# Nombres:

# *Iván Camilo Rincón Saavedra*

# *Miguel Ángel Fúquene*

# PROGRAMACIÓN ORIENTADA A OBJETOS

**Herencia e interfaces ADEMAS Java desde consola 2020-1**

**Laboratorio 3/6**

**OBJETIVOS**

Desarrollar competencias básicas para:

1. Aprovechar los mecanismos de la herencia y el uso de interfaces.
2. Organizar las fuentes en paquetes.
3. Usar la utilidad jar de java para entregar una aplicación.
4. Extender una aplicación cumpliendo especificaciones de diseño, estándares y verificando su corrección.
5. Vivenciar las prácticas XP : The project is divided into [iterations](http://www.extremeprogramming.org/rules/iterative.html).
6. Utilizar los programas básicos de java (javac, java, javadoc, jar), desde la consola.

# ENTREGA

 Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.

 En el foro de entrega deben indicar el estado de avance de su laboratorio y los problemas pendientes por resolver.

 Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios preparados para tal fin.

**DESARROLLO**

# Contexto

Un autómata célular (A.C.)[1](#_bookmark0) es un modelo matemático para representar sistemas que puedan ser descritos como una colección masiva de objetos simples que interactúan localmente unos con otros y que evolucionan a pasos discretos. Sus caractísticas son:

1. La células se ubican en una rejilla, máximo una en cada celda.
2. Una célula puede estar en uno de un conjunto posible de estados. En nuestro caso: viva (●) o muerta(+).
3. Cada célula decide qué va suceder en la siguiente etapa de tiempo de acuerdo a su naturaleza, su estado y sus vecinas.
4. En cada momento, todas las células toman la decisión de su acción futura y luego todas cambian. Si hay nacimiento en su decisión, este será real al instante siguiente.

(ver wikipedia <http://es.wikipedia.org/wiki/automata_celular)>

**Conociendo** [En lab03.doc y automata.asta ]

1. En el directorio descarguen los archivos contenidos en automata.zip. Revisen el código de la aplicación a) ¿Cuántos paquetes tiene? b) ¿Cuántas clases tiene en total? ¿Cuántas tienen fuentes? c) ¿Cuál es la clase ejecutiva? ¿Por qué?
2. hay 2 paquetes, presentación y aplicación.
3. Existen 4 clases, AutomataCelular, celula, elemento,utomataGUI

AutomataCelular

Celula

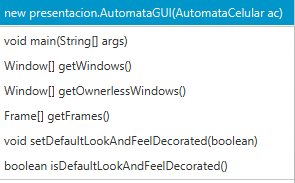
automataGUI

**c)** La clase ejecutiva es AutomataCelular

1. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?

Clase ejecutable es Automata GUI

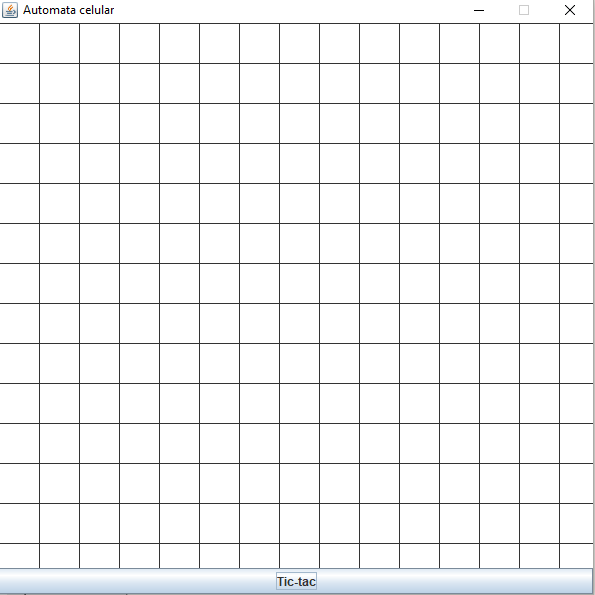
¿Qué funcionalidades ofrece?



¿Qué hace actualmente? ¿Por qué?

Únicamente genera el tablero en el cual se va a simular los autómatas celulares, porque es la única que utiliza librerías para interfaz gráfica, e instancia un tablero de 20x20 de autómatas cuando se llama el constructor.

Como se logra observar en la siguiente imagen.



**Arquitectura general.** [En lab03.doc y automataasta]

1. Consulte el significado de las palabras package e import de java. ¿Qué es un paquete? ¿Para qué sirve? Explique su uso en este programa.
   * **Package :**

Palabra reservada de java

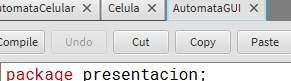
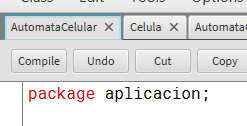
**package** *nombre del paquete;*

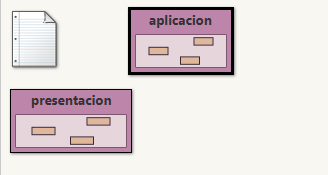
**¿Para qué sirve?**

Para el agrupamiento de clases, paquetes e interfaces, el cual permite un mejor modularidad del código **(subdivisión de una aplicación en partes más pequeñas llamadas módulos),** las cuales deben ser independientes del demás parte y la aplicación en sí **(División de tareas)** sin olvidar que los componentes que pertenecen a un paquete comparten un ámbito común.

**su uso en este programa.**

La separación de la creación de interfaz, con la implementación de la aplicación, con los siguientes comandos.





* + **Import:**

Palabra reservada de java

**import** *nombre del paquete.***\*;**

que es equivalente a decir impórteme todo del paquete *nombre del paquete*

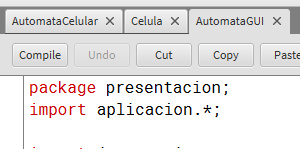
**¿Para qué sirve?**

Es usada con el fin de importar las definiciones de otro paquete, para poder usarlas en el procedimiento actual como:

* creando objetos
* accediendo a los servicios de las clases

**su uso en este programa.**

Dado que **AutomataGUI** necesita una implementación implícitamente también necesita del paquete aplicación para poder hacer cambios en la interfaz, este debe usar sus métodos y clases, este lo hace usando el siguiente código.



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

Ambos agrupan una serie de archivos, ya que un directorio puede ser una carpeta.

1. Inicie el diseño con un diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos.

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

1 [http://es.wikipedia.org/wiki/automataCelular\_célular](http://es.wikipedia.org/wiki/automataCelular_c%C3%A9lular)

**Arquitectura detallada.** [En lab03.doc y automataasta]

1. Usando ingeniería reversa prepararen el proyecto para **MDD**. Presente el diseño estructural actual de la aplicación (diagrama de clases). Las clases de la capa de presentación sólo deben tener los elementos públicos.
2. Adicione en las fuentes la clase de pruebas necesaria para **BDD.** (No lo adicione al diagrama de clases) ¿En qué paquete debe estar? ¿Por qué? ¿Asociado a qué clase?

¿Por qué?

**Ciclo 1. Iniciando con las células normales [En lab03.doc y \*.java]**

# (NO OLVIDE BDD - MDD)

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

Utiliza una Array de array´s para almacenar los elementos, si seria posible agregarla a la colección de elementos, porque la célula implementa la interfaz de elemento.

1. Estudie el código de la clase Celula, ¿qué otras clases la definen? ¿cómo?

La clase célula es definida por la interfaz Elemento, por la relación que existe entre estas y por la clase Autómata, ya que es necesaria un autómata para su creación.

1. Todas las células ¿qué saben hacer? ¿qué no puede hacer distinto? Justifique su respuesta.

Saben hacer:

**¿qué saben hacer?**

* getFila()
* getColumma()
* getColor()
* isVivo()
* edad()
* junto a los métodos de la interfaz

**¿qué no puede hacer distinto?**

* getFila()
* getColumma()
* getColor()
* isVivo()
* edad()
* Cambie()

Todos los métodos de Célula

**¿qué debe aprender a hacer?**

* + - getColor()

1. Por comportarse como un elemento, ¿qué sabe hacer? ¿qué no puede hacer distinto?

¿qué debe aprender a hacer? Justifique su respuesta.

Saben hacer:

**¿qué saben hacer?**

* decida()
* cambie()
* getForma()
* isVivo()

**¿qué no puede hacer distinto?**

* Todo puede ser sobreEscrito en la clase Elemento.

Todos los métodos de Célula

**¿qué debe aprender a hacer?**

* + - getColor()

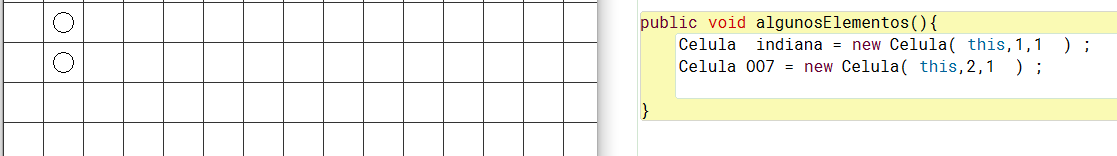
1. Considerando lo anterior, una Celula ¿de qué color es? ¿cómo decide? ¿cómo cambia? Justifiquen sus respuestas.

* La célula empieza sin un color ya que solo sale con los bordes de color negro, como se aprecia en la imagen.



* La célula decide su color según su estado futuro.
* La célula definirá su color según el estado actual que tenga

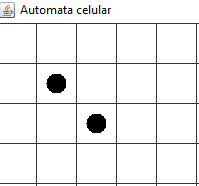
Ahora vamos a crear dos células en diferentes posiciones (1,1) (2,2) llámelos indiana y 007 usando el método algunosElementos() . Ejecuten el programa, ¿Cómo quedan todas las células? Capturen una pantalla significativa.

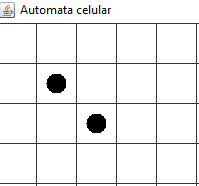


1. En este punto vamos a construir (diseño y código) el método que atiende el botón **Tic-tac**: el método llamado ticTac() de la clase AutomataCelular. ¿Cómo quedarían indiana y 007 después de uno, dos y tres **Tic-tac**? Escriba la prueba correspondiente.
2. Construyan el método. ¿Es correcto?

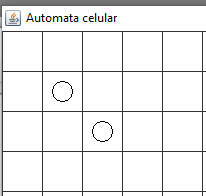
Si es correcto

1. Ejecuten el programa y hagan tres clics en el botón. ¿Como quedan las células? Capturen una pantalla significativa.

**Uno:**

**Dos:**

**Tres:**



**Ciclo 2. Incluyendo a las células izquierdosas** [En lab03.doc y automataasta]

# (NO OLVIDE BDD - MDD)

El objetivo de este punto es incluir en el autómata algunas células “izquierdosas”.Estas células son de color rojo y deciden morir si hay algún elemento vivo a su derecha (oeste).

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

* Para el primer click, ambas células se encuentran vivos
* Para el segundo click muere la célula de la izquierda
* Para el Tercer click se mantiene igual en el click2.

1. ¿Cuáles son las adiciones necesarias en el diseño? ¿y los cambios? ¡Hágalos! ¿cuáles métodos se sobre-escriben *(overriding)*? Ahora escriba el código correspondiente a la célula Izquierdosa ¿Las pruebas son correctas?

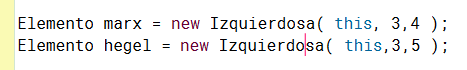
Se debe crear una nueva clase llamada Izquierdosa, la cual hereda de la clase célula, se debe sobre-escribir el método decidir ya que cada célula se comportará de forma diferente.

Dado que las células se comportan de manera diferente.

1. Para aceptar la célula Izquierdosa en AutomataCelular, ¿debe cambiar en el código del AutomataCelular en algo? ¿por qué?

En AutomataCelular solo se debe instanciar los dos objetos izquierdosa y no realizar ningún cambio.

1. Adicionen juntas una pareja de células izquierdosas en la fila 3, llámenlas marx y hegel, ¿Cómo quedarían después de uno, dos y tres **Tic-tac**? Escriba la prueba correspondiente.

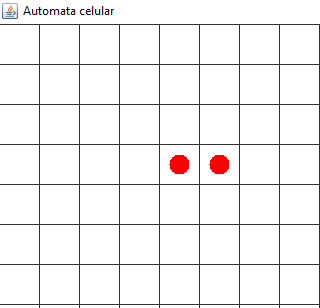


1. Construyan el método. ¿Es correcto?

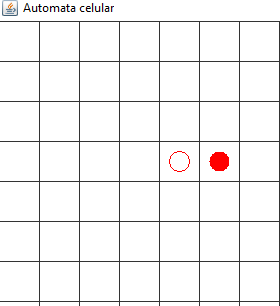
Es correcto

1. Ejecuten el programa y hagan dos clics en el botón. ¿Como quedan las células? Capturen una pantalla significativa.

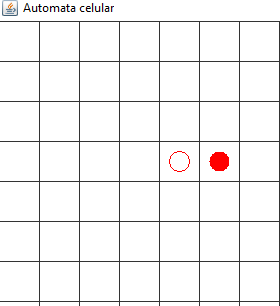
**Uno:**

****

**Dos:**

****

**Tres:**

****

**Ciclo 3. Adicionando una barrera** [En lab03.doc, automata.asta y \*.java]

El objetivo de este punto es incluir en el AutomataCelular barreras (sólo vamos a permitir un tipo de barreras). Las barreras son cuadradas, negras y como barreras están muertas: ni deciden ni cambian.

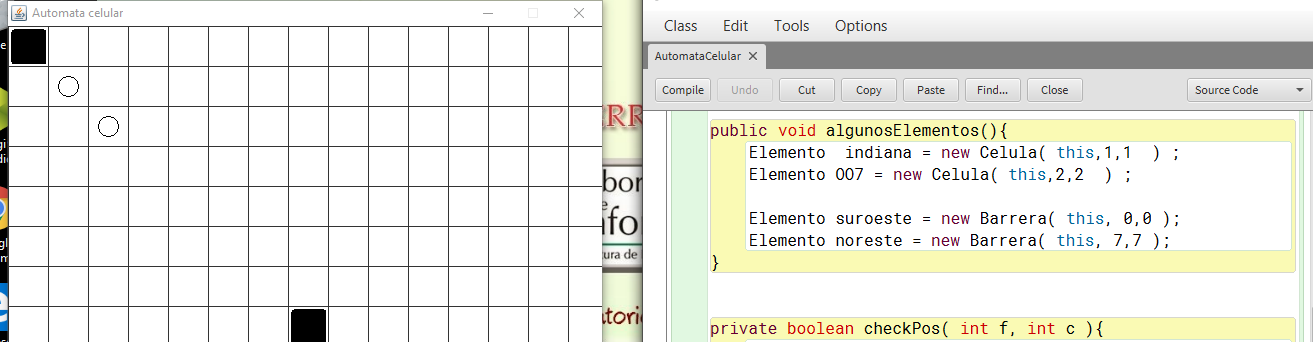
# (NO OLVIDE BDD – MDD)

1. Construyan la clase Barrera para poder adicionaría en el AutomataCelular ¿qué hicieron?

Se debe añadir una nueva clase llamada barrera, la cual implementa la interfaz Elemento.

1. Para aceptar este elemento, ¿debe cambiar en el código del AutomataCelular en algo? ¿por qué?

No debe cambiar nada, ya que en este caso la barrera implementa un elemento, lo único que se debe hacer es crear un nuevo elemento con los nombres correspondiente y modificar el método getForma() , para poder definir su forma en la interfaz.

1. Adicionen dos Barreras cerca en las esquinas del AutomataCelular, llámenlas suroeste y noreste, ¿Cómo quedarían después de uno, dos y tres **Tic-tac**? Escriba la prueba correspondiente.

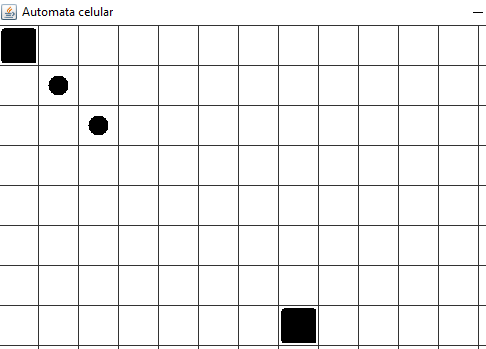
Debería quedarse estática en todas las iteraciones.

1. Construyan el método. ¿Es correcto?

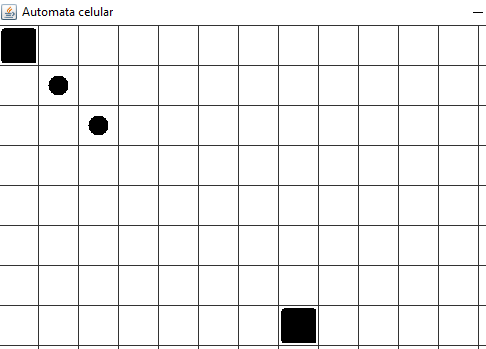
Si ya que se no se superponen células en las barreras

1. Ejecuten el programa y hagan tres clics en el botón. Capturen una pantalla significativa. ¿Qué pasa? ¿es correcto?

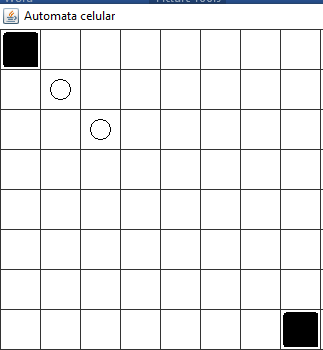
Uno:



Dos:



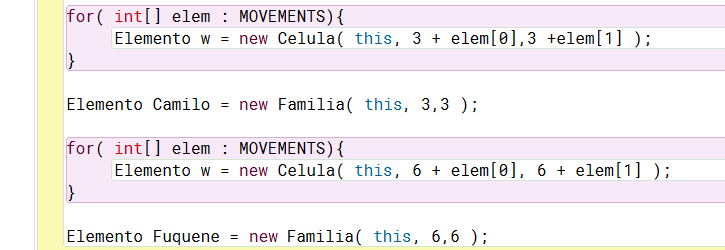
Tres:



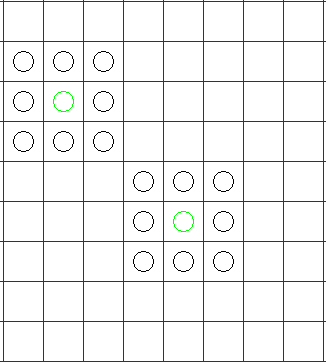
# Ciclo 4. Nueva Célula: Proponiendo y diseñando

El objetivo de este punto es permitir recibir en un nuevo tipo de célula (NO OLVIDE BDD - MDD)

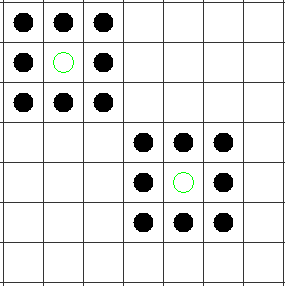
1. Propongan, describan e Implementen un nuevo tipo de células.
2. Incluyan una pareja de ellos con el nombre de ustedes. ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes.



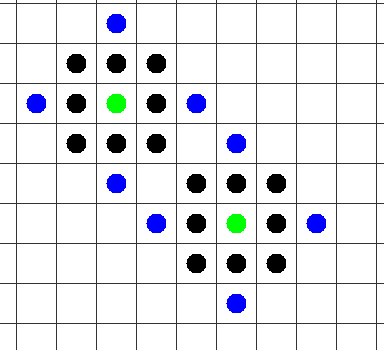
La nueva célula familia vive si y solo si está rodeada de cualquier tipo de célula

 Estado Inicial:

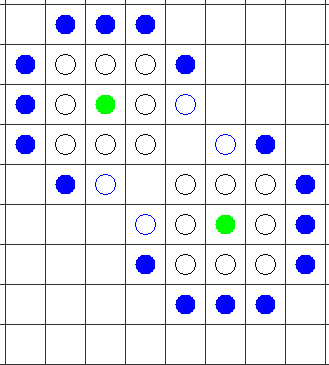
Primer Tic-Tac:



Segundo Tic-Tac



Tercer Tic-Tac



# Ciclo 5. Nuevo elemento: Proponiendo y diseñando

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

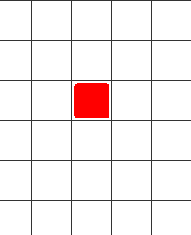
AutomataCelular.

(NO OLVIDE BDD - MDD)

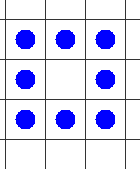
* + Propongan, describan e Implementen un nuevo tipo de elemento
* Incluyan un par de ellos con los nombres semánticos. ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes.

Se creo un elemento llamado Respawn, el cual en un tictac después de su creación genera células conway, en sus 8 posiciones vecinas.

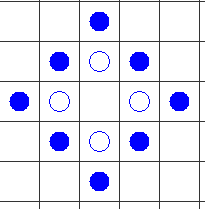
Estado Inicial:



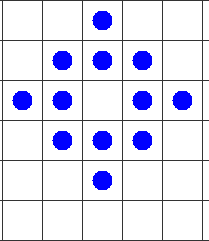
Uno:



Dos:



Tres:



# Caso 6. El Juego de la vida

El juego de la vida es el mejor ejemplo de un autómata célular, diseñado por el matemático británico John Horton Conway en 1970. Un AutomataCelular célular es una malla con células. Las células pueden estar vivas o muertas y pueden estar listas para vivir o para morir en el siguiente momento. Cada célula tiene como vecinas las que están próximas a ella, incluso en las diagonales.

En el juego de la vida el estado del AutomataCelular evoluciona a lo largo de unidades de tiempo y los cambios dependen del número de células vecinas vivas:

* + Una célula muerta con exactamente 3 células vecinas vivas "revive" (al tiempo siguiente estará viva).
  + Una célula viva con 2 ó 3 células vecinas vivas sigue viva.
  + Si la célula tiene una o más de tres vecinas muere o permanece muerta por "soledad" o superpoblación".
  + Si en el vecindario, hay una celda vacia rodeada por 3 células vivas “nace” una nueva cédula (al tiempo siguiente estará viva).

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

Existen numerosos tipos de patrones que pueden tener lugar en el juego de la vida:

El bloque y el barco son estáticos, el parpadeador y el sapo son osciladores y el planeador y la nave espacial ligera viajan por el AutomataCelular.



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.

Las células solo estarán vivas al en el primer tic-tac pero después de el de este ellas morirán por soledad ya que cada una solo tendrá una célula de vecina.

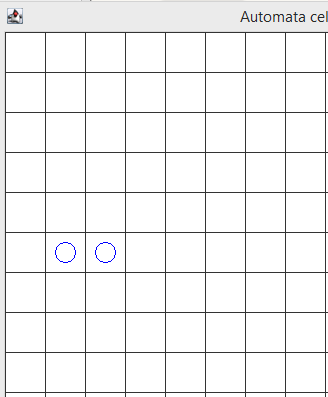
1. Para crear una célula Conway ¿Cuáles son las adiciones necesarias en el diseño? ¿y los cambios? ¡Hágalos! Ahora codifique. Estas células van a ser azules. ¿Las pruebas son correctas?

Se debe crear una nueva célula llamada conway, la cual se le cambiara el atributo color, que extienda de la clase célula, con un método decidir que sigue las reglas del juego de la vida.

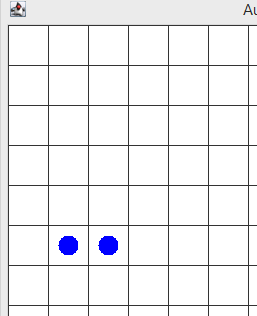
1. Adicionen juntas en la fila cinco, una pareja de células Conway llámenlas john y horton. Ejecuten el programa, hagan tres clics en el botón **Tic-tac** y capturen la pantalla final.¿Qué pasa? ¿es correcto?

Las células solo estarán vivas al en el primer tic-tac, pero después de este ellas morirán por soledad ya que cada una solo tendrá una célula de vecina.

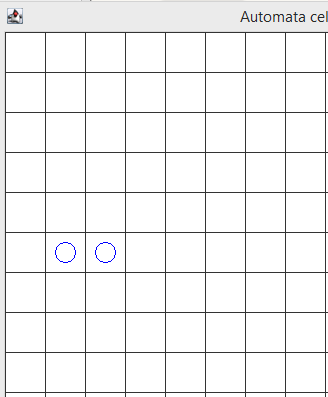
Estado Inicial.



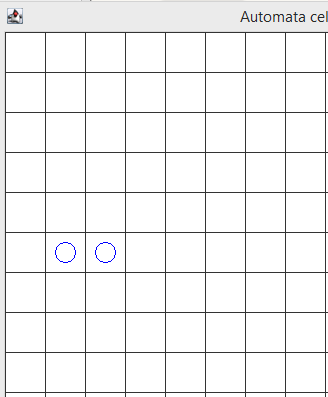
Uno:



Dos:



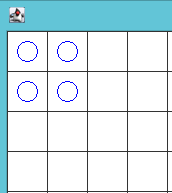
Tres:



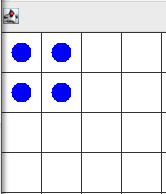
1. Adicionen en la esquina inferior izquierda un Bloque y ejecuten la aplicación, ¿qué pasa? ¿queda estático? Capture una pantalla. No olviden escribir la prueba correspondiente.

Quedara estático ya que todas las células tendrán exactamente tres vecinas y nunca morirán.

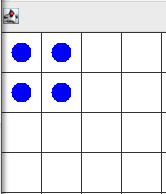
Estado inicial



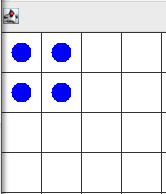
Click1



Click2



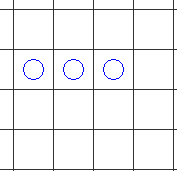
Click3



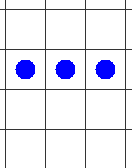
1. Adicionen en la parte central inferior un Parpadeado (con espacio para parpadear) y ejecuten la aplicación, ¿qué pasa? ¿parpadea? Capture dos pantallas de parpadeo. No olviden escribir la prueba correspondiente.

A lo largo de los tic-tac, las células entraran en un estado de loop, es decir volverán al estado que dejaron.

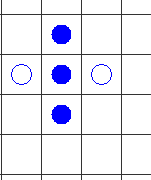
Estado inicial



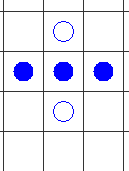
Click1



Click2



Click3



**Empaquetando la versión final para el usuario.** [En lab03.doc, automata.asta , \*.java, automata.jar]

1. Revise las opciones de BlueJ para empaquetar su programa entregable en un archivo .jar. Genere el archivo correspondiente.
2. Consulte el comando java para ejecutar un archivo jar. ejecutennlo ¿qué pasa?
3. ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.

# 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** [En lab03.doc]

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

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

Los comandos son los siguientes:

Crear: comando MKDIR que sirve para añadir un directorio en la dirección donde se está parado.

Borrar: comando RMDIR que sirve para borrar el directorio que se especifique en la dirección en la cual este parado.

Listar contenido: comando DIR sirve para listar el contenido de una carpeta donde estemos parados.

Copiar: comando COPY sirve para copiar archivos desde la carpeta donde nos encontramos parados a otra dirección diferente, se puede copiar un archivo en específico o varios archivos de una misma clase.

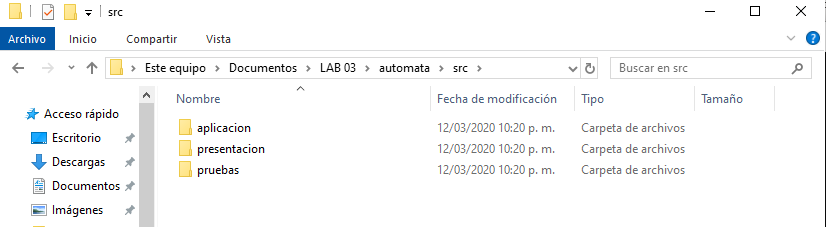
Eliminar: comando DEL sirve para eliminar uno o más archivos.

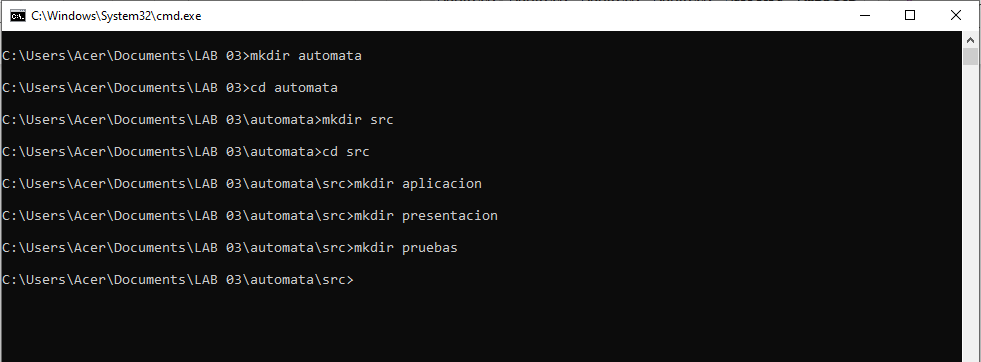
1. Organicen un nuevo directorio con la estructura propuesta para probar desde allí su habilidad con los comandos de consola. Consulten y capturen el contenido de su directorio

automata

src

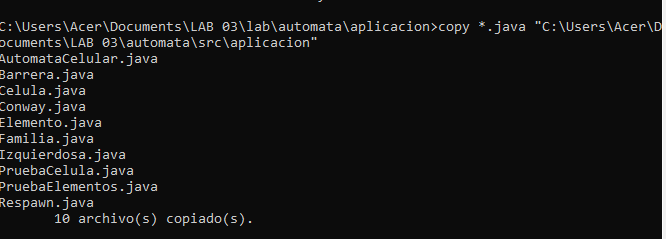
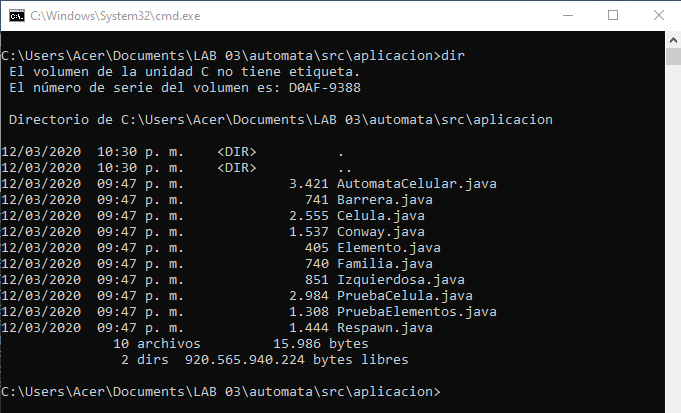
aplicacion presentacion pruebas





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

Consulte y capture el contenido de src/aplicación



**Estructura de proyectos java** [En lab03.doc]

En java los proyectos se estructuran considerando tres directorios básicos.

automata

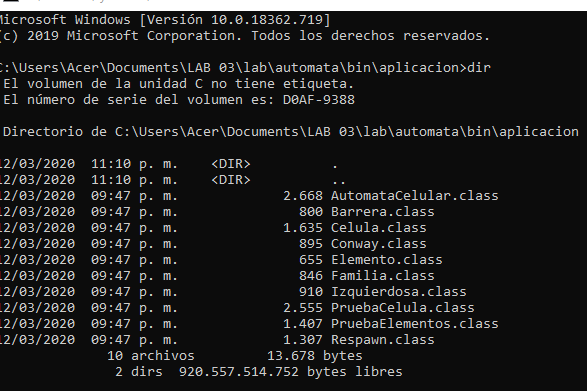
src bin docs

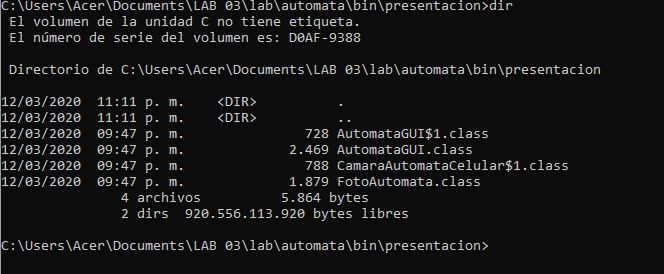
1. Investiguen los archivos que deben quedar en cada una de esas carpetas y la organización interna de cada una de ellas.

Los archivos que deben quedar en el SRC son los archivos .java , en BIN van los archivos .class y en DOC van los archivos de documentación y su organización interna va por paquetes, dentro de cada uno tendrá una carpeta llamada presentación y otra de aplicación donde irán los archivos que le corresponden a cada uno.

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

Debería copiar los archivos .class que aparecen tanto en la carpeta aplicación como en la de presentación, cada uno en su respectivo paquete.





**Comandos de java** [En lab03.doc]

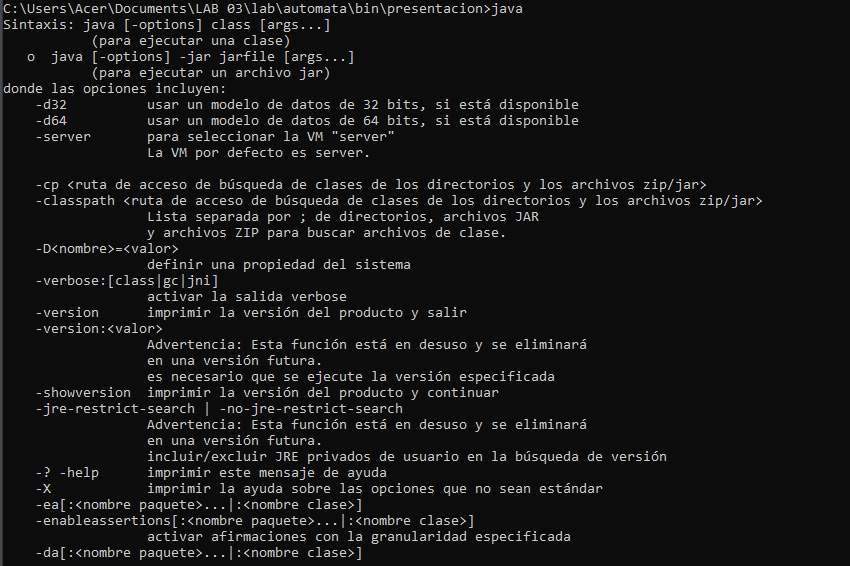
1. Consulte para qué sirven cada uno de los siguientes comandos:

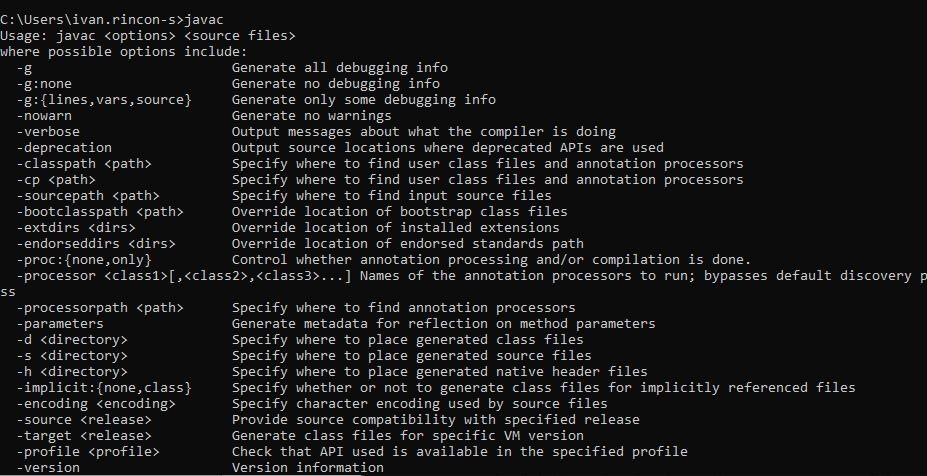
Javac: Sirve para compilar uno o varios archivos .java de la dirección donde estoy parado.

Java : Sirve para ejecutas uno o varios archivos .java de la dirección donde estoy parado.

Javadoc: Sirve para generar la documentación de los archivos java que tengamos en la dirección donde estamos parados.

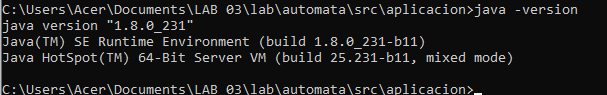
Jar: Sirve para crear el archivo ejecutable estableciendo siempre cual es el archivo desde el cual se ejecutará el main.

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

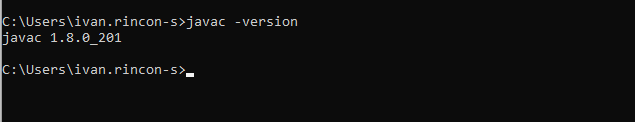


1. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.

Como resultado me arroja que tengo la versión de Java 1.8.0\_231-b11

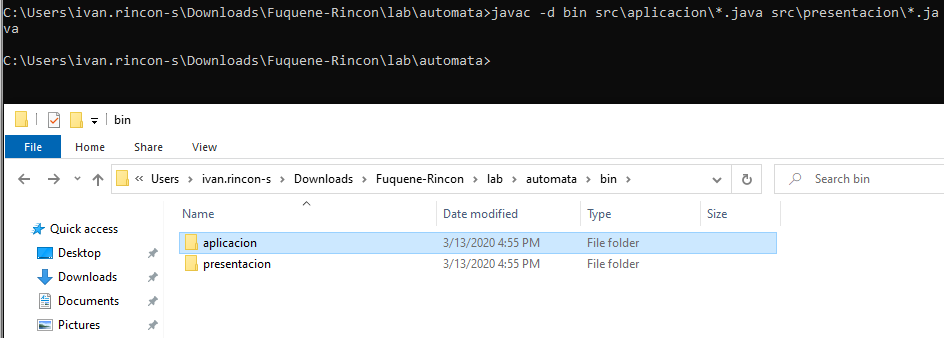


Como resultado me arroja que tengo la versión de Java 1.8.0\_231-b11

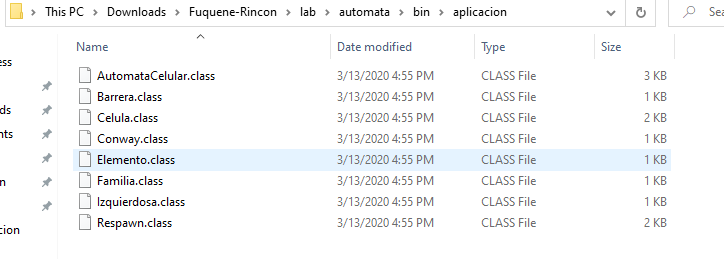


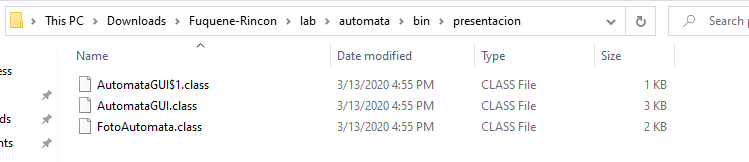
**Compilando** [En lab03.doc]

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



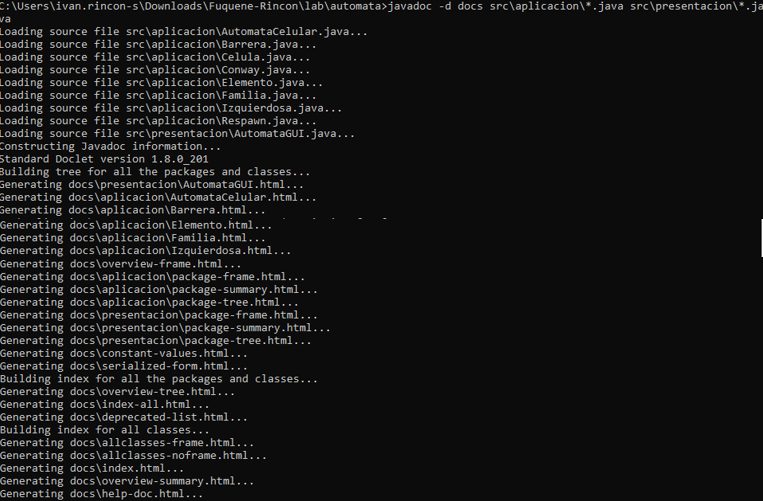
1. Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles nuevos archivos aparecen ahora y dónde se ubican?





**Documentando** [En lab03.doc]

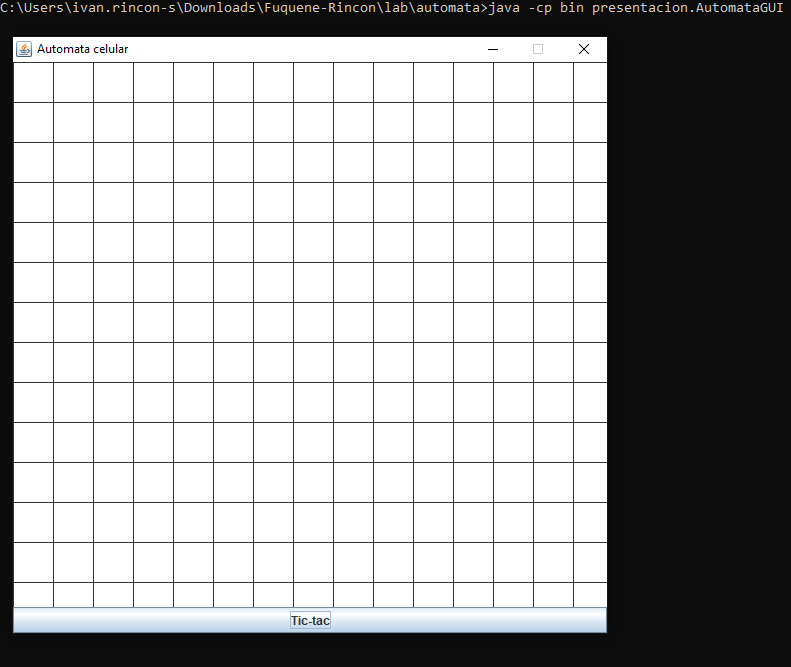
1. Utilizando el comando javadoc, desde el directorio raiz, genere la documentación (API) en formato html, en este directorio. ¿cuál es el comando completo para generar esta documentación?



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

**Ejecutando** [En lab03.doc]

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



**Probando** [En lab03.doc]

1. Adicione ahora los archivos del directorio pruebas y trate de compilar nuevamente el programa.Tenga en cuenta que estas clases requieren la librería junit 4.8. ¿Cómo se incluye un paquete para compilar? ¿Qué instrucción completa tuvo que dar a la consola para compilar?
2. 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>

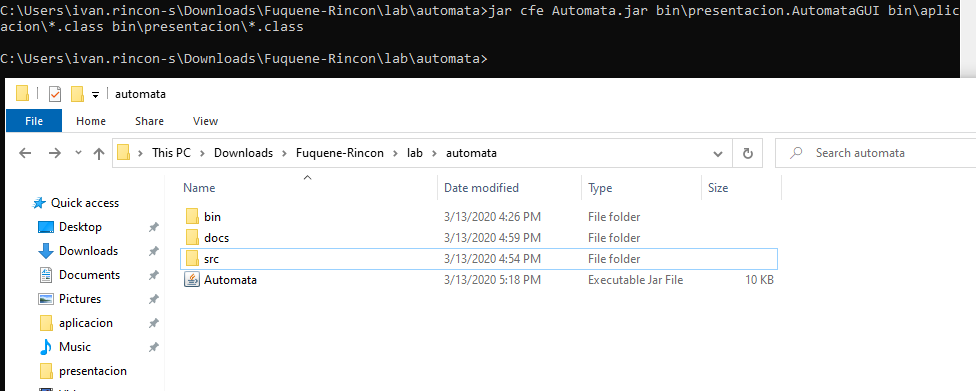


1. Pegue en su documento el resultado de las pruebas

**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 ?



1. ¿Cómo se ejecuta el proyecto empaquetado?



# RETROSPECTIVA

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

# (Horas/Hombre)

# Iván Camilo Rincon 15 horas

# Miguel Ángel Fúquene Arias 15 hora

# 

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

# Casi terminado, puesto que le invertimos el tiempo y la disposición durante toda la semana.

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

# La programación a pares, ya que con estos ambos integrantes del grupo aprenden los temas de forma igualitaria.

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

# Pudimos aplicar los conceptos vistos en clase satisfactoriamente puesto que pudimos realizar todas las funcionalidades que se requerían.

# 

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

# resolverlo?

# Problemas en la compilación por consola debido que tenemos no tenemos las variables de entorno de java no están configuradas en nuestros computadores.

# 

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

# resultados?

# Distribuimos bien el tiempo y utilizamos espacios de estudio fuera de clase para fortalecer los conocimientos y poder realizar todas las funcionalidades que proponía el laboratorio. Nos comprometemos a reforzar los temas vistos.