**CONOCIENDO**

1. En el directorio descarguen los archivos contenidos en automata.zip. Revisen el código de la aplicación
   1. ¿Cuántos paquetes tiene?

2

* 1. ¿Cuántas clases tiene en total? ¿Cuántas tienen fuentes?

4. Todas tienen fuente.

* 1. ¿Cuál es la clase ejecutiva? ¿Por qué?

AutomataCelular

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

Crear un autómata celular, mostrar su longitud, mostrar un elemento especificado, agregar un elemento al autómata, algunos elementos y tictac.

* 1. ¿Qué hace actualmente? ¿Por qué?

Actualmente solo se crea el autómata, podemos poner y consultar los elementos que queramos, pero no se hace nada más porque hacen falta métodos por implementar.

**ARQUITECTURA GENERAL**

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

Un paquete es un contenedor de un grupo de clases, relacionadas entre sí.

1. ¿Para qué sirve?

Agrupar clases que tienen algo en común, reutilizar código y seguridad gracias a los modificadores de acceso.

1. Explique su uso en este programa.

Un ejemplo es el paquete de aplicación donde cada una de sus clases debe tener “package aplicacion” en la parte superior del código.

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

Que los paquetes se generan a partir de directorios y hay algunos directorios que necesitan de otros paquetes

**ARQUITECTURA DETALLADA**

Adicione en las fuentes la clase de pruebas necesaria para BDD. (No lo adicione al diagrama de clases)

1. ¿En qué paquete debe estar? ¿Por qué?
2. ¿Asociado a qué clase? ¿Por qué?

**Ciclo 1**

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

Esta clase utiliza una matriz de elementos, puede albergar células porque el autómata alberga elementos y todas las células son elementos.

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

Célula es definida por Elemento, ya que Elemento es una interfaz y Célula la implementa.

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

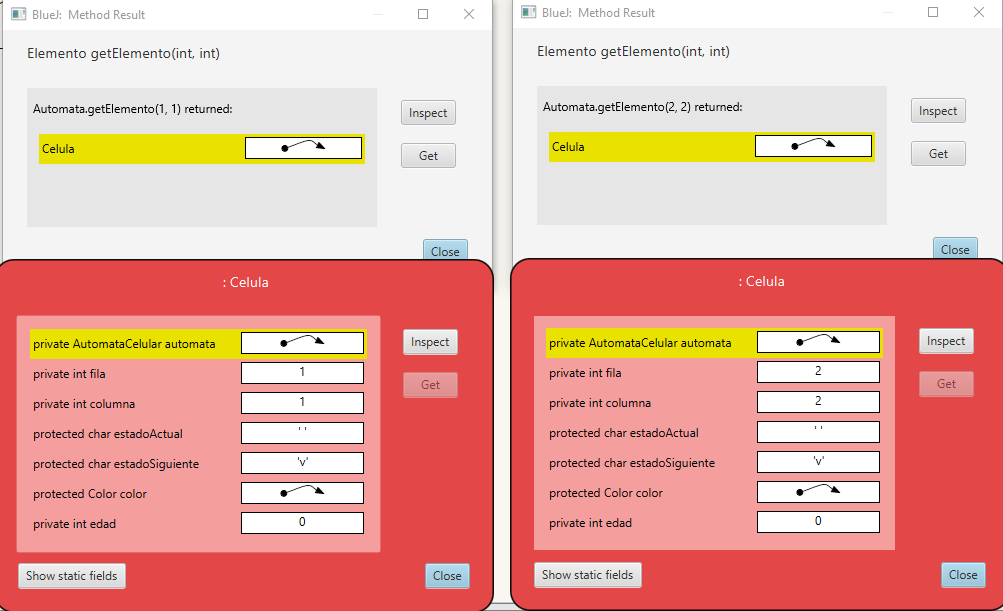
Las células saben hacer getFila, getcolumna, getcolor, isVivo, decida, cambie. Pueden tener un color puede estar viva o muerta.

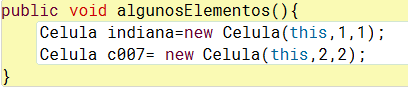
1. Por comportarse como un elemento, ¿qué sabe hacer? ¿qué no puede hacer distinto? ¿qué debe aprender a hacer? Justifique su respuesta.

Un Elemento sabe decidir, cambiar, dar su forma y color y decir si está vivo o no.

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

Una celula es de color negro. Para decidir, si una celula tiene más de dos años en el siguiente instante estará muerta. Cada vez que cambia aumenta un año y su estado actual será igual al siguiente.

1. 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.

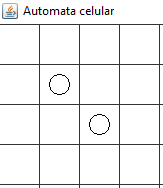
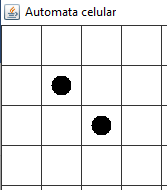
Después del primer tictac: Ambas células quedan vivas y su edad pasa a ser 1.

Después del segundo tictac: Las dos células siguen vivas y su edad pasa a ser 2.

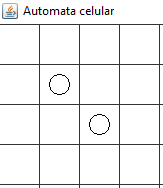
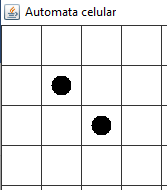
Después del tercer tictac: Las dos células mueren.

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

Así están las células antes del primer tictac: Primer tictac:



Segundo tictac: Tercer tictac:



**Ciclo 2**

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.

En el primer segundo ambas células tendrían el estado vacío.

En el segundo siguiente, la célula de la izquierda quedará viva y la de la derecha morirá.

En el tercer segundo el estado será el mismo.

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?

Es necesario agregar una nueva clase, Izquierdosa, que sea una herencia de Celula. Se tiene que sobre-escribir el método decida, ya que la célula izquierdosa decide de manera diferente a una célula normal.

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

No debería cambiar el código en AutomataCelular porque AutomataCelular acepta elementos y una celula izquierdosa lo es.

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. Se tiene la misma situación que se tiene en el numeral 1.

1. Construyan el método. ¿Es correcto?

El método es correcto, las barreras se crean satisfactoriamente.

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

**Ciclo 3**

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

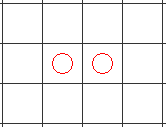
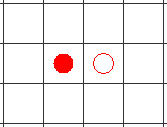
Creamos la clase Barrera que va a implementar la interfaz Elemento, sobre-escribimos algunos métodos y agregamos las barreras al autómata.

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

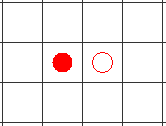
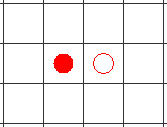
No tuvimos que cambiar nada en la clase AutomataCelular ya que solo tuvimos que crear una nueva clase que implemente a Elemento.

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.
2. Construyan el método. ¿Es correcto?
3. Ejecuten el programa y hagan tres clics en el botón. Capturen una pantalla significativa. ¿Qué pasa? ¿es correcto?

Antes del primer click: Primer click:



Segundo click: Tercer click:



**Ciclo 4**

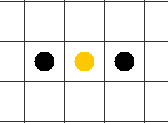
1. Propongan, describan e Implementen un nuevo tipo de células.

Nuestra célula propuesta se llama célula sándwich. Es un tipo de célula que es de color anaranjado y se muere cuando tiene un elemento vivo seguido a la izquierda y otro elemento vivo seguido a la derecha.

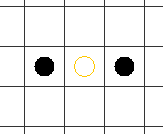
1. Incluyan una pareja de ellos con el nombre de ustedes. ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes.

En el primer caso incluimos una célula sándwich que está encerrada a los lados por dos células vivas, este caso es para ver si se cumple la característica principal de esta célula:

En el primer tictac nuestra célula sale viva ya que decide primero que la célula a su derecha entonces en el momento en que decide no sabe si la célula siguiente está viva:

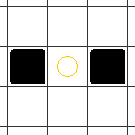


En el segundo tictac nuestra célula muere ya que al momento de decidir ve que las células a sus costados están vivas:

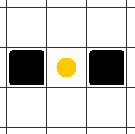


En el segundo caso añadimos una célula sándwich en medio de dos barreras, la intención de este caso es ver que, aunque esté en medio de dos elementos, si estos no están vivos nuestra célula seguirá funcionando y seguirá viva:

Su estado inicial será:



Y luego después de n Tic-Tac la célula seguirá viva



**Ciclo 5**

1. Propongan, describan e Implementen un nuevo tipo de elemento.

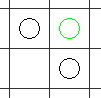
El elemento que propusimos se llama Virus, es de color verde y su forma de decidir es la siguiente:

Un Virus decide morir si no hay al menos un elemento vivo en su misma fila o columna dentro del autómata.

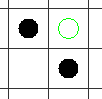
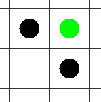
1. Incluyan un par de ellos con los nombres semánticos. ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes.

En el primer caso, incluimos un virus en la misma fila y columna que dos células normales. Hicimos este caso con el fin de mostrar que la manera de decidir del virus funciona correctamente, ya que el virus estará vivo mientras las células estén vivas.

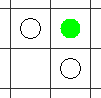
Estado inicial:



Primer y segundo tictac:

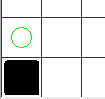
 

Tercer tictac:



En el segundo caso, añadimos un virus en la misma columna que una barrera, la finalidad de este caso es mostrar que, aunque haya un elemento, si no esta vivo el virus estará muerto.

El estado del virus siempre será este:



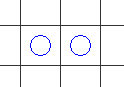
Podemos ver que el virus nunca estará vivo ya que el único elemento con el que comparte una fila o columna es una barrera, como la barrera no está viva el virus tampoco lo estará.

**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.
2. 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?

Adicionamos una nueva clase llamada Conway que extiende de Célula, además modificamos el tictac.

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?



Ambas células quedan muertas, incluso después del primer click.

Es correcto ya que desde el primer instante ambas mueren de soledad.

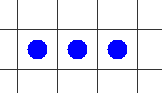
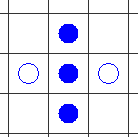
1. Adicionen en la esquina inferior izquierda un Bloque y ejecuten la aplicación, ¿qué pasa? ¿queda estático? Capture una pantalla. No olviden escribir la prueba correspondiente.

Al adicionar el bloque este queda estático siempre.



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

Al adicionar este parpadeador este pasa de estar horizontal a estar vertical, es decir, que si parpadea.

**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.

**DE BLUEJ A CONSOLA**

**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.

crear directorio: mkdir

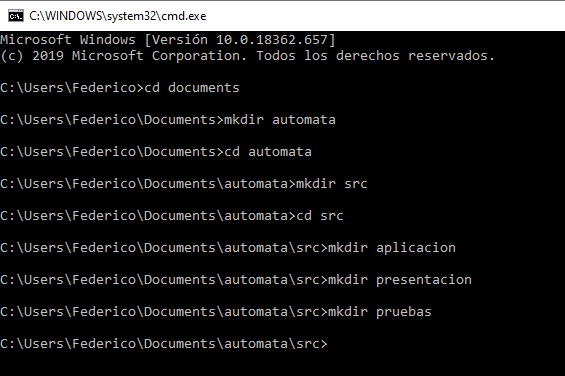
eliminar directorio: rd /S

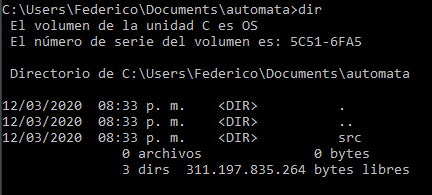
moverse a un directorio: cd

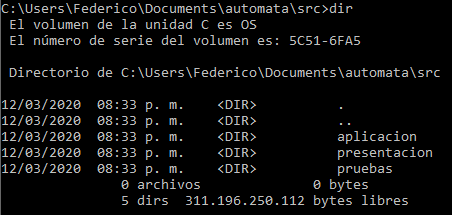
ver contenido del directorio: dir

copiar archivos: copy

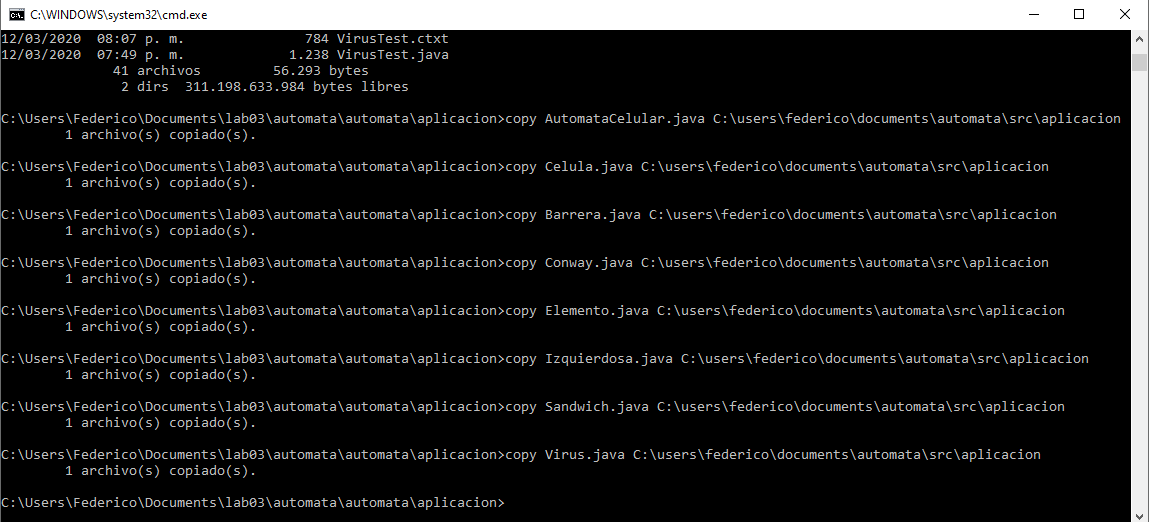
eliminar archivos: del

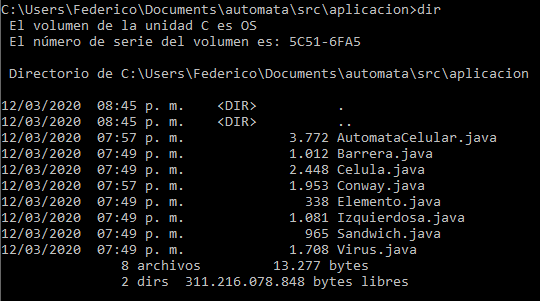
1. Organicen un nuevo directorio con la estructura propuesta para probar desde allí su habilidad con los comandos de consola. Consulten y capturen el contenido de su directorio



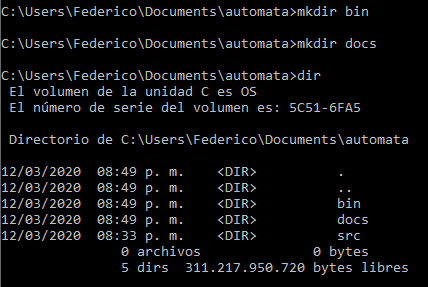


1. En el directorio copien únicamente los archivos \*.java del paquete de aplicación. Consulte y capture el contenido de src/aplicación





**Estructura de proyectos java**



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

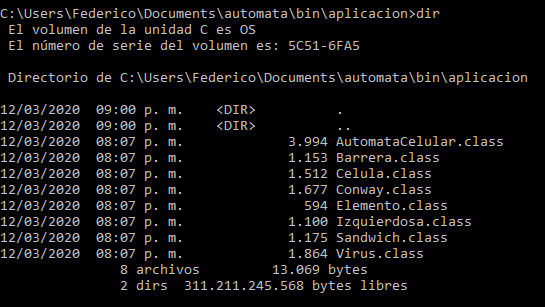
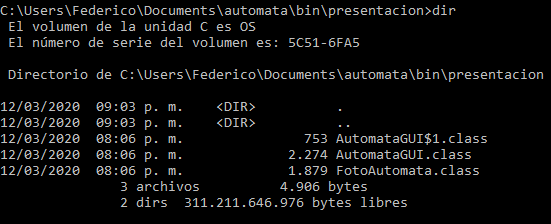
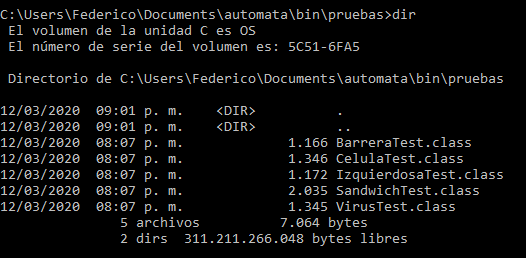
En el directorio src deben ir los archivos .java es decir el código fuente.

En el directorio bin deben ir los archivos .class que es el código máquina para la jvm.

En el directorio docs deben ir los archivos .html es decir la documentación de nuestro código.

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

En el directorio bin vamos a copiar los archivos .class de la aplicación, incluyendo las pruebas, y de la GUI. Ya que en este directorio se ponen los archivos que va a leer la máquina virtual de java.

**Comandos de java**

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

javac: sirve para compilar nuestro programa.

java: sirve para ejecutar nuestro programa.

javadoc: sirve para generar la documentación del programa.

jar: sirve para ejecutar un archivo jar.

RETROSPECTIVA

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

Federico Barrios: 12 horas

Cristian Ruiz: 12 horas

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

Incompleto, porque tuvimos problemas con los comandos en consola desde nuestros computadores personales, creemos que fue porque no teníamos la versión requerida de java.

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

Porque cuando se trabaja en grupo los aportes de los integrantes son significativos, pero están sujetos a que los demás integrantes los modifiquen.

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

Nuestro mayor logro fue poder conocer más librerías de java que nos facilitan herramientas para poder diseñar aplicaciones con una interfaz gráfica.

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

Trabajamos juntos la totalidad del tiempo, organizar mejor el tiempo para realizar la entrega completa.