PROGRAMACIÓN ORIENTADA A OBJETOS
Herencia e interfaces
ADEMÁS Java desde consola
2021-1
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.
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 es un modelo matemático para representar sistemas que puedan ser descritos como una colección masiva de elementos simples que interactúan localmente unos con otros y que evolucionan a pasos discretos.

- Los elementos se ubican en una rejilla, máximo uno en cada celda.
- En cada momento, todos los elementos toman la decisión de su acción futura y luego todas cambian.
- Los elementos deciden qué va suceder en la siguiente etapa de tiempo de acuerdo a su naturaleza, su estado y sus vecinos (norte, sur, este u oeste).

Algunos de los elementos son células. Las características específicas de las células con:

1. Una célula puede estar en uno de un conjunto posible de estados. En nuestro caso: viva (●) o muerta(o).
2. 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?

2 paquetes:dominio y presentación.

b) ¿Cuántos componentes en total?

2 paquetes
3 clases
1 clase interface

1 clase abstract

c) ¿Qué tipos de componentes son?

Dos paquetes Presentación y domain, donde presentacion tiene automatasGUI, y domain tiene Elemento como interface, y ser como clase abstracta y célula utiliza la interfaz Elemento y existe una clase Automata Celular

d) ¿Cuál es el propósito del paquete presentación?

ofrecer las clases que se encarga de implementar las vistas de nuestro programa

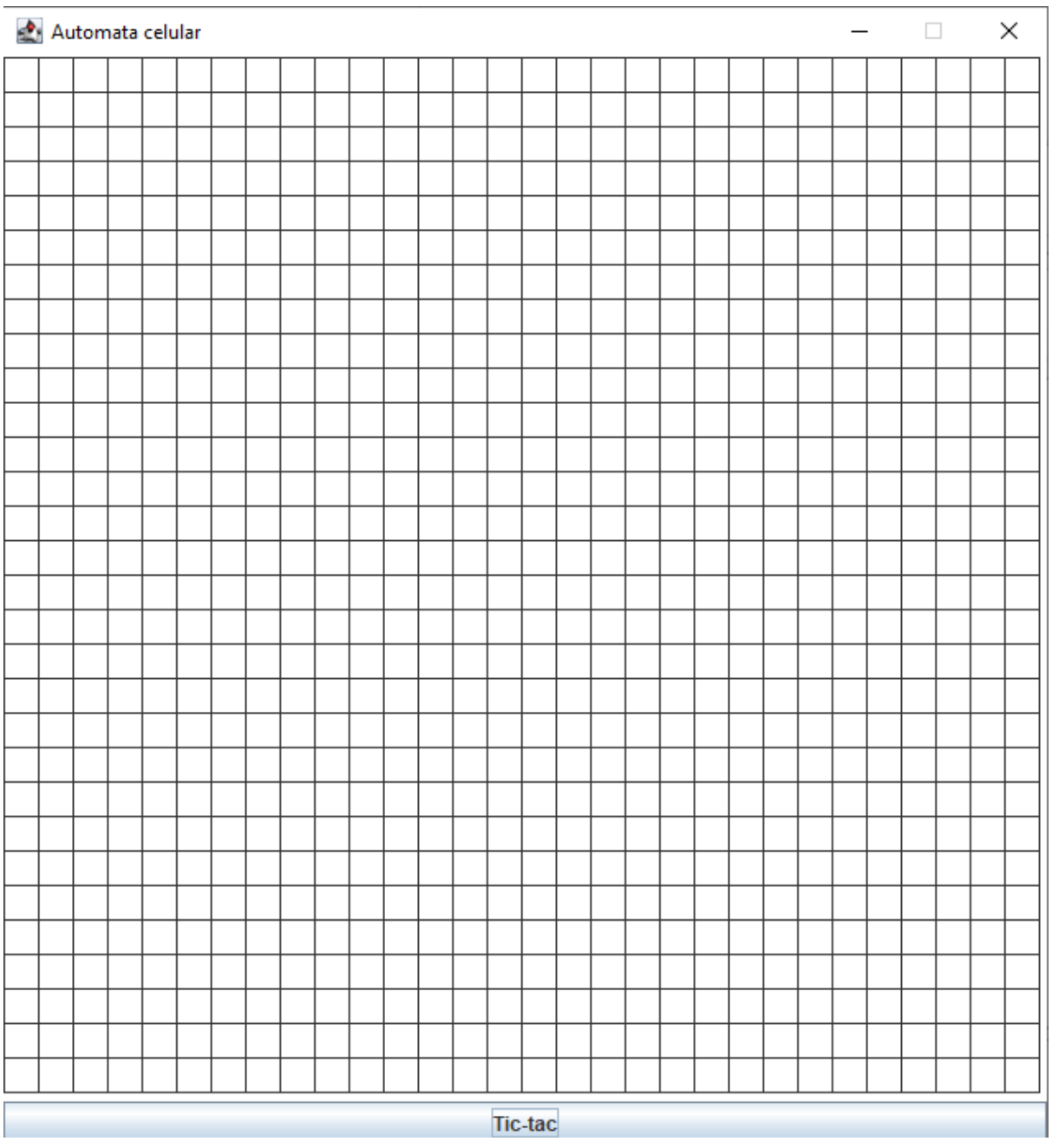
e) ¿Cuál es el propósito del paquete dominio?

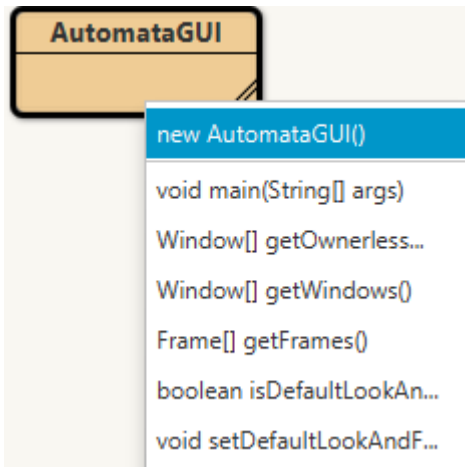
Ofrecer las clases encargadas de la correcta funcionalidad de nuestro programa

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

automataGUI el método de void main(args[])

3. Ejecuten el programa.
¿Qué funcionalidades ofrece?





¿Qué hace actualmente?

nada

¿Por qué?

Hace falta implementar el código de la solucion

(Deben ejecutar la aplicación java, no crear un objeto como lo veníamos haciendo

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: sirve para almacenar las clases que tienen una funcionalidad en común

IMPORT: Invoca los paquetes de java/paquetes de clase, los cuales podremos manipular dentro de la clase la cual uso el Import.

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

```
C:\Users\user\Desktop\Universidad\POOB 2021 ahora si\2 Tercio\Lab03\automata>dir
 El volumen de la unidad C no tiene etiqueta.
 El número de serie del volumen es: BA67-DBDF

 Directorio de C:\Users\user\Desktop\Universidad\POOB 2021 ahora si\2 Tercio\Lab03\automata

12/03/2021  07:17 a. m.    <DIR>          .
12/03/2021  07:17 a. m.    <DIR>          ..
10/04/2019  10:00 a. m.    <DIR>          doc
12/03/2021  08:31 a. m.    <DIR>          domain
11/03/2021  09:54 p. m.               775 package.bluej
12/03/2021  08:36 a. m.    <DIR>          presentacion
12/03/2021  07:17 a. m.                 0 README.TXT
               2 archivos            775 bytes
               5 dirs  109.739.184.128 bytes libres

C:\Users\user\Desktop\Universidad\POOB 2021 ahora si\2 Tercio\Lab03\automata>
```

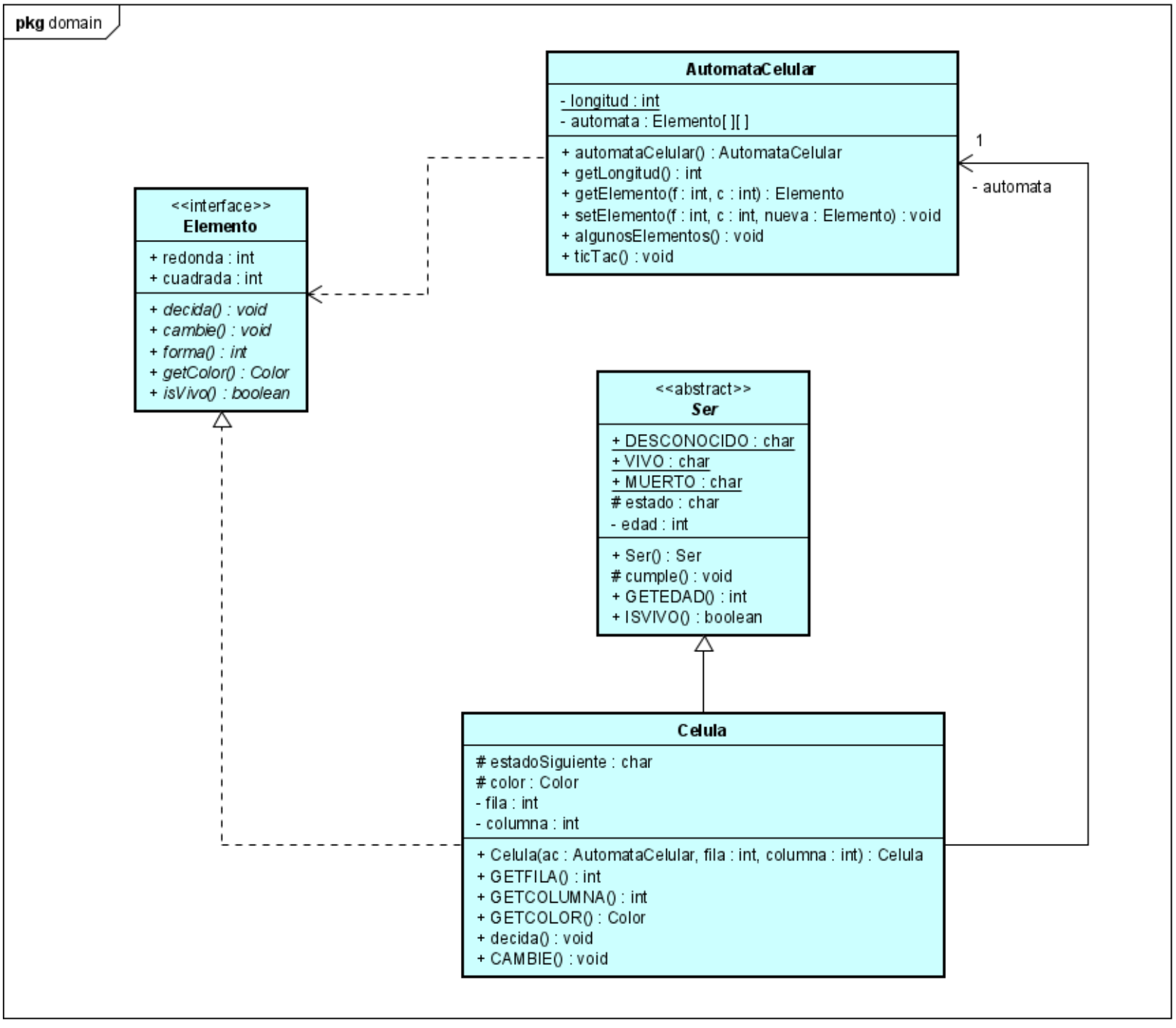
cada paquete tiene su respectiva carpeta.

La coincidencia es que en cada directorio se adjuntan clases que trabajan en tareas comunes.

3. Inicie el diseño con un diagrama de paquetes en el que se presente los componentes 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. [En lab03.doc y automataasta]

1. Usando ingeniería reversa prepararen el proyecto para MDD. Presente el diseño estructural actual del paquete de dominio (diagrama de clases). No vamos a hacer el diseño del paquete de presentación.



2. Adicione en las fuentes la clase de pruebas unitarias necesaria para BDD. (No lo adicione al diagrama de clases)
¿En qué paquete debe estar?
En el paquete domain
¿Por qué?
Porque allí se encuentra las clases que se encargan del funcionamiento del programa
¿Asociado a qué clase?
la clase Celula.
¿Por qué?
la clase Célula la cual es la base para el funcionamiento, en esta clase se encuentra toda la implementación de los métodos.

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?
utiliza una matriz
¿Puede recibir células?

¿Por qué?

2. Estudie el código asociado a la clase Celula, ¿en qué estado se crea? ¿qué forma usa para pintarse? ¿cuándo aumenta la edad? Justifique la respuesta.

3. En general, ¿qué clases definen la clase Celula?

4. Todas las células (incluyendo los nuevas subclases) ¿qué saben hacer (métodos definidos)? ¿qué no puede hacer distinto? ¿qué debe aprender a hacer (métodos a definir)? Justifique su respuesta.

5. Por ser un ser, ¿qué datos tiene? ¿qué ya sabria hacer? ¿qué decide hacer distinto? ¿qué no puede hacer distinto a todos los seres? ¿qué debe aprendar a hacer? Justifique su respuesta.

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

7. 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 todos las células? Capturen una pantalla significativa.

8. 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.

9. Construyan el método. ¿Es correcto?

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

Ciclo 2. Incluyendo a las células especiales [En lab03.doc y automataasta]
(NO OLVIDE BDD – MDD)

El objetivo de este punto es permitir recibir en el automata celulas especiales. Las especiales son de color verde y si se sienten solas (no hay un elemento vivo alrededor) generan una nueva celula normal para que las acompañe, si pueden; sino (estan rodeadas de células muertas) , mueren.

1. Implemente esta nueva célula. ¿cuáles métodos se sobre-escriben (overriding)?
2. Adicione una pareja de especiales, llámelos agamenon y venus, ¿Cómo quedarían después de uno, dos y tres Tic-tac? Escriba la prueba correspondiente.

3. Diseñen y construyan el método. ¿Es correcto?

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

Ciclo 3. Adicionando una mini-calefactores [En lab03.doc, automata.asta y *.java]

El objetivo de este punto es incluir en el AutomataCelular calefactores (sólo vamos a permitir un tipo calefactores). Los calefactores son cuadrados, no deciden y siempre cambian de color de rojo a amarillo y de amarillo a rojo. (NO OLVIDE BDD – MDD)

1. Construyan la clase Calefactor para poder adicionaría en el AutomataCelular ¿qué hicieron? ¿debe cambiar en el código del AutomataCelular en algo? ¿por qué?

2. Para aceptar este elemento ,

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

4. Construyan el método. ¿Es correcto?

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

Ciclo 4. Nueva Celula: Proponiendo y diseñando
El objetivo de este punto es permitir recibir en un nuevo tipo de célula (NO OLVIDE BDD – MDD)

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

6. 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.

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 el nombres semánticos. ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes.

C**aso 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 menos de dos o más de tres vecinas vivas 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.

3. 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.

4. 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?

5. 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?

6. 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.

7. 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.

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.

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 automata
src
aplicacion
presentacion
pruebas

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

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.

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

Comandos de java [En lab03.doc]

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

javac = los javac comando en Java compila un programa desde un símbolo del sistema. Lee un programa de código Java a partir de un archivo de texto y crea un archivo de clase Java compilado. La forma básica de la javac comando es javac nombre [opciones]

java=El comando se utiliza para ejecutar una aplicación Java desde la línea de comandos.

javadoc= Este comando sirve para generar páginas HTML de documentación de API a partir de archivos de origen Java.

jar= Son archivos que se utilizan para archivar o compilar un programa.
Una de las características de estos archivos es la compresión de datos sin sufrir pérdidas de información.

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 qué corresponden estos dos comandos. Documente el resultado.

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.

2. 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?

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

Ejecutando [En lab03.doc]

6. Empleando el comando java, desde el directorio raíz, 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

3. Pegue en su documento el resultado de las pruebas

Empaquetando [En lab03.doc]

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

2. ¿Cómo se ejecuta el proyecto empaquetado?

COSA PARA EMPAQUETAR
JAR

RETROSPECTIVA

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

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

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

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

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