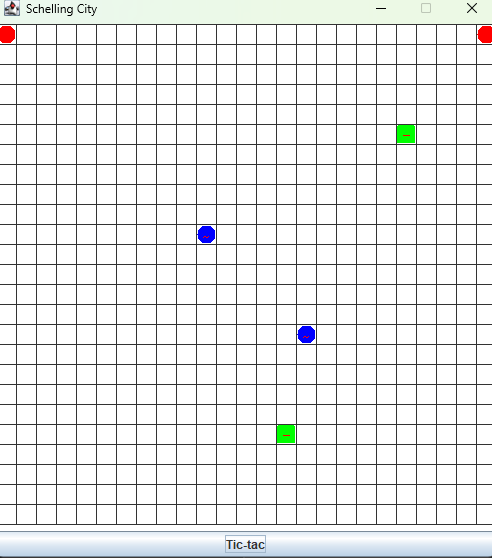
1. Diseñen , construyan y prueben esta nueva clase. (Mínimo dos pruebas de unidad)



1. 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?

Despues de 4 tic-tacs como comienza en rojo, deberian ser: primer tic amarillo, segundo verde, tercero amarillo, y deberia quedar en rojo de nuevo.



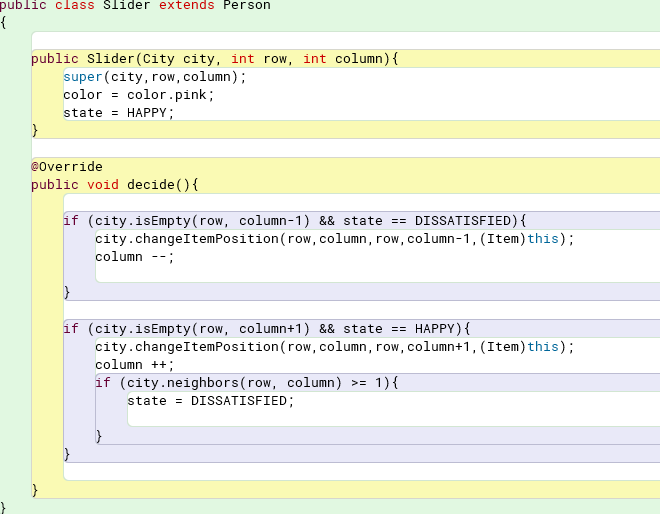
Es correcto ya que comienza en rojo.

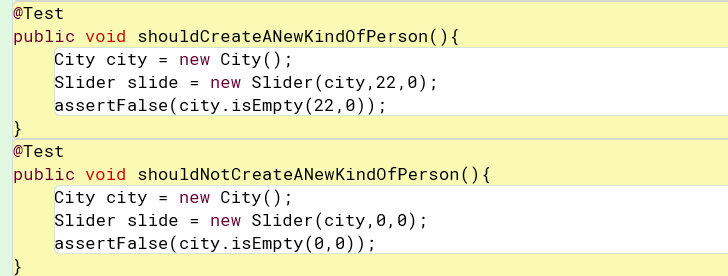
**Ciclo 4. Nueva persona: Proponiendo y diseñando**

El objetivo de este punto es permitir recibir en un nuevo tipo de Item.

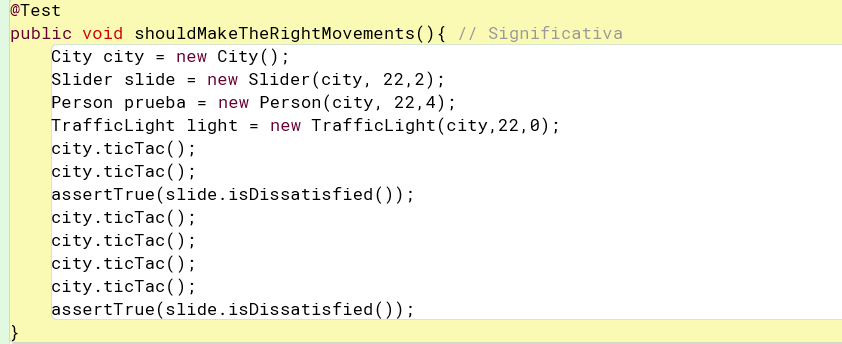
1. Propongan, describan e implementen el nuevo tipo de persona. (Mínimo dos pruebas de unidad)

Nosotros vamos a proponer la nueva persona llamada slider, la cual va a ser una persona que sale a trotar hacia el este. Son personas deportivas asi que en cada tic-tac se va a mover una casilla a la derecha, pero estas personas tambien son muy introvertidas: por lo que, si se encuentran alrededor de otra persona, van a devolverse (hacia la izquierda).





1. Considerando una pareja de ellas con el apellido de ustedes. (a) Piensen en otra prueba significativa y expliquen la intención. (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.

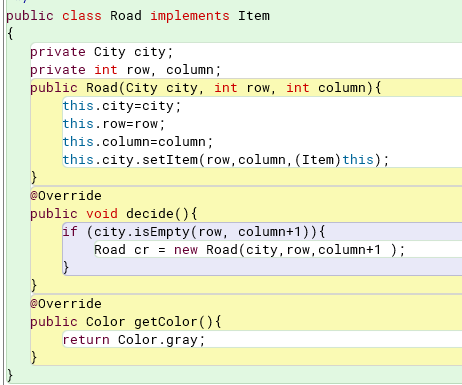


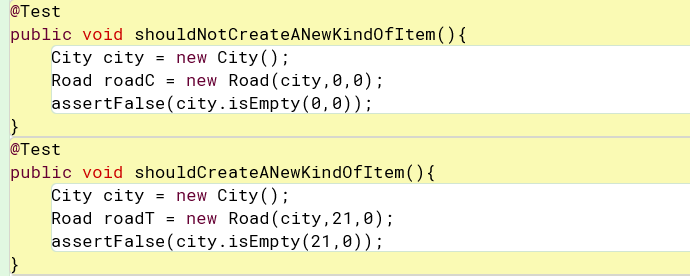
**Ciclo 5. Nuevo item: Proponiendo y diseñando**

El objetivo de este punto es permitir recibir un nuevo item (no persona) en la ciudad (NO OLVIDE BDD – MDD)

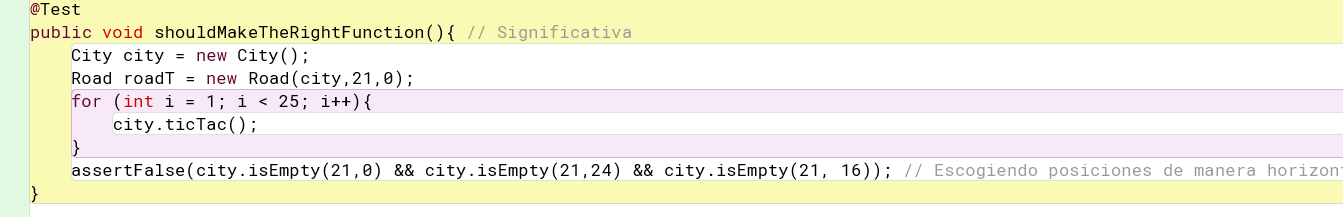
1. Propongan, describan e implementen el nuevo tipo de item. (Mínimo dos pruebas de unidad)

El nuevo item que proponemos va a ser una carretera en construccion, la cual va a atravesar la cuiduad de izquierda a derecha con cada tic-tac. Por lo que las personas no la pueden cruzar.





1. Considerando un par de ellos con el nombre de ustedes. (a) Piensen en otra prueba significativa y expliquen la intención. (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.



1 Primero, todos los caminantes deciden hacia donde se van a mover y luego todos los caminantes se mueven. Dos caminantes no se pueden mover hacia la misma posición.

**Ciclo 6. BONO**.

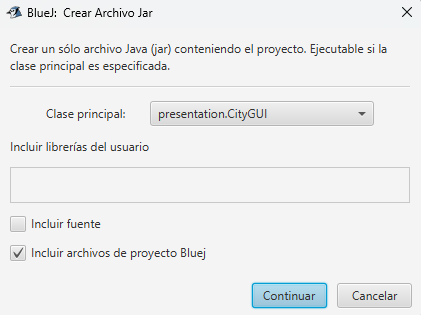
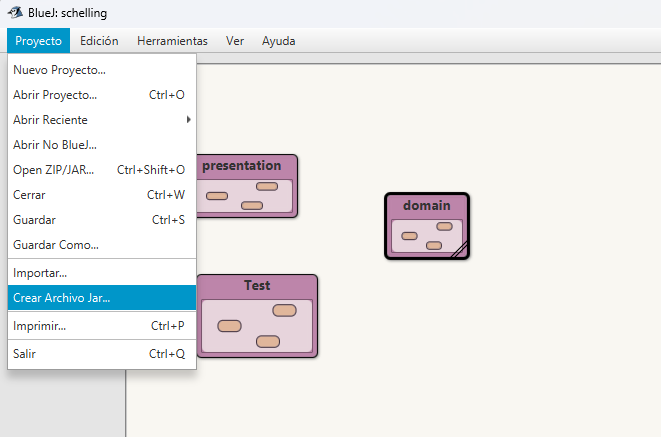
Persona Schelling [[http://ncase.me/polygons-es/]](http://ncase.me/polygons-es/%5D) La persona Schelling se rigue por las siguientes reglas:

* 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. Primero todas las personas Schelling toman la decisión de lo que pasará en el tiempo siguiente y luego la realizan.

**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.

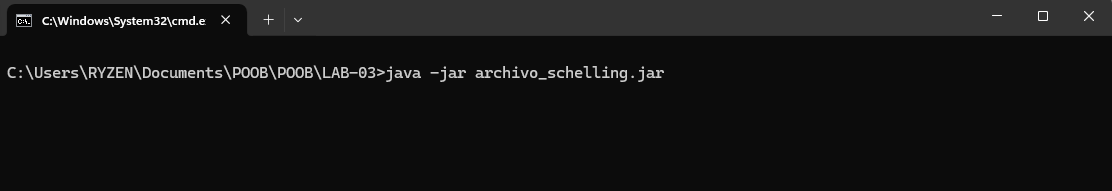
Bluej nos presenta una opción que es crear archivo .jar acá debemos seleccionar la clase principal que va a ser presentation, la podemos identificar ya que tiene un método main.

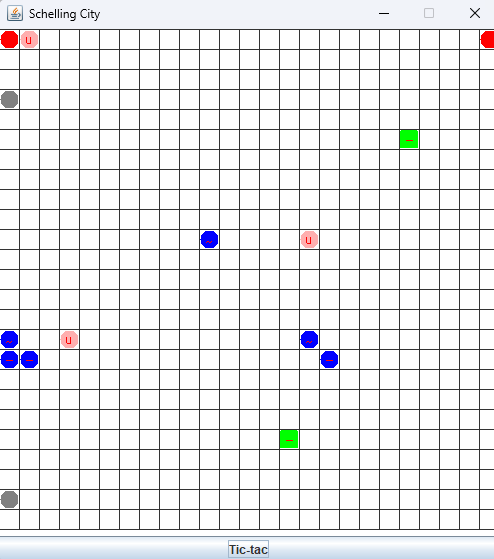


Desactivamos la opción de incluir fuente ya que solo queremos el ejecutable y no el código fuente.

1. Consulte el comando java para ejecutar un archivo jar. ejecutennlo ¿qué pasa?

El comando para ejecutar desde la consola es:

Cuando se ejecuta podemos ver la cuadricula correspondiente



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

Facilita que el cliente tenga que manejar múltiples archivos .class, facilita el manejo del programa ya que este va a contener todos los archivos .jar donde contiene las clases compiladas y otros recursos (imágnes, archivos de configuración). Permite ejecutar el programa fácilmente mediante la consola o dando doble clic en el archivo que se genera en la carpeta correspondiente.

**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.

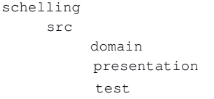
**Comandos básicos del sistema operativo**

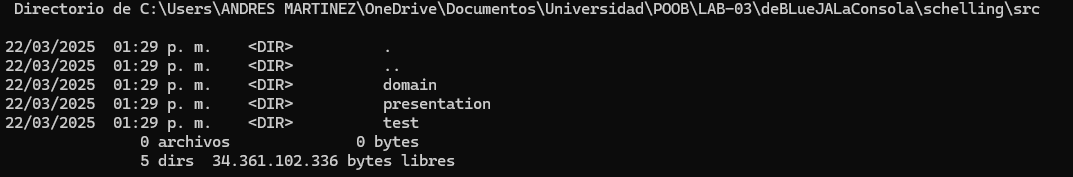
es de iniciar debemos repasar los comandos básicos del manejo de la consola.

1. Investiguen los comandos para moverse en la estructura de directorios:

|  |  |
| --- | --- |
| COMANDO | FUNCION |
| CD: | Uno de los comandos más esenciales de la consola de Windows. Sirve para cambiar de directorio, utilizando la fórmula cd < RutaDirectorio > para ir al directorio o carpeta concreta que le digas, o cd.. (con dos puntos) para salir de una carpeta e ir al nivel superior o carpeta donde estaba alojada. |
| Crear: | Hacer un directorio: mkdir |
| Borrar/Eliminar un archivo:  DEL “Archivo o Carpeta” | Borra el archivo o carpeta que le indiques. |
| listar su contenido: dir | El comando lista el contenido del directorio o carpeta donde te encuentras, mostrando todas las subcarpetas o archivos que tiene. Con este comando podrás saber si el archivo que buscas está ahí o a qué subcarpeta navegar. |
| Copiar:  COPY “Archivo Destino” | Copia uno o más archivos en la dirección que tu elijas. |
| MOVE Archivo Destino | Mueve el archivo concreto que quieras del lugar o carpeta en el que está a otra dirección que le digas. Es como copiar, pero sin dejar el archivo en su ubicación original. |

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





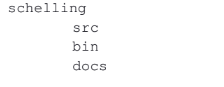
3. En el directorio copien únicamente los archivos \*.java del paquete de aplicación .

Consulte y capture el contenido de src/domain



**Estructura de proyectos java**

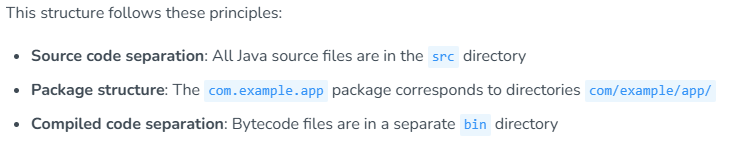
En java los proyectos se estructuran considerando tres directorios básicos.



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

organización interna de cada una de ellas.

Un proyecto en Java suele organizarse en estas carpetas, en este caso schelling se organiza:

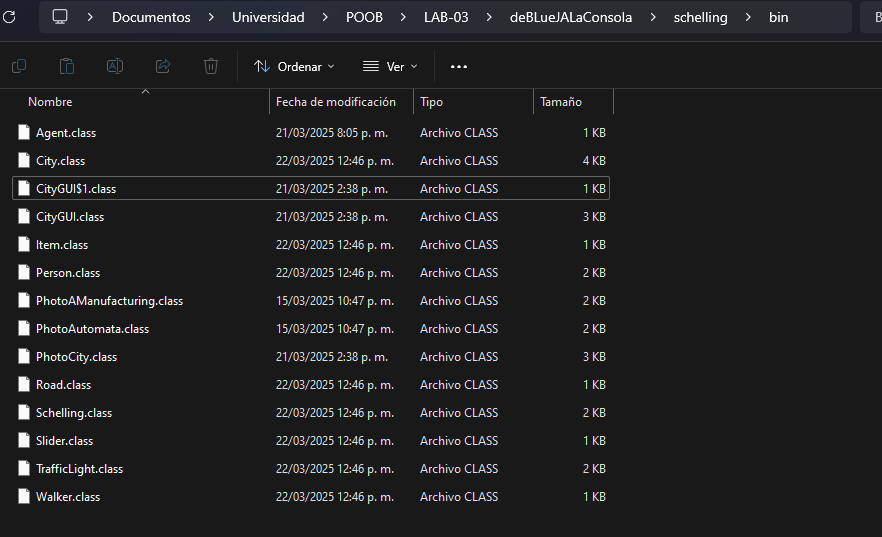
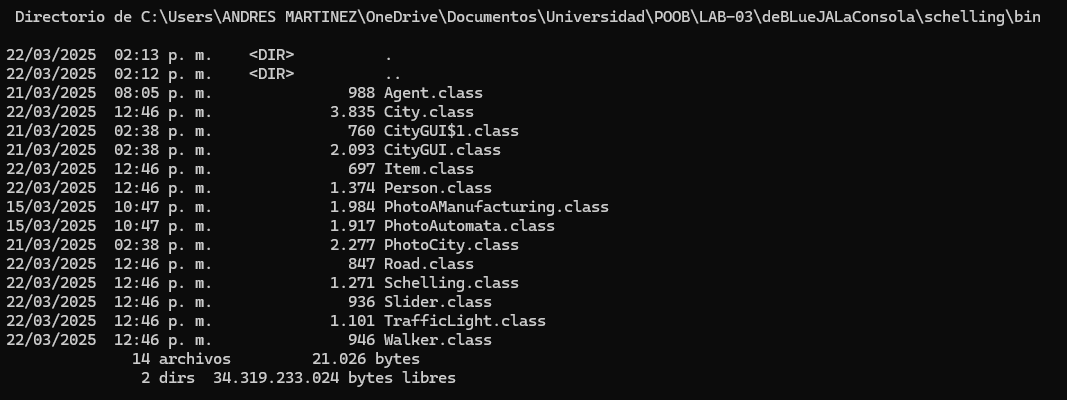


* src (source): Contiene el código fuente en subcarpetas organizadas por paquetes, aqui encontraremos el dominio, la presentación y test.
* bin (binary): Contiene los archivos .class generados tras compilar el código que esta en src.
* docs (documentación): Contiene la documentación generada con javadoc.

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

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

Del proyecto original solo deberia copiar los archivos generados tras compilar el codigo, estos siendo los .class ya que es lo que debe ir en bin.



**Comandos de java**

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



Javac es el compilador de java. Convierte archivos fuente.java en archivos de bytecode .class, que pueden ser ejecutados por la máquina virtual de java (JVM).

“javac HolaMundo.java”

Java es el intérprete de este. Se usa para ejecutar programas .class compilados con javac.

“java HolaMundo”

Javadoc genera documentación HTML a partir de los comentarios en el código fuente.

“javadoc -d docs HolaMundo.java”

Jar empaqueta varios archivos .class en un solo archivo .jar (Java ARchive). Se usa para distribuir aplicaciones o bibliotecas.

“jar cfe HolaMundo.jar MiPrograma \*. class”

C -> Crea un archivo .jar

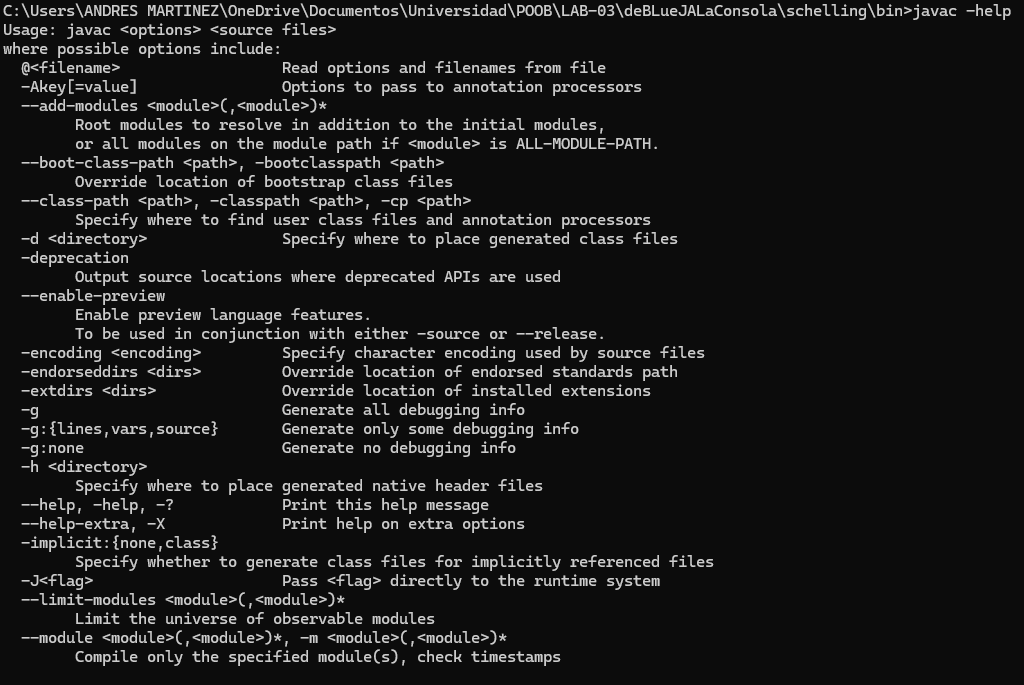
F -> Especifica el nombre del archivo .jar

E -> Especifica la clase principal con main()

\*.class -> Agrega todos los .class al .jar

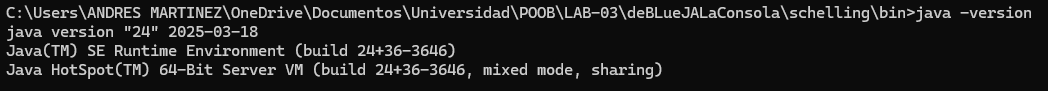
2. Cree una sesión de consola y consulte en línea las opciones de los comandos java y

javac. Capture las pantallas.



3. Busque la opción que sirve para conocer la versión a que corresponden estos dos

comandos. Documente el resultado.



**Compilando**

1. Utilizando el comando javac, desde el directorio raiz (desde schelling con una

sóla 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

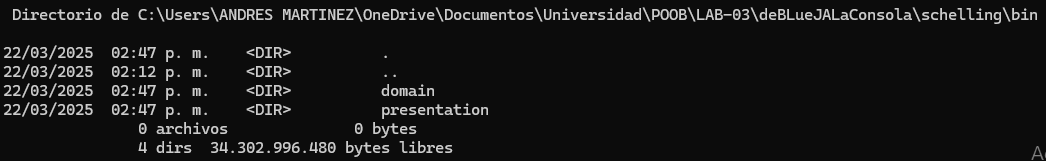
comando y que los archivos compilados deben quedar en los directorios respectivos.

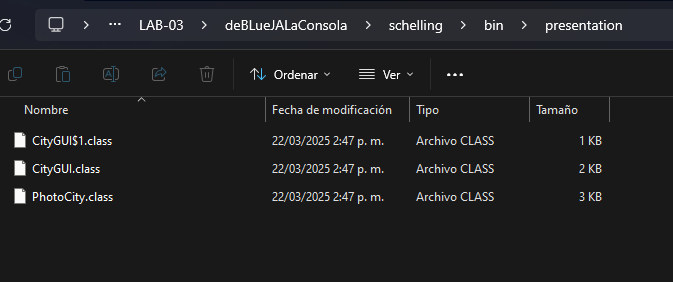
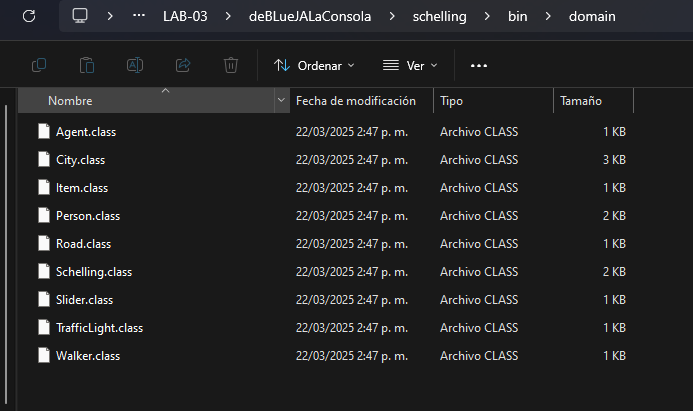
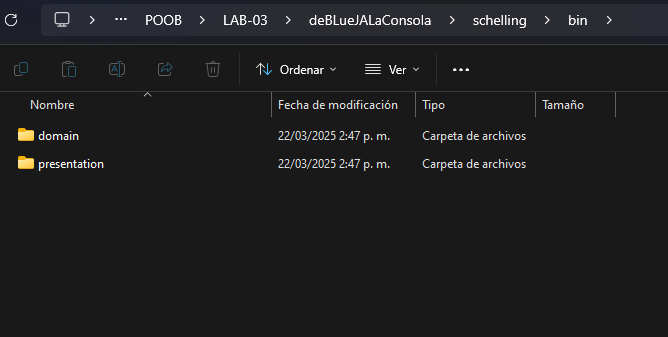
javac -d bin -sourcepath src src/presentation/ \* .java

Con esta instrucción estamos diciendo que ejecute src y lo guarde en bin.

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

nuevos archivos aparecen ahora y dónde se ubican?

Revisando el nuevo contenido, observamos que se crearon todos los archivos .class, que son los generados tras compilar el codigo que esta en src. Tambien vemos que se guardan en sus respectivas carpetas:



**Documentando**

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?

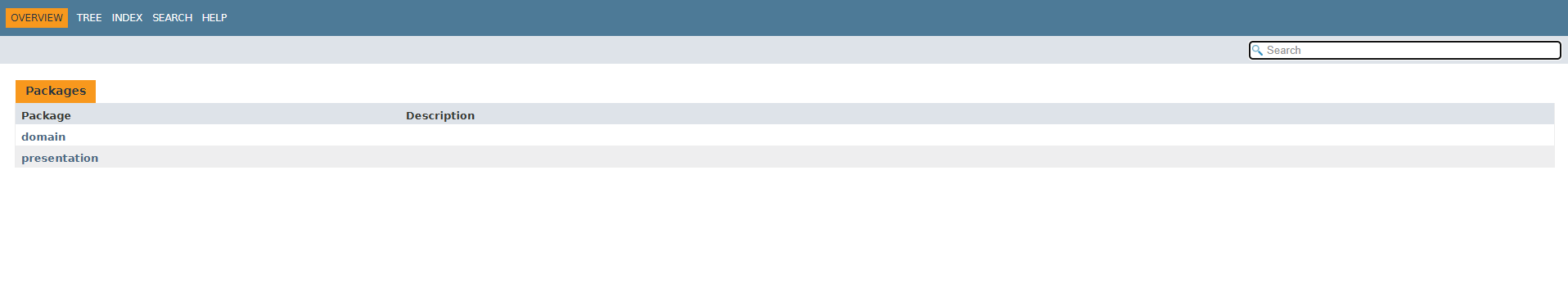
javadoc -d docs -cp src -subpackages presentation domain

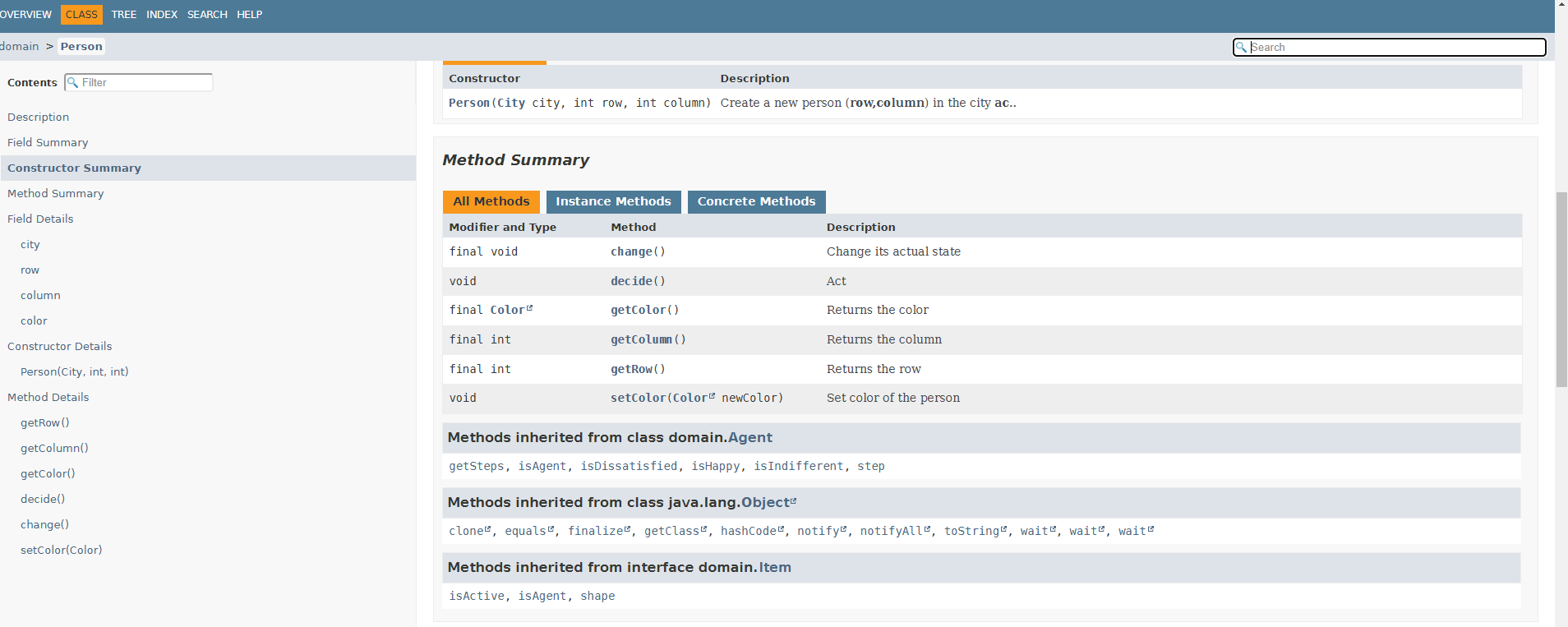
Este comando nos va a permitir generar la documentación del proyecto, lo vamos a guardar en un directorio llamado docs y le estamos especificando con –cp la ruta de busqueda de archivos .class y bibliotecas .jar luego le especificamos la fuente y los subpaquetes que serían presentacion y domain que es con lo que se va a generar la documentación.

2. ¿Cuál archivo hay qué 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. Al abrirlo podemos ver

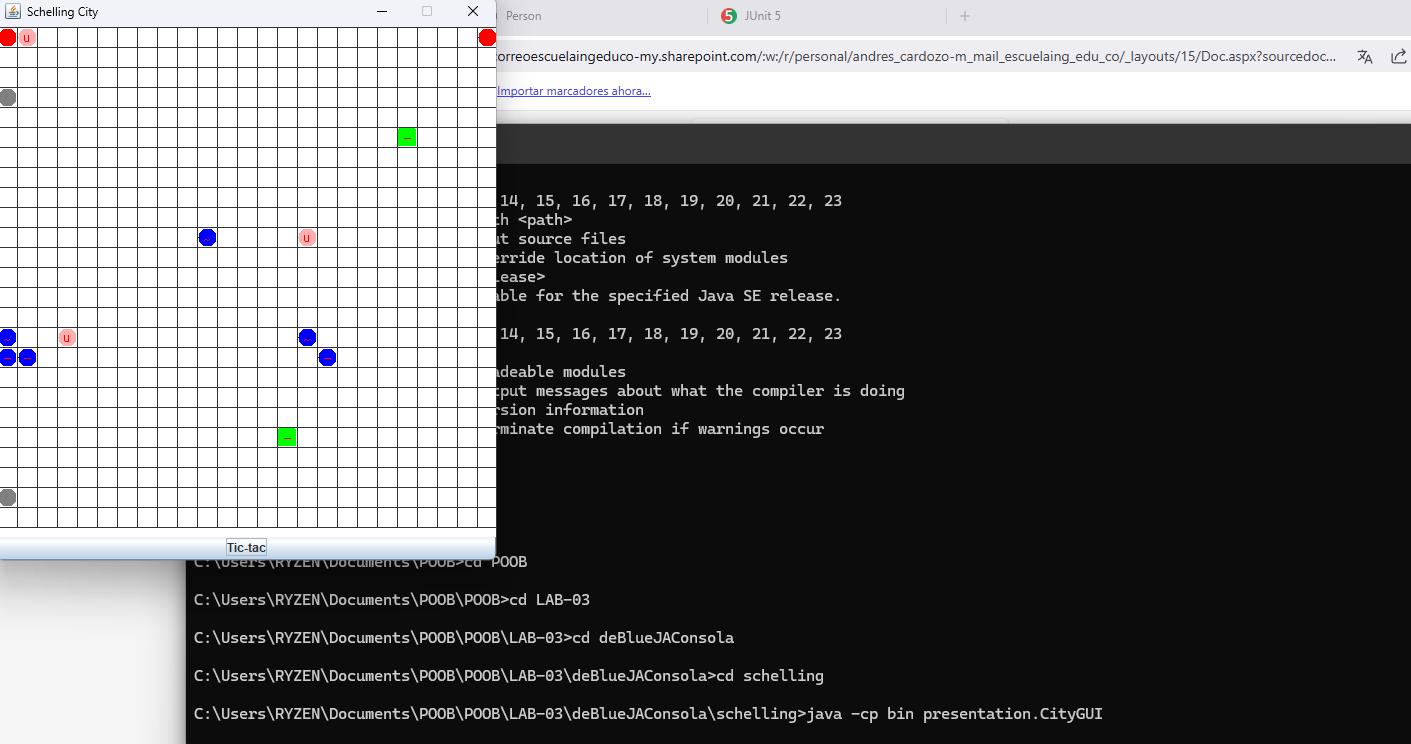




**Ejecutando**

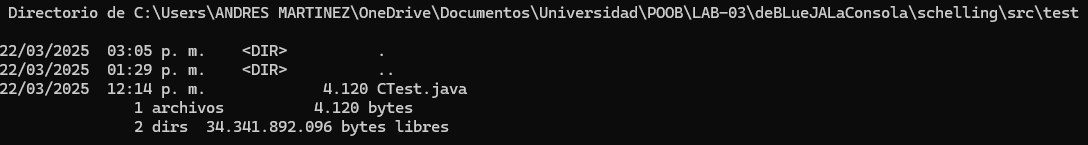
4. Empleando el comando java, desde el directorio raiz, ejecute el programa. ¿Cómo

utilizó este comando?



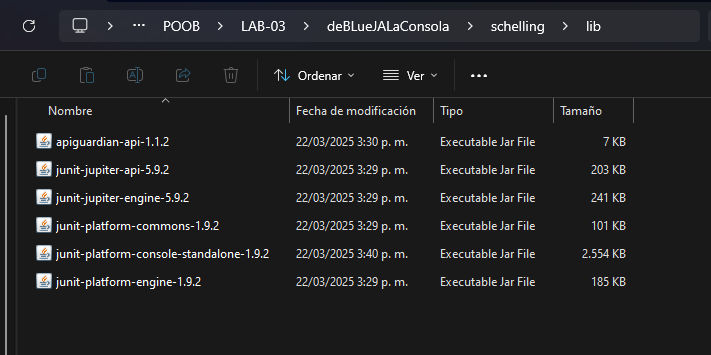
**Probando**1

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?



1. 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.

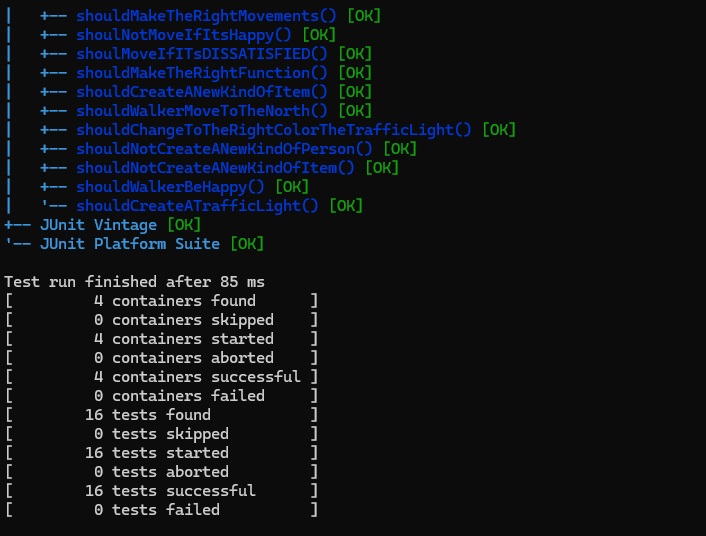
Consultando nos tocaba agregar una libreria de JUnit:



Para compilar las pruebas usamos: javac -d bin -cp "bin;lib/*" src/test/*.java

Y para probar usamos el comando: java -jar lib/junit-platform-console-standalone-1.9.2.jar --class-path bin --scan-class-path

1. Pegue en su documento el resultado de las pruebas



**Empaquetando [En lab03.doc]**

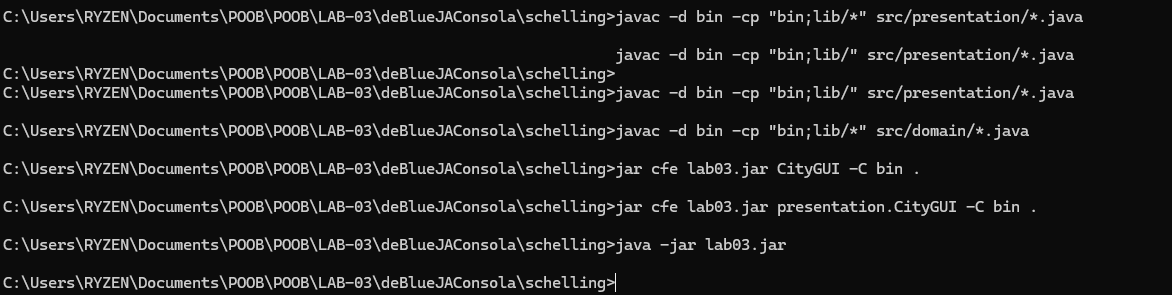
1. Consulte como utilizar desde consola el comando jar para empaquetar su programa

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 ?



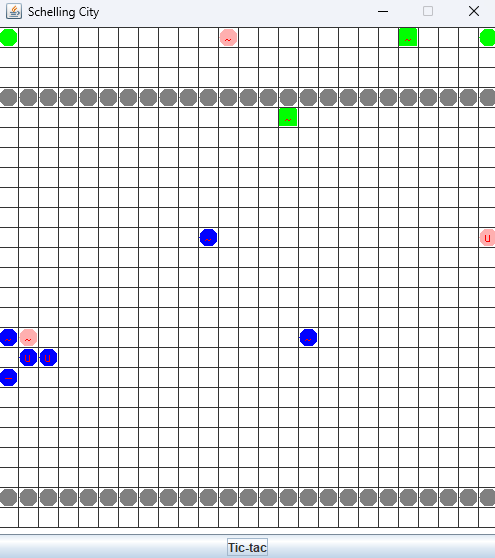
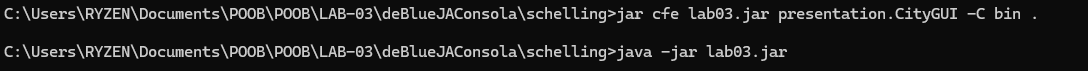
javac -d bin -cp "bin;lib/*" src/presentation/\**.java

javac -d bin -cp "bin;lib/*" src/domain/\**.java

jar cfe lab03.jar presentation.CityGUI -C bin .

Debemos especificar el paquete que contiene la clase principal (Main)

2. ¿Cómo se ejecuta el proyecto empaquetado?

cl

**RETROSPECTIVA**

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

Hombre)

Andrés Cardozo: 20 Horas

Tulio Riaño: 20 Horas

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

estado)

El estado actual del laboratorio es completo, ya que desarrollamos de manera acertada cada uno de los puntos propuestos incluyendo el bono.

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

Importe?

Las prácticas XP que se presentaron son importantes ya que nos permite una mayor comprensión del código y de la manera en que este se escribe (respetando los respectivos estándares). Por otra parte, las pruebas nos permiten ver que lo que estamos desarrollando sea de manera correcta y no llegue a presentarse ningún inconveniente.

4. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor

problema? ¿Qué hicieron para resolverlo?

Nuestro mayor logro fue el manejo de todo lo relacionado al proyecto mediante la consola, ya que tuvimos que buscar diferentes fuentes para entender cómo funcionaba los diferentes comandos y especificamente para ejecutar las pruebas ya que teníamos que descargar ciertos archivos de JUnit 5.

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

Resultados?

De nuevo resaltamos la comunicación y disposición para desarrollar el laboratorio, así mismo mantenemos nuestro compromiso por la ayuda mutua que se evidencia en el desempeño.

6. ¿Qué referencias usaron? ¿Cuál fue la más útil?

Incluyan citas con estándares adecuados.

**Referencias**

1. <https://dar10comyr.blogspot.com/2014/10/jframe.html>
2. <https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/javax/swing/JFrame.html>
3. <https://www.datacamp.com/es/doc/java/package>
4. <https://www.datacamp.com/es/doc/java/import>
5. <https://colorizer.org/>
6. <https://developer.mozilla.org/es/docs/Web/API/console>
7. <https://www.xataka.com/basics/comandos-basicos-para-dar-tus-primeros-pasos-consola-windows-cmd>
8. <https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html>
9. <https://labex.io/tutorials/java-how-to-manage-java-project-structure-419476>
10. <https://repo1.maven.org/maven2/org/junit/jupiter/junit-jupiter-api/5.9.2/>
11. <https://repo1.maven.org/maven2/org/junit/jupiter/junit-jupiter-engine/5.9.2/>
12. <https://repo1.maven.org/maven2/org/junit/platform/junit-platform-commons/1.9.2/>
13. <https://repo1.maven.org/maven2/org/junit/platform/junit-platform-engine/1.9.2/>
14. <https://repo1.maven.org/maven2/org/apiguardian/apiguardian-api/1.1.2/>
15. <https://repo1.maven.org/maven2/org/junit/platform/junit-platform-console-standalone/1.9.2/>
16. <https://stackoverflow.com/questions/2235276/how-to-run-junit-test-cases-from-the-command-line>
17. <https://stackoverflow.com/questions/2235276/how-to-run-junit-test-cases-from-the-command-line>