PROGRAMACIÓN ORIENTADA A OBJETOS

## Herencia e interfaces

## ADEMAS Java desde consola2022-1

## Laboratorio 3/6

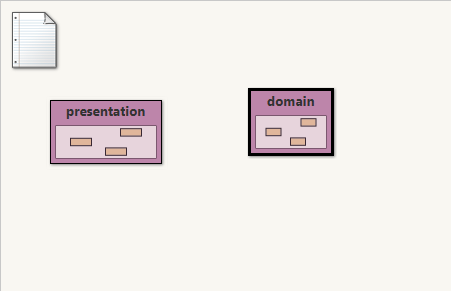
## Andrés Camilo Oñate Quimbayo, Nicolas Ariza Barbosa

## A. Conociendo

1. En el directorio descarguen los archivos contenidos en automata.zip. Revisen el código de la aplicación.

a) ¿Cuántos paquetes tiene?

En total el código de la aplicación posee dos paquetes: *presentation y domain*



b) ¿Cuál es el propósito del paquete presentación?

Su propósito es generar una graphical user interface que permita a los usuarios interactuar con los objetos.

c) ¿Cuál es el propósito del paquete dominio?

Dominio se encarga de predefinir las reglas del autómata y comportamiento de cada célula en cada casilla.

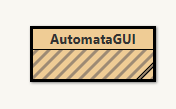
1. Revisen el paquete de dominio
2. ¿Cuáles son los diferentes tipos de componentes de este paquete?

El paquete tiene componentes de tipo interface (Item), dos clases concretas (Cell y CellularAutomata) y una clase abstracta(Agent).

* + La clase Cell tiene una relación de uso y de herencia con la Agent, relación de atributo con la clase CellularAutomata e implementa la interface Item.
  + La clase CellularAutomata tiene una relación de atributo con la interface Item y de uso con la clase Cell.

1. ¿Qué implica cada uno de estos tipos de componentes?
   * Las Clases Cell y CellularAutomata pueden crear objectos.
   * La clase Agent no puede ser instanciada y contiene un método abstracto.
   * La interface Item puede ser implementada por cualquier clase.
2. Revisen el paquete de presentación
3. ¿Cuántos componentes tiene?

Solo posee un componente y es la clase *AutomataGUI*



1. ¿Cuántos métodos ofrece?

La clase *AutomataGUI* ofrece un total de 6 métodos

- *main*

- *getWindows*

*- getOwnerlessWindows*

*- getFrames*

*- SetDefaultLookAndFeelDecorated*

*- IsDefaultLookAndFeelDecorated*

1. Para ejecutar un programa en java
2. ¿Qué método se debe ejecutar?

El método main. El método main en java es un estándar utilizado por la JVM para iniciar la ejecución de cualquier programa Java

1. ¿En qué paquete de autómata se encuentra? ¿En qué clase?

En presentación, clase AutomataGUI.

1. ¿Por qué?

Ya que este paquete se encarga de la interface gráfica, tiene sentido que el programa se ejecute haciendo el llamado al método main en la clase AutomataGUI.

1. Ejecuten el programa.
2. ¿Qué funcionalidades ofrece?

Ofrece una operación Tic-Tac.

1. ¿Qué hace actualmente?

Por el momento no se hace nada.

1. ¿Por qué?

Algunas funcionalidades no han sido completadas.

## B. Arquitectura general

1. Consulte el significado de las palabras package e import de java.
2. ¿Para qué sirve package?

Un paquete en Java se usa para agrupar clases relacionadas. Se puede pensar como una carpeta en un directorio de archivos. Usamos paquetes para evitar conflictos de nombres y para escribir un código mejor mantenible.

Los paquetes se dividen en dos categorías:

Paquetes integrados (paquetes de la API de Java)

Paquetes definidos por el usuario (cree sus propios paquetes)

1. ¿Para qué sirve import?

Para importar clases de un paquete se usa el comando import. Se puede importar una clase individual o bien, se puede importar las clases declaradas públicas de un paquete completo.

1. Explique su uso en esta aplicación.

Presentación importa las clases públicas del paquete dominio, además se usa para importar librerías del API de Java.

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

Contiene los subdirectorios:

- doc

- domain

- presentation

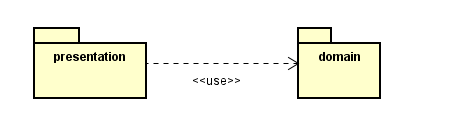
doc tiene archivos con extensiones .html, .txt, un subdirectorio presentation, aplicación y resources.

Domain y presentation contiene archivos con extensiones. Java, .Class, .ctxt y .bluej.

1. ¿Qué coincidencia hay entre paquetes y directorios?

Los directorios conservan el nombre del paquete y contiene todo lo relacionado con a las clases contenidas en el paquete.

1. 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.



## C. Arquitectura detallada.

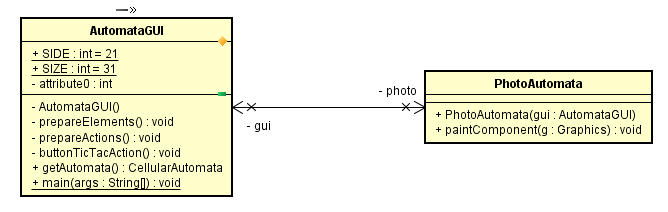
1. Para preparar el proyecto para BDD. Completen el diseño detallado del paquete de dominio. Adicionen el diagrama de clases en el paquete correspondiente.
2. ¿Qué componentes hacían falta?

La interfaz Item y la clase concreta Cell.

Diagrama

Descripción generada automáticamente

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?

La clase AutomataGUI es publica usa a PhotoAutomata como un atributo.

1. Adicione en las fuentes la clase de pruebas unitarias necesaria para BDD. (No lo adicione al diagrama de clases) ¿En qué paquete debe estar? ¿Por qué? ¿Asociado a qué clase? ¿Por qué?

Debe estar en el paquete de domain ya que este se encarga de definir la funcionalidad del programa. Asociado a la clase CellularAutomata, ya que es la clase principal del paquete.

## D. Ciclo 1. Iniciando con las células normales

1. Estudie la clase CellularAutomata ¿Qué tipo de colección usa para albergar los ítems? ¿Puede recibir células? ¿Por qué?

Usa un array doble (matriz) definido de tipo *Item* de tamaño 30x30, el cual es inicializado en vacío.

Texto

Descripción generada automáticamente

Este puede recibir células nuevas con el método *SetItem* el cual recibe una posición dentro de la matriz (tablero) y una célula nueva.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. Estudie el código asociado a la clase Cell ¿qué clases definen la clase Cell? Justifique sus respuestas.

¿en qué estado se crea?

En el constructor se definen:

1. El tablero
2. Una posición en el tablero
3. Un estado
4. Un color

Texto

Descripción generada automáticamente

¿Qué forma usa para pintarse?

Usa un círculo para pintarse.

Como estamos hablando de un paquete de clases, está en específico pertenece al back end de nuestro programa y lo que utiliza para dibujarse son las implementaciones en el front end y para ello debemos remitirnos al paquete *presentation.*

*Forma, Rectángulo

Descripción generada automáticamente*

En este se encuentra la clase *AutomataGUI* la cual importa el paquete *Domain* que es nuestro back end.

Texto, Carta

Descripción generada automáticamente

De parámetro en su método principal (main) la información del tablero y las células en este.

¿Cuándo aumenta la edad?

Al realizar el llamado al método turn() de la clase Agent, esto debido a las especificaciones del autómata.

¿Qué clases definen la clase Cell?

La clase Agent de la cual hereda sus atributos y métodos, ademas implementa la interfaz Item.

1. Cell por ser un Agent,

* ¿Qué atributos tiene?

Tiene atributos UNKNOWN, ALIVE, DEAD

Un atributo State al cual puede acceder y un atributo del cual accede mediante un método get.

* ¿Qué puede hacer (métodos)?

Llamados a los métodos de la clase Agent:

Cambiar estado, obtener la edad y saber si esta viva.

* ¿Qué decide hacer distinto?

Retornan sus propiedades como lo son: fila, columna, color tomar una decisión y cambiar de estado.

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

Los métodos implementados en la clase abstracta.

* ¿Qué debe aprender a hacer?

La célula debe aprender a decidir y a cambiar.

1. Por comportarse como un Item

* ¿Qué sabe hacer?

Obtener la forma

* ¿Qué decide hacer distinto?
* isAlive() y getColor()
* ¿Qué no puede hacer distinto?

decide()

* ¿Qué debe aprender a hacer?

Decidir, ya que el método no tiene cuerpo.

1. Ahora vamos a crear dos células en diferentes posiciones (1,1) (2,2) llámelos indiana y 007 usando el método someItems() .

Texto, Aplicación

Descripción generada automáticamente con confianza media

Ejecuten el programa, ¿Qué pasa con las células?

Son ubicadas dentro de la matriz en las posiciones especificadas, lo que se puede apreciar al ejecutar el programa.

Imagen que contiene biombo, edificio

Descripción generada automáticamente

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

Diseño: Se itera por todos los items de la matriz autómata, llamando al método decide () de la clase Cell (todos los ítems toman la decisión de su acción futura). Después, se llama al método change() para que estos cambien (luego todos cambian).

Texto

Descripción generada automáticamente

1. ¿Cómo quedarían indiana y 007 después de uno, dos y tres Tic-tac? Ejecuten el programa y hagan tres clics en el botón. Capturen una pantalla significativa. ¿Es correcto?

* Tic-tac 1 El atributo age cambia a 1, las células nacen.

Imagen que contiene biombo, edificio

Descripción generada automáticamente

* Tic-tac 2 El atributo age cambia a 2, células vivas.

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza baja

* Tic-tac 3 El atributo age cambia a 3, células vivas.

Una captura de pantalla de un celular

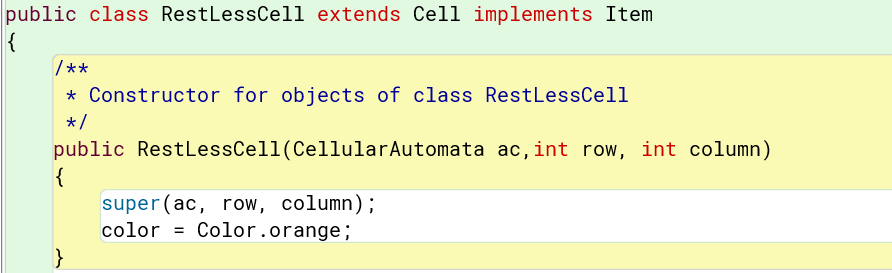
Descripción generada automáticamente con confianza baja

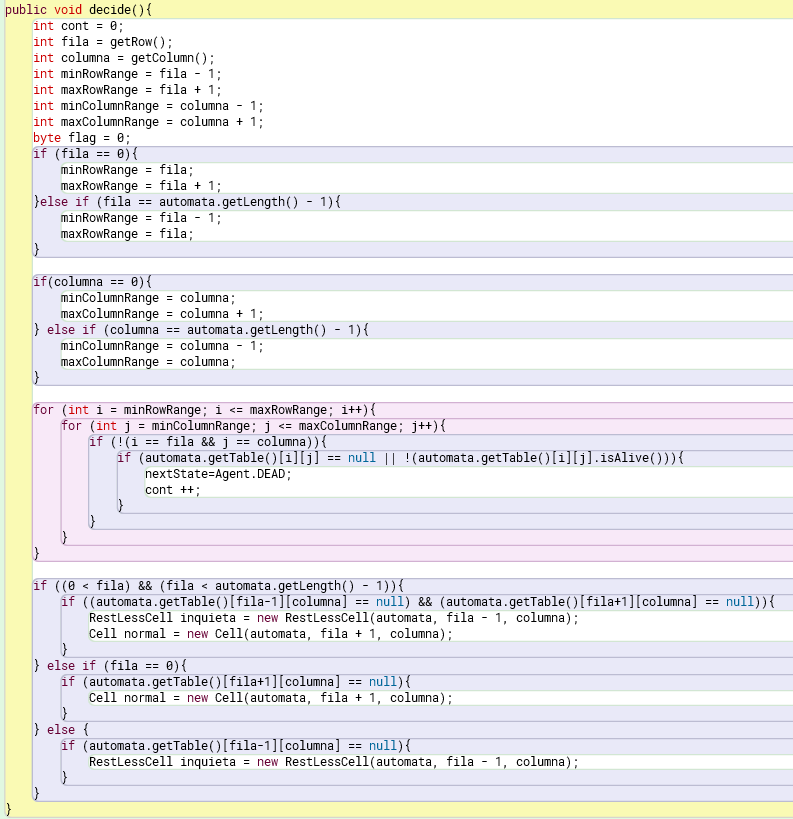
## E. Ciclo 2. Incluyendo a las células inquietas

El objetivo de este punto es permitir recibir en el autómata células inquietas. Las inquietas (i) son de color naranja; (ii) si alguna de sus vecinas está muerta, ella muere; (iii) si no tiene vecinas, genera una inquieta al norte y una normal al sur, si es posible.

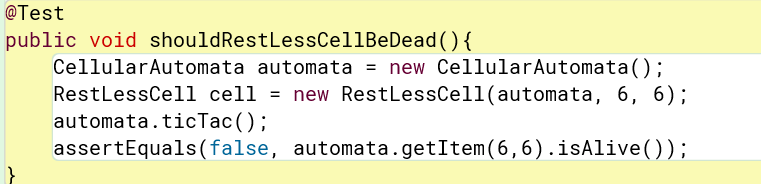
1. Para implementar esta nueva célula. ¿cuáles métodos se sobrescriben (overriding)? Diseñen, construyan y prueben esta nueva clase

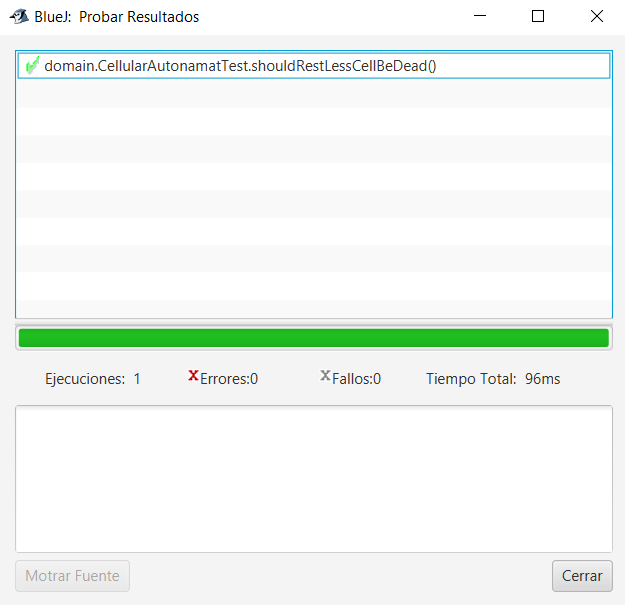
El método que se sobrescribió para esta práctica fue *decide* ya que nuestra nueva clase hereda la clase *Celula* e implementa la interfaz *Item,* además *decide*  funciona diferente para nuestra célula inquieta por su comportamiento.





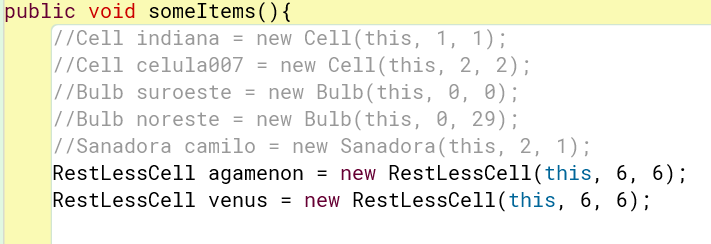
Para probar la nueva clase se ha implementado en el unit test de *CellularAutomata* la prueba shouldRestLessCellBeDead



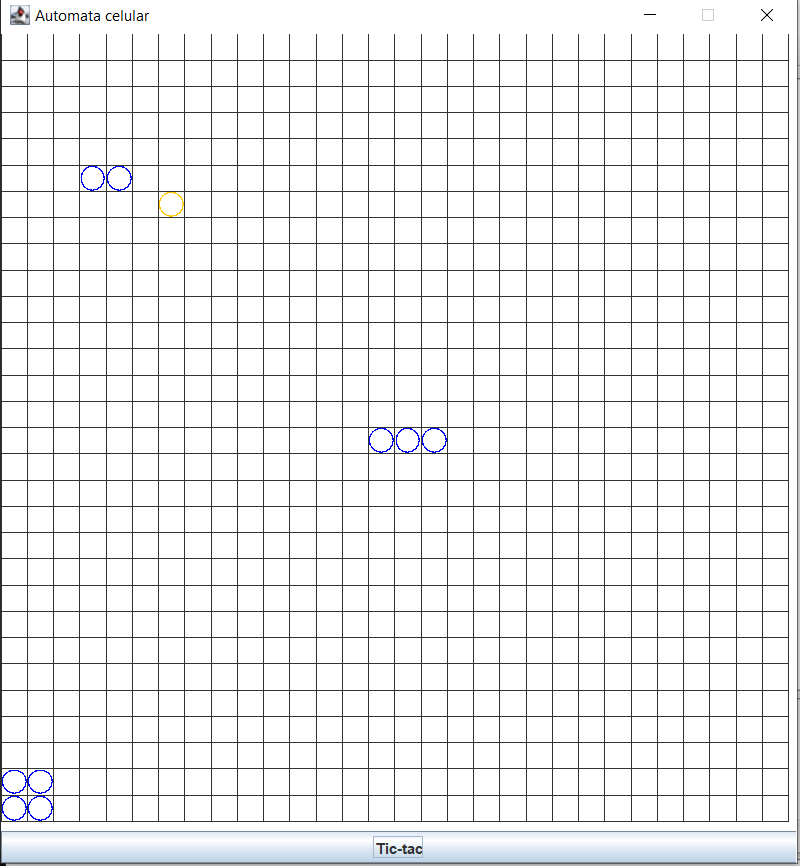


2. Adicione una pareja de inquietas, llámelas agamenon y venus

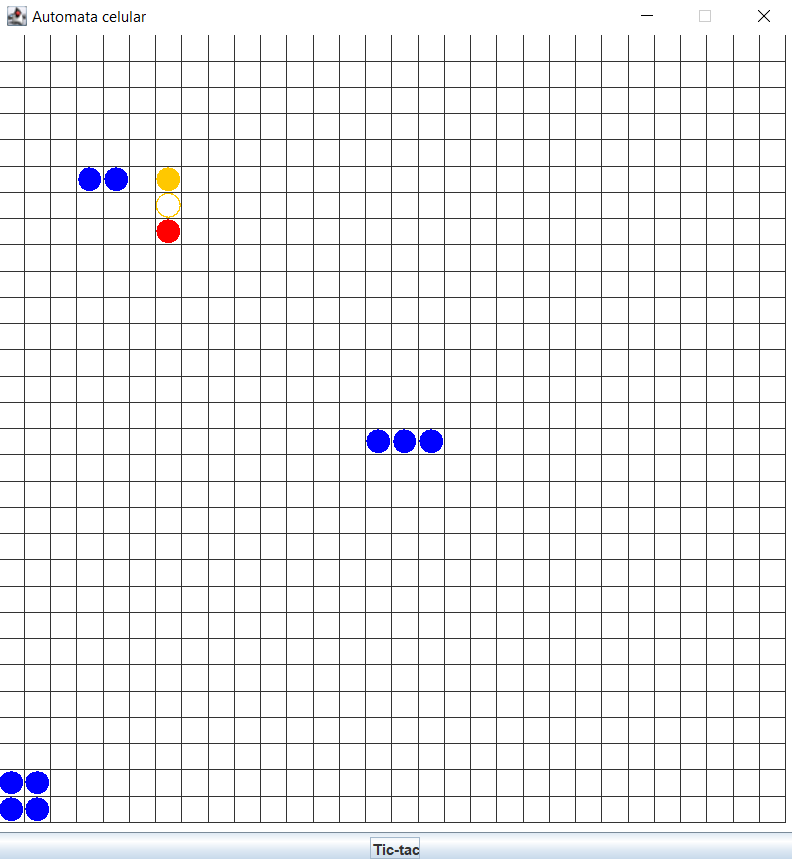
(a) ¿Cómo quedarían después de tres Tic-tac? Ejecuten el programa y hagan tres clics en el botón. Capturen una pantalla significativa. (b) ¿Es correcto?



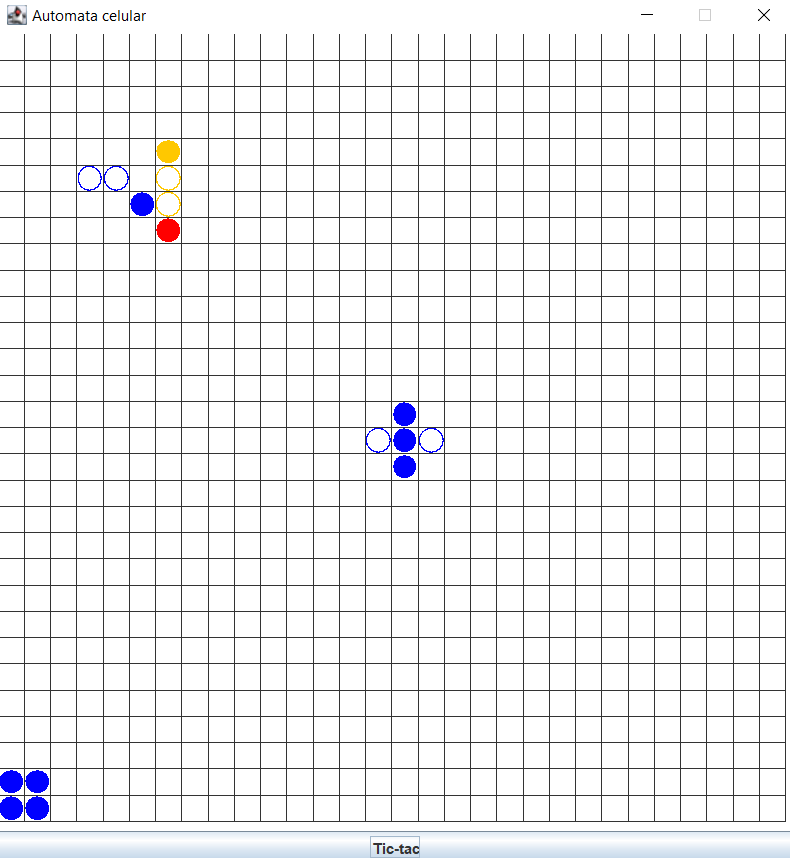
**Tic-Tac 0**



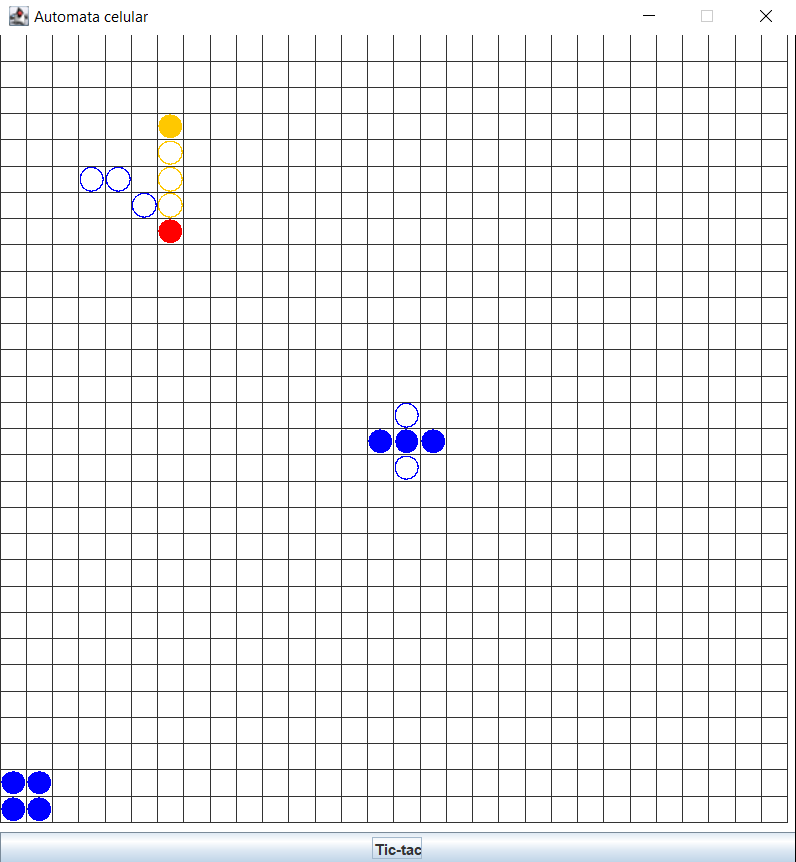
**Tic-Tac 1**



**Tic-Tac 2**



**Tic-Tac 3**



El comportamiento es correcto ya que nuestra célula inquieta está viva para Tic-Tac 0, en Tic-Tac 1 está al tener una de sus vecinas muertas ella muere además genera una nueva célula inquieta al norte y una nueva normal al sur.

En Tic-Tac 2 se la nueva inquieta muere y genera una nueva célula inquieta al norte mientras que al sur no se forma nada ya que existe otra célula en el espacio solicitado.

En Tic-Tac 3 ocurre lo mismo que en el Tic-Tac 2, la célula anterior muere y se genera una nueva célula inquieta al norte.

## F. Ciclo 3. Adicionando bombillos

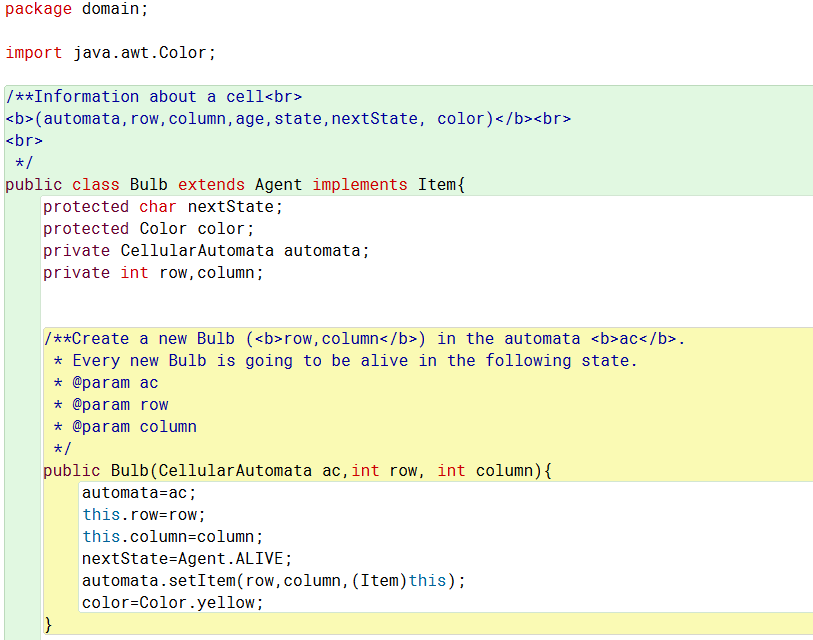
El objetivo de este punto es incluir en el CellularAutomata bombillos (sólo vamos a permitir el tipo básico de bombillos). Los bombillos son amarillos, cambian de forma de redondo a cuadrado y de cuadrado a redondo y después de 100 tics se funden (quedan redondos y grises).

1. Construyan la clase Bulb para poder adicionaría en el CellularAutomata? ¿debe cambiar en el código del CellularAutomata en algo? ¿por qué?

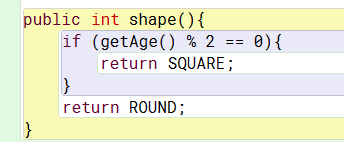
No se realizan cambios en esta clase debido a que esta se encarga de generar y actualizar el tablero de juego mediante el botón Tic tac. Se agregan en el método someItems() las líneas correspondientes a la creación de los bombillos en la matriz autómata.

2. Diseñen, construyan el método y prueben esta nueva clase.

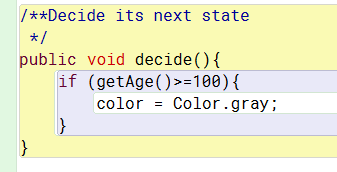
Bulb debe ser una extensión de la clase Abstracta Agent, ya que esta debe tener los comportamientos que en ella se describen, comparte métodos similares con la clase Cell, implementa la interface Item, pero especifica como se implementará el método shape().



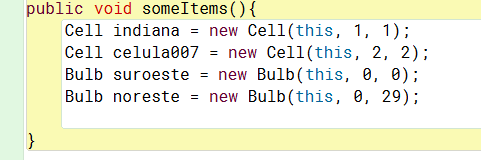
“Los bombillos son amarillos, cambian de forma de redondo a cuadrado y de cuadrado a redondo”



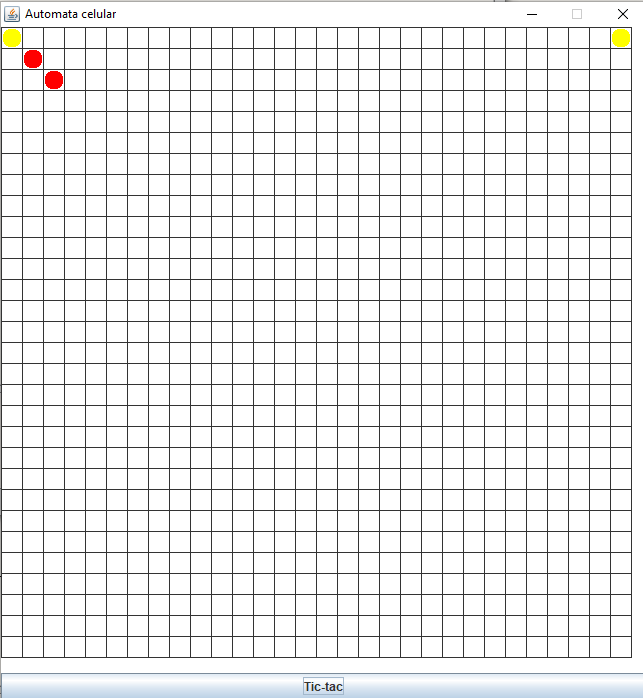
“después de 100 tics se funden (quedan redondos y grises)”



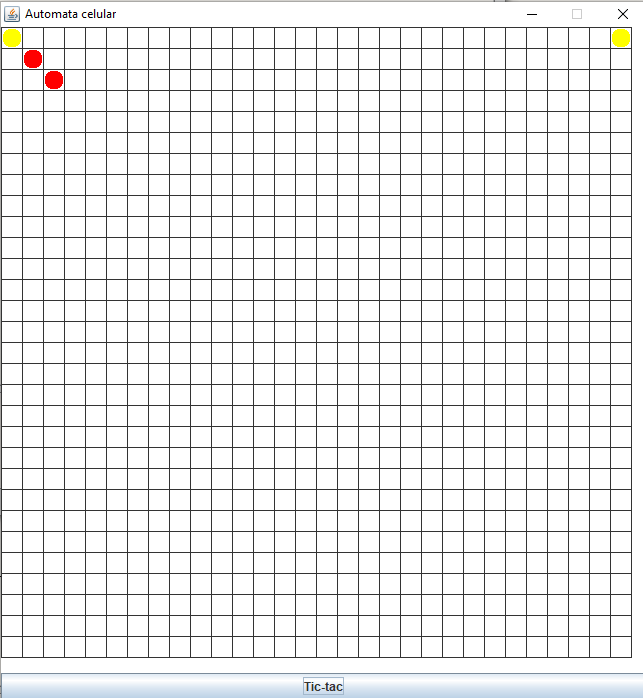
3. Adicionen dos bombillos cerca en las esquinas del CellularAutomata, llámenlas suroeste y noreste, (a) ¿Cómo quedarían después de tres Tic-tac? Ejecuten el programa y hagan tres clics en el botón. Capturen una pantalla significativa. (b) ¿Es correcto?



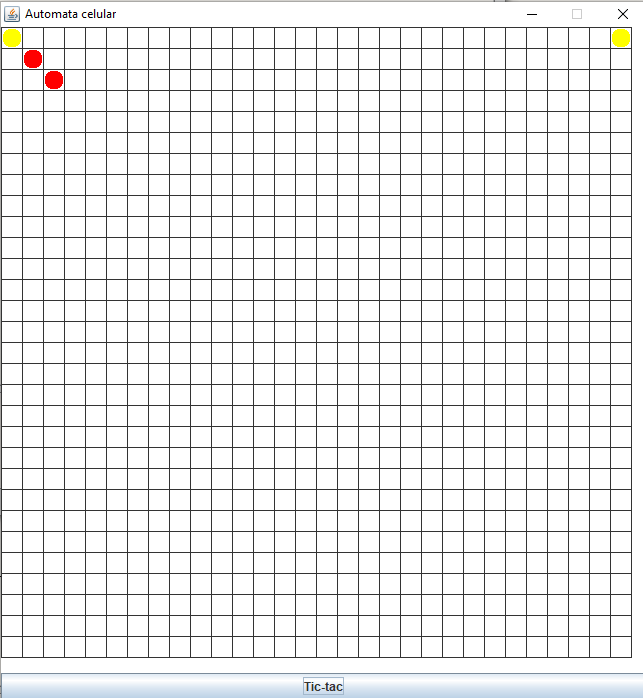
**Tic-tac 1**



**Tic-tac 2**



**Tic-tac 3**



Es correcto, cumple con las características del diseño.

## G. Ciclo 4. Nueva Célula: Proponiendo y diseñando

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

1. Propongan, describan e Implementen el nuevo tipo de células.

**Célula Sanadora:** Este tipo de célula nace después de 4 tics, si es posible, “sana” a totas sus células vecinas que este vivas.

Hereda atributos y métodos de la clase Cell, implementa la interfaz Ítem

Texto

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

1. Incluyan una pareja de ellas con el nombre de ustedes. Ejecuten el programa con dos casos significativos. Explique la intención de cada caso y capturen las pantallas correspondientes.

* Una célula sanadora tiene células al norte y al este. ¿Qué pasa con las células vecinas? ¿Qué pasa cuando la célula a la derecha tiene otra sanadora?

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza baja

Nacen:

Imagen de la pantalla de un celular

Descripción generada automáticamente con confianza media

Nacen las sanadoras:

Imagen que contiene biombo, juego, edificio, ventana

Descripción generada automáticamente

Imagen que contiene biombo, edificio, juego, oscuro

Descripción generada automáticamente

Cumplen su función y mueren

## H. Ciclo 5. Nuevo items: Proponiendo y diseñando

El objetivo de este punto es permitir recibir en un nuevo item (no célula) en el CellularAutomata.

* + - 1. Propongan, describan e Implementen un nuevo tipo de item.

Se propone el Item BombBulb (cuadrados de color Magenta), que después de dos tics tac elimina del tablero todos los otros ítems (Bombillos y células que estén vivos) que sean sus vecinos, incluyendo las diagonales. Al terminar su color pasa a ser negro

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

2. Incluyan un par de ellos con los nombres semánticos. Ejecuten el programa con dos casos significativos. Explique la intención de cada caso y capturen las pantallas correspondientes

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza bajaGráfico

Descripción generada automáticamente

## J. Caso 6. El Juego de la vida

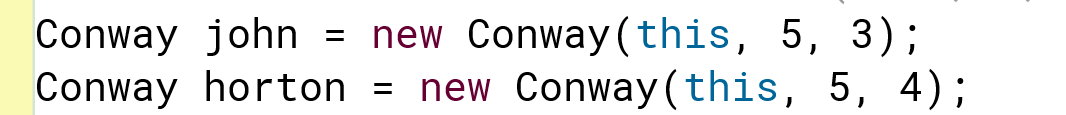
* + - 1. Si tenemos seguidas dos células Conway vivas en la misma fila, ¿qué debería pasar en el primer, segundo y tercer clic? ¿por qué? Escriba la prueba correspondiente.

Primer clic: Las células nacen

Segundo clic: Las células mueren (menos de dos vecinas)

Tercer clic: Las células permanecen muertas.

* + - 1. Para crear una célula Conway ¿Cuáles son las adiciones necesarias en el diseño? ¿y los cambios? ¡Hágalos! Ahora codifique. Estas células van a ser azules. ¿Las pruebas son correctas?
* Se agrega a la clase CellularAutomata el método cellBorn que evalúa las condiciones necesarias para que una célula nazca o reviva.
* Se crea la clase Conway como una subclase de Cell.
* Las pruebas son correctas.
  + - 1. Adicionen juntas en la fila cinco, una pareja de células Conway llámenlas john y horton. Ejecuten el programa, hagan tres clics en el botón Tic-tac y capturen la pantalla final.¿Qué pasa? ¿es correcto?



**Tic-tac 1** Las células nacen.

**Imagen que contiene biombo, edificio

Descripción generada automáticamente**

**Tic-tac 2** Mueren por “soledad” (Condición 2 del enunciado)

Una captura de pantalla de un celular con texto e imágenes

Descripción generada automáticamente con confianza baja

**Tic Tac 3** Permanecen muertas

Imagen que contiene biombo, edificio

Descripción generada automáticamente

* + - 1. Adicionen en la esquina inferior izquierda un Bloque y ejecuten la aplicación, ¿qué pasa? ¿queda estático? Capture una pantalla. No olviden escribir la prueba correspondiente

Tabla

Descripción generada automáticamente con confianza media

El bloque queda estático, las células no mueren dado que es un bloque.

Imagen que contiene biombo, edificio, ventana

Descripción generada automáticamente

* + - 1. Adicionen en la parte central inferior un Parpadeador (con espacio para parpadear) y ejecuten la aplicación, ¿qué pasa? ¿parpadea? Capture dos pantallas de parpadeo. No olviden escribir la prueba correspondiente.

Texto

Descripción generada automáticamente con confianza media

Tic Tac 1 Las células nacen:

Gráfico, Gráfico de burbujas

Descripción generada automáticamente

Tic Tac 2: Dos células Conway nacen, dado que están rodeadas por tres células vivas. Dos células (izquierda y derecha mueren por “soledad”)

Gráfico, Gráfico de burbujas

Descripción generada automáticamente

Tic tac 3: El comportamiento se repite, las células muertas tienen exactamente 3 células vivas vecinas, por lo que reviven. Las células del anterior tic tac mueren por “soledad”.

Una captura de pantalla de un celular con texto e imágenes

Descripción generada automáticamente con confianza baja

El patrón se repite.

## 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.
* Se va abre el menú del proyecto, se selecciona **Crear Archivo Jar,** se pide seleccionar una clase principal, en este caso será la encargada de ejecutar el programa AutomataGUI.
  + - 1. Consulte el comando Java para ejecutar un archivo jar. ejecutennlo ¿qué pasa?

Comando: java -jar “nombredelarchivo”.jar

Captura de pantalla de un celular

Descripción generada automáticamente

Se ejecuta el programa:

Gráfico

Descripción generada automáticamente

* + - 1. ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.
* El programa se convierte en un ejecutable de un solo archivo, lo que facilita el uso del programa para el usuario.

## 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.

En windows:

-Para crear directorios se hace uso del comando MKDIR

- Para eliminar directorios se hace uso del comando RMDIR

- Para copiar archivos o directorios se hace uso del comando COPY

- Para eliminar archivos se hace uso del comando DEL

- Para listar los archivos de un directorio se hace uso del comando DIR

* + - 1. 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

Texto

Descripción generada automáticamente

* + - 1. En el directorio copien únicamente los archivos \*.java del paquete de aplicación. Consulte y capture el contenido de src/aplicación

Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

## Estructura de proyectos java

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

* src: Código fuente \*.java
* bin: bytecodes de los códigos fuentes.
* doc: Documentación

¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué? Cópielos y consulte y capture el contenido del directorio que modificó.

* Lo archivos con extensión \*.class.

Tabla

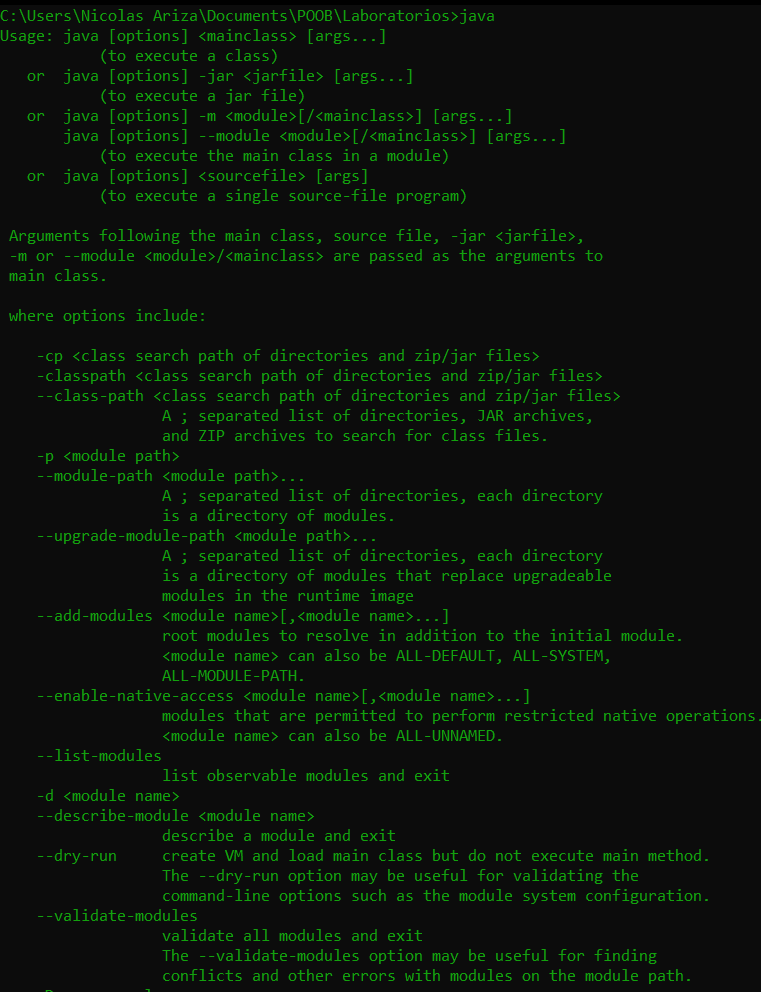
Descripción generada automáticamente

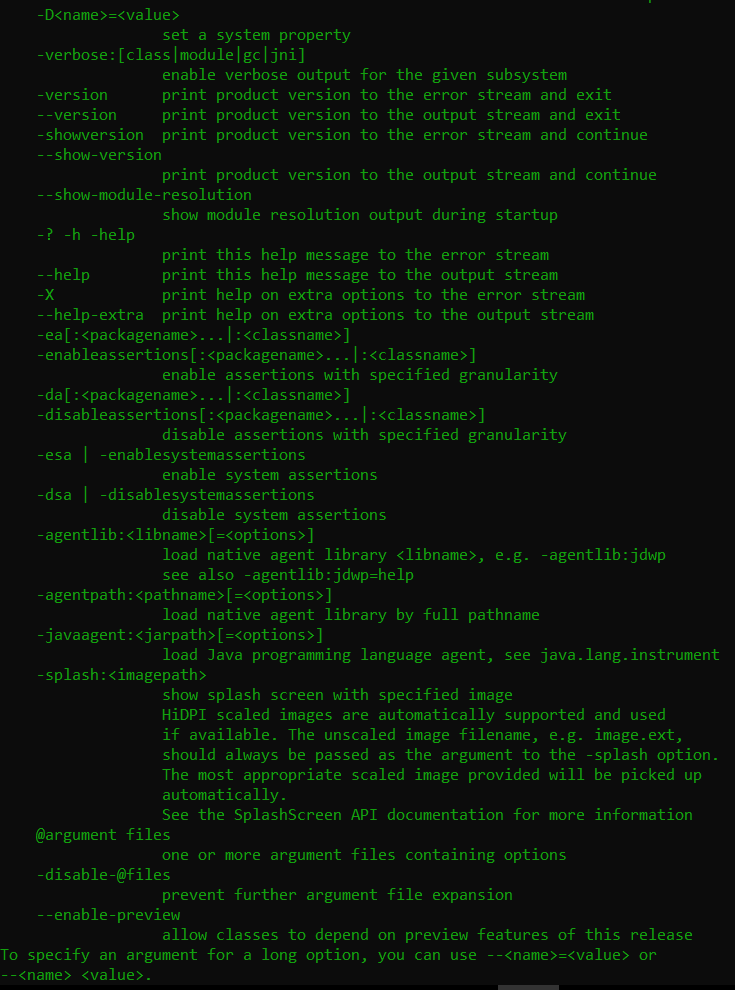
## Comandos de java

* + - 1. Consulte para qué sirven cada uno de los siguientes comandos
* Javac:

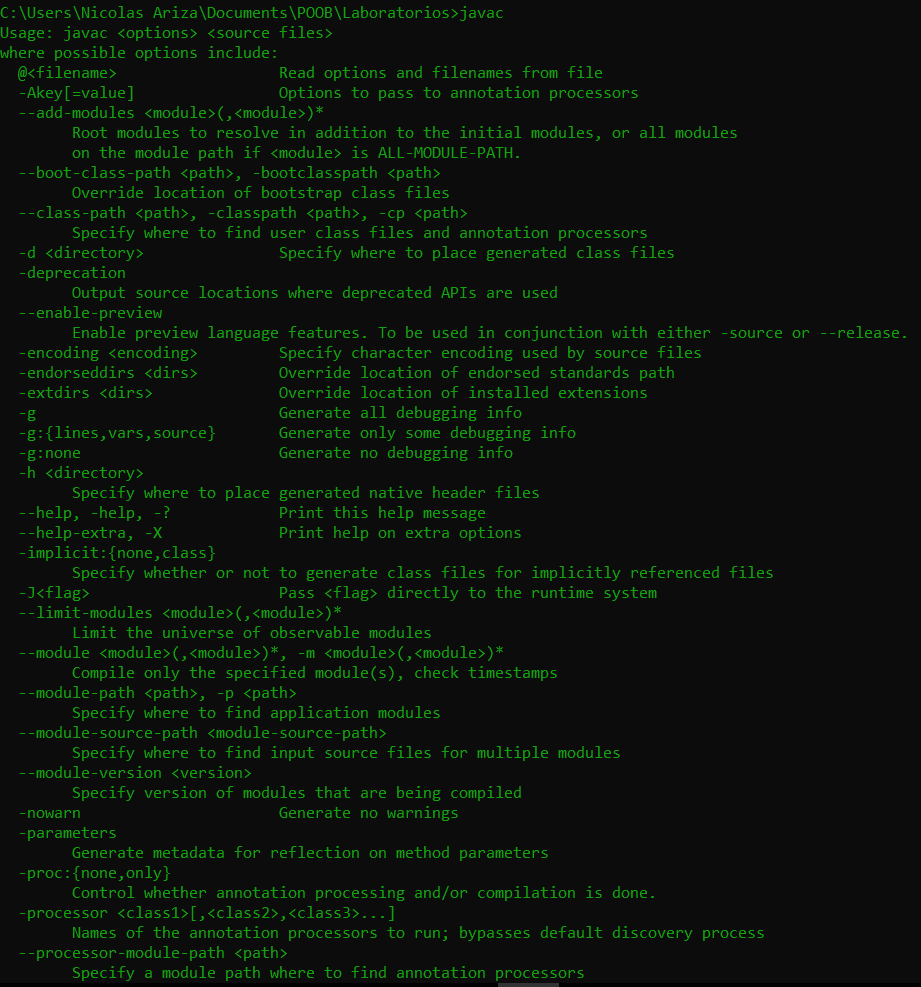
Es un comando que permite al sistema operativo compilar código fuente de Java a código de bytes o código de máquina que puede leer el procesador

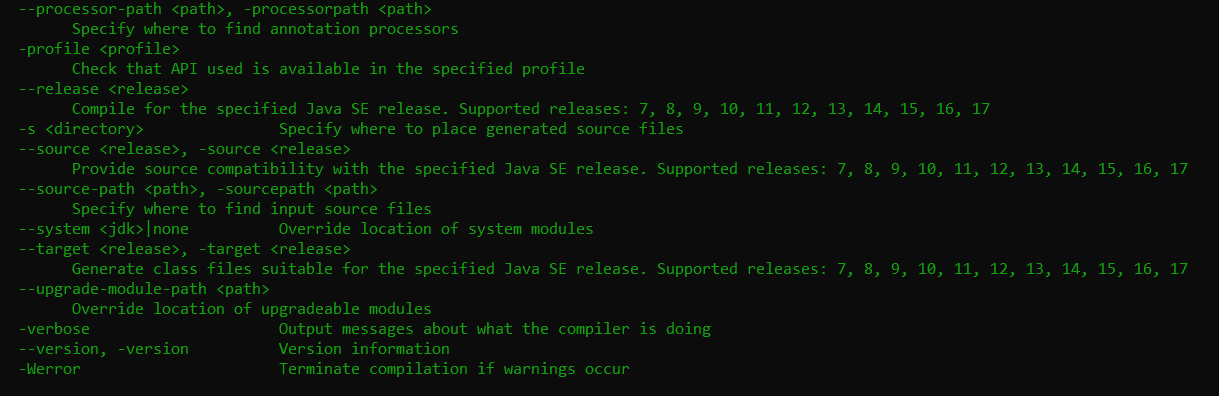
* Java: Es el comando encargado de llamar al intérprete de java que ejecuta los byte-codes creados por nuestro Javac (compilador). Funciona como vía de acceso hacia la máquina virtual Java.
* Javadoc: El comando se encarga de generar la documentación pertinente API para el paquete que hizo el llamado de esta instrucción.
* Jar: Permite la manipulación de los archivos de tipo Jar (java archive tool)
  + - 1. Cree una sesión de consola y consulte en línea las opciones de los comandos java y javac. Capture las pantallas.
* Java





* Javac





* + - 1. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.

Texto

Descripción generada automáticamente

## Compilando

* + - 1. Utilizando el comando javac, desde el directorio raiz (desde automata 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.

Instrucción:



* + - 1. Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles nuevos archivos aparecen ahora y dónde se ubican?

En el directorio bin aparecieron dos directorios, domain y presentatión, los cuales contienen los archivos .class del proyecto.

## 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?

Tabla

Descripción generada automáticamente con confianza media

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

Index

Tabla

Descripción generada automáticamente

Patrón de fondo

Descripción generada automáticamente

## Ejecutando

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

Sintaxis:

## 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?

Se descargaron los paquetes necesarios del internet y se ubicaron en el directorio raíz.

Texto

Descripción generada automáticamente con confianza media

Para incluir la librería junit se agrega el comando -cp

* + - 1. Ejecute desde consola las pruebas. ¿Cómo utilizó este comando? Puede ver ejemplos de cómo ejecutar el“test runner”en: <http://junit.sourceforge.net/doc/cookbook/cookbook.htm>

Comando:

* + - 1. Pegue en su documento el resultado de las pruebas

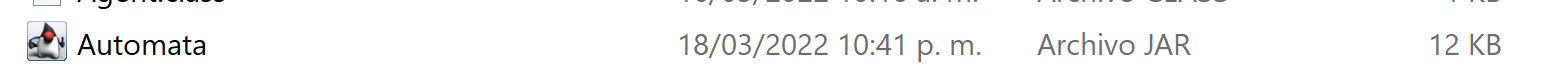
Texto

Descripción generada automáticamente con confianza baja

## Empaquetando

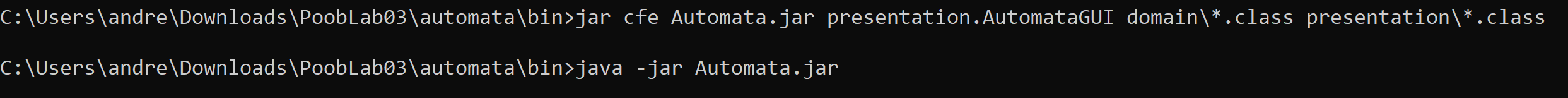
* + - 1. Consulte como utilizar desde consola el comando jar para empaquetar su programa entregable en un archivo .jar, que contenga los archivos bytecode necesarios (no las fuentes ni las clases de prueba), y que se pueda ejecutar al instalarlo en cualquier directorio, con solo tener la máquina virtual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

Comando:



* + - 1. ¿Cómo se ejecuta el proyecto empaquetado?

Comando:



RETROSPECTIVA

¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?

(12) (Horas/ Hombre)

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

El laboratorio se completó exitosamente.

Considerando las prácticas XP del laboratorio de hoy ¿por qué consideran que son importante?

Collective Ownership y integrate code at a time son importantes porque permite una mejor ejecución del laboratorio, en términos de optimización de tiempo, sin dejar atrás el desarrollo de las habilidades de los integrantes del grupo.

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

Lograr el comportamiento correcto de las células, describir adecuadamente los casos de frontera contra los que se topaban las células.

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

Planear adecuadamente la estrategia de relaciones entre las diferentes clases de los paquetes de este laboratorio