

PROGRAMACIÓN ORIENTADA A OBJETOS

Herencia e interfaces

ADEMÁS Java desde consola

2022-1

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 : Use [collective ownership](#). Only one pair [integrates code at a time](#).
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 celular (A.C.)¹ es un modelo matemático para representar sistemas que puedan ser descritos como una colección masiva de items simples que interactúan localmente unos con otros y que evolucionan a pasos discretos.

- Los items se ubican en una rejilla, máximo uno en cada celda.
- En cada momento, todos los items toman la decisión de su acción futura y luego todos cambian.
- Los items 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 items 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)

A. 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) ¿Cuál es el propósito del paquete presentación? c) ¿Cuál es el propósito del paquete dominio?
2. Revisen el paquete de dominio, a) ¿Cuáles son los diferentes tipos de componentes de este paquete? b) ¿Qué implica cada uno de estos tipos de componentes?
3. Revisen el paquete de presentación, a) ¿Cuántos componentes tiene? b) ¿Cuántos métodos ofrece?
4. Para ejecutar un programa en java, a) ¿Qué método se debe ejecutar? b) ¿En qué paquete de automata se encuentra? ¿En qué clase? c) ¿Por qué?
5. Ejecuten el programa. a) ¿Qué funcionalidades ofrece? c) ¿Qué hace actualmente? c) ¿Por qué?

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

1 http://es.wikipedia.org/wiki/automataCellr_célular

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

1. Consulte el significado de las palabras **package** e **import** de java. a) ¿Para qué sirve **package**? b) ¿Para qué sirve **import**? c) Explique su uso en esta aplicación.
2. Revise el contenido del directorio de trabajo y sus subdirectorios. a) Describa su contenido. b) ¿Qué coincidencia hay entre paquetes y directorios?
3. Adicione al diseño la arquitectura general con un diagrama de paquetes en el que se presente los paquetes y las relaciones entre ellos. Consulte la referencia en moodle.
En astah, crear un diagrama de clases (cambiar el nombre por Package Diagram0)

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

1. Para preparar el proyecto para **BDD**. Completen el diseño detallado del paquete de dominio. Adicionen el diagrama de clases en el paquete correspondiente. a) ¿Qué componentes hacían falta?
2. Complementen 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?
3. 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?

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

(NO OLVIDE BDD - MDD)

1. Estudie la clase **CellularAutomata** ¿Qué tipo de colección usa para albergar los items? ¿Puede recibir células? ¿Por qué?
2. Estudie el código asociado a la clase **Cell**, ¿en qué estado se crea? ¿qué forma usa para pintarse? ¿cuándo aumenta la edad? ¿qué clases definen la clase **Cell**? Justifique sus respuestas.
3. **Cell** por ser un **Agent**, ¿atributos tiene? ¿qué puede hacer (métodos)? ¿qué decide hacer distinto? ¿qué no puede hacer distinto a todos los agentes? ¿qué debe aprender a hacer? Justifique sus respuestas.
4. Por comportarse como un **Item**, ¿qué sabe hacer? ¿qué decide hacer distinto? ¿qué no puede hacer distinto? ¿qué debe aprender a hacer? Justifique sus respuestas.
5. Ahora vamos a crear dos células en diferentes posiciones (1,1) (2,2) llámelos **indiana** y **007** usando el método **someltems()**. Ejecuten el programa, ¿Qué pasa con las células? ¿Por qué? Capturen una pantalla significativa.
6. Diseñen, construyan y prueben el método llamado **ticTac()** de la clase **CellularAutomata**.
7. ¿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?

E. Ciclo 2. Incluyendo a las células inquietas [En lab03.doc y automataasta]

(NO OLVIDE BDD - MDD)

El objetivo de este punto es permitir recibir en el automata celulas inquietas. Las inquietas (i) son de color naranja; (ii) si algunas 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 sobre-escriben (overriding)?
2. Diseñen, construyan y prueben esta nueva clase.
3. 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?

F. Ciclo 3. Adicionando bombillos [En lab03.doc, automata.asta y *.java]

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).

(NO OLVIDE BDD - MDD)

1. Construyan la clase **Bulb** para poder adicionarla en el **CellularAutomata**? ¿debe cambiar en el código del **CellularAutomata** en algo? ¿por qué?
2. Diseñen , construyan el método y prueben esta nueva clase.
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?

G. 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 el nuevo tipo de células.
2. 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.

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**.

(NO OLVIDE BDD - MDD)

1. Propongan, describan e Implementen un nuevo tipo de item.
2. 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.

J. Caso 6. El Juego de la vida

El juego de la vida es el mejor ejemplo de un autómata celular, diseñado por el matemático británico John Horton Conway en 1970. Un CellularAutomata celular 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 CellularAutomata 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 vacía rodeada por 3 células vivas "nace" una nueva célula (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 CellularAutomata.



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?
3. 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?
4. 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.
5. 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
java
javadoc
jar
```

2. Cree una sesión de consola y consulte en línea las opciones de los comandos java y javac. Capture las pantallas.
3. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.

Compilando [En lab03.doc]

1. Utilizando el comando javac, **desde el directorio raiz (desde automata con una sola 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 que abrir para empezar a navegar por la documentación? Ábralo y capture la pantalla.

Ejecutando [En lab03.doc]

4. 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 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` ?
2. ¿Cómo se ejecuta el proyecto empaquetado?

RETROSPECTIVA

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