PROGRAMACIÓN ORIENTADA A OBJETOS
Herencia e interfaces
2022-2
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.

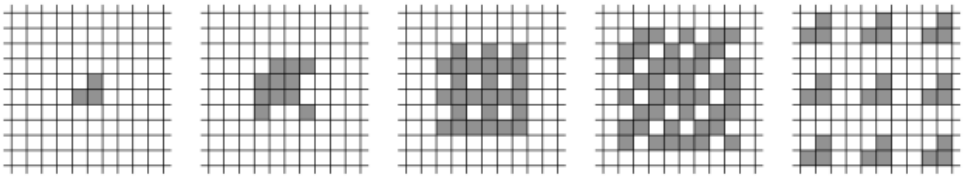
ENTREGA

- ➔ 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 Inspirado en 2017 Problem G Replicate Replicate Rfplicbte

El propietario de la corporación **Automatic Cellular** acaba de patentar un nuevo proceso para la producción en masa de piezas idénticas. Su enfoque utiliza una red bidimensional de celdas que contienen elementos que pueden estar en dos estados activos e inactivos. En una secuencia de pasos discretos, cada elemento en la retícula actualiza **simultáneamente** su estado examinando su propio estado y, si es necesarios, aquellos de sus ocho vecinos circundantes. Los detalles exactos son, por supuesto, propietarios.
De todos los elementos, las células son las más importates. Para iniciar la producción un conjunto de celdas en la red se llena con células en la parte que se va a producir.
Las células **Rfplicbte,** un caso especial del células que están experimentando, si encuentran un número impar de las nueve vecinas llena con células activas, quedan activas, de lo contrario quedan inactivas. La figura muestra varios pasos en el proceso de replicación para un patrón simple que consta de tres celdas activas **Rfplicbte** activas ubicadas en un cultivo de células.



Conociendo [En lab03.doc y replicate.asta]

1. En el directorio descarguen los archivos contenidos en replicate.zip. Revisen el código de la aplicación a) ¿Cuántos paquetes tiene? d) ¿Cuál es el propósito del paquete presentación? e) ¿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 dominio, a) ¿Cuántos componentes tiene? b) ¿Cuántos métodos ofrece?
4. Para ejecutar un programa en java, ¿Qué método se debe ejecutar? ¿En qué clase de replicate se encuentra?
5. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?
   (Deben ejecutar la aplicación java, no crear un objeto como lo veníamos haciendo)

Arquitectura general. [En lab03.doc y replicate.asta]

1. Consulten el significado de las palabras package e import de java. ¿Qué es un paquete? ¿Para qué sirve? Explique su uso en este programa.
2. Revisen el contenido del directorio de trabajo y sus subdirectorios. Describa su contenido. ¿Qué coincidencia hay entre paquetes y directorios?
3. Adicionen 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)

Arquitectura detallada. [En lab03.doc y replicate.asta]

1. Para preparar el poyecto para **BDD.** Completen el diseño detallado del paquete de dominio. a) ¿Qué componentes hacían falta?
2. Revisen 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. Adicionen en las fuentes la clase de pruebas unitarias necesaria para **BDD.** (No lo adicione al diagrama de clases) ¿a) En qué paquete debe estar? b) ¿Asociado a qué clase? c) ¿Por qué?

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

(NO OLVIDE BDD – MDD)

1. Estudie la clase AManufacturing ¿Qué tipo de colección usa para albergar las Thing? ¿Puede recibir células? ¿Por qué?
2. Estudie el código asociado a la clase Cell, ¿a) en qué estado se crea? b) ¿qué forma usa para pintarse? c) ¿cuándo cambia de etapa? d) ¿qué componentes la definen? e) Justifique sus respuestas.
3. Cell por ser un Artefact, a) ¿qué atributos tiene? b) ¿qué puede hacer (métodos)? c) ¿qué decide hacer distinto (métodos sobreescritos)? d) ¿qué no puede hacer distinto (métodos finales)? e) ¿qué debe aprendar a hacer (métodos abstractos)? f) Justifique sus respuestas.
4. Por comportarse como un Thing, a) ¿qué atributos tiene? b) ¿qué puede hacer (métodos)? c) ¿qué decide hacer distinto (métodos sobreescritos)? d) ¿qué no puede hacer distinto (métodos finales)? e) ¿qué debe aprendar a hacer (métodos abstractos)? f) Justifique sus respuestas.
5. Ahora vamos a crear dos células activas en diferentes posiciones (1,1) (2,2) llámelos simba y dala usando el método someThings() . Ejecuten el programa, a) ¿Qué pasa con las células? b) ¿Por qué? c) Capturen una pantalla significativa.
6. Diseñen, construyan y prueben el método llamado ticTac() de la clase AManufacturing.
   Incluyan el caso siguiente en sus pruebas de unidad.
7. a) ¿Cómo quedarían simba y dala después de uno, dos y tres `Tic-tac`? Ejecuten el programa y hagan tres clics en el botón. b) ¿Es correcto?. pa) Capturen las pantallas con los resultados de la prueba.

   .

Ciclo 2. Incluyendo a las células mimos [En lab03.doc y replicate.asta]

(NO OLVIDE BDD – MDD)

El objetivo de este punto es permitir recibir células mimos. Las replicadoras (i) son de color naranja; ellas van imitando el estado de sus células vecinas, iniciando en dirección norte y siguiendo las manecillas del reloj.

1. Para implementar esta nueva célula. ¿cuáles métodos se sobre-escriben (overriding)?
2. Diseñen, construyan y prueben esta nueva clase.
   Incluyan el caso siguiente en sus pruebas de unidad.

3. Adicione un trio de mimos activos, llámelos mufasa, scar, rafiki. Coloquelos en línea horizontal. a) ¿Cómo quedarían después de tres **Tic-tac**? Ejecuten el programa y hagan tres clics en el botón. b) ¿Es correcto?. pa) Capturen las pantallas con los resultados de la prueba.

Ciclo 3. Adicionando kriptonita [En lab03.doc, replicate.asta y *.java]

El objetivo de este punto es incluir kriptonita (sólo vamos a permitir el tipo básico de kriptonita). La kriptonita es redonda y verde, sólo se activa si tiene un vecino. En cada etapa se reproduce en las celdas vecinas vacías.

(NO OLVIDE BDD – MDD)

1. Para incluir `Kriptonita` en `AManufacturing.` a) ¿debe cambiar en el código del `AManufacturing` en algo? b) ¿por qué?
2. Diseñen , construyan el método y prueben esta nueva clase.
   Incluyan el caso siguiente en sus pruebas de unidad.
3. Adicionen kriptonita activa en las esquinas del `AManufacturing`, llámenlas `uone, two,three, four` , a) ¿Cómo quedarían después de tres **Tic-tac**? Ejecuten el programa y hagan tres clics en el botón. b) ¿Es correcto?. pa) Capturen las pantallas con los resultados de la prueba.

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. Piensen dos pruebas significativos y a2) expliquen la intención de cada caso. Incluyan las pruebas de unidad asociadas y ejecuten el programa para capturar las pantallas correspondientes.

Ciclo 5. Nuevas Cosas: Proponiendo y diseñando

El objetivo de este punto es permitir recibir en un nuevo Thing (no célula) en el `AManufacturing.`

(NO OLVIDE BDD – MDD)

1. Propongan, describan e Implementen un nuevo tipo de Thing.
2. Incluyan un par de ellos con el nombres semánticos. Piensen dos pruebas significativos y a2) expliquen la intención de cada caso. Incluyan las pruebas de unidad asociadas y ejecuten el programa para capturar las pantallas correspondientes.

Caso 6. Replicate con Rfplicbte

Las células Rfplicbte son azules.

1. Diseñen, construyan y prueben esta nueva clase. Incluyan el caso siguiente en sus pruebas de unidad.
2. Adicionen el caso de la figura inicial y de los cuatro **Tic-tac**, ¿qué pasa? ¿funciona? Capture las cuatro pantallas del ejemplo.

Empaquetando la versión final para el usuario. [En lab03.doc, replicate.asta , *.java, replicate.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

```
replicate
     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.

```
replicate
       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 `replicate` 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]

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. ¿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?