CONOCIENDO:

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? c) ¿Cuál es el propósito del paquete dominio?
   1. Encontramos que el proyecto contiene dos paquetes, uno llamado presentation y el otro llamado domain
   2. Este paquete permite crear la red bidimensional en donde se colocarán las células y partes de domain
   3. Domain se encarga de crear las celulas que se van a poder ver en presentation
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?
   1. En el paquete tenemos 4 componentes (Thing, Cell, Amanufacturing, Artefact). Thing es una interfaz, Artefact es una clase abstracta y Cell va a heredar a Artefact
3. Revisen el paquete de dominio, a) ¿Cuántos componentes tiene? b) ¿Cuántos métodos ofrece?

a. Tiene tres componentes: una interface, una clase abstracta y una clase que hereda a la abstracta.

b. Tiene 18 metodos : la interface nos da los siguientes metodos: decide, change, shape, getColor, isActive, artefact tiene su constructor, step,getSetps y isActive y AManufacturing su constructor, getSize, getThing, neigborsActive, isEmpty, inLatice y ticTac

1. Para ejecutar un programa en java, ¿Qué método se debe ejecutar? ¿En qué clase de replicate se encuentra?
2. Se ejecuta el método main
3. Se encuentra dentro del paquete presentation y en la clase AMAnufacturateGUI
4. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?
5. Por ahora solo ofrece el botón de Tic-Tac
6. No ofrece funciones.
7. Dado a que en constructor privado de presentation solo nos da la función de crear el botón y no le otorga ninguna acción

ARQUITECTURA GENERAL

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.
   1. Un paquete es un conjunto donde se van a agrupar y organizar las clases e interfaces, y se van a dividir por medio de namespaces.
   2. En este caso estamos usando estos paquetes para organizar, por medio de las funcionalidades de cada clase, a las interfaces y clases, estos espacios son domain y presentation.
2. Revisen el contenido del directorio de trabajo y sus subdirectorios. Describa su contenido. ¿Qué coincidencia hay entre paquetes y directorios?
   1. En este caso tenemos uno general en el cual podemos encontrar 3 subdirectorios, doc, en el cual tenemos toda la documentación del proyecto. Otro que tenemos es domain, en este tenemos todas las clases y los respectivos ctxt, .java y .class de cada una, y por último tendríamos, al igual que en domain, a presentation, donde se encuentran las clases que manejan la visualización.
   2. La coincidencia que encontramos es que ambos son lugares de almacenaje, aunque no guardan lo mismo.

ANALISIS DETALLADO

1. Para preparar el poyecto para BDD. Completen el diseño detallado del paquete de dominio. a) ¿Qué componentes hacían falta?
   1. Dentro del diagrama de clases del paquete domain hace falta la clase Cell
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?
   1. Dado a que en el Amanufacturing.java está creando ambas clases, estas son: aManufacturing y PhotoManufacturing.

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?

En domain, dado a que en este paquete esta la base del programa

b) ¿Asociado a qué clase? c) ¿Por qué?

Estaría unido con la clase aManufacturing dado que es el estado de la célula.

**Ciclo 1. Iniciando con las células normales**

1. Estudie la clase AManufacturing ¿Qué tipo de colección usa para albergar las Thing? ¿Puede recibir células? ¿Por qué?

* + Va a guardar en una lista de listas de thing.
  + Si, dado que la clase cell implementa la interface, que, en este caso, es thing, y por el principio de substitución se puede decir que Cell es una Thing sin tener que instanciar Thing, algo que no se puede por ser una interfaz.

2. Estudie el código asociado a la clase Cell

* 1. ¿en qué estado se crea?

Depende de los valores con los que creemos cell, dado que, el estado depende de active, eso significa que el estado de cell va a ser active o inactive según active.

* 1. ¿qué forma usa para pintarse?

Se pinta en forma de un cuadrado de color negro, dado que implementa la interface Thing, y esta tiene definida esta forma

* 1. ¿cuándo cambia de etapa?

Cuando el número de pasos sea múltiplo de 3

* 1. ¿qué componentes la definen?

Lo definen sus atributos nextState, color, aManufacturing, row y colum, y sus métodos.

3. Cell por ser un Artefact,

* 1. ¿qué atributos tiene?

Además de los propios, tiene steps, stage, unknown, active e inactive.

* 1. ¿qué puede hacer (métodos)?

De los métodos que hereda de la clase abstracta, puede obtener el número de pasos, avanzar un paso y decir si está activo o no.

* 1. ¿qué decide hacer distinto (métodos sobrescritos)?

Los métodos de la interface change y getColor.

* 1. ¿qué no puede hacer distinto (métodos finales)?

No puede cambiar el comportamiento del método isActive, el cual dice si está activo o no, ni tampoco el de getSteps, para obtener el número de pasos.

* 1. ¿qué debe aprender a hacer (métodos abstractos)? f) Justifique sus respuestas.

4. Por comportarse como un Thing,

1. ¿qué atributos tiene?

Tiene dos ROUND y SQUARE, los cuales me determinaran la forma que van a tomar

1. ¿qué puede hacer (métodos)?

Puede decidir su comportamiento y estado, decide, puede cambiar su estado, change, puede devolver su forma, shape, puede devolver su color, getColor, y puede decir si esta activa, isActive.

1. ¿qué decide hacer distinto (métodos sobrescritos)?

Cada Thing puede decidir cómo se va a comportar, decide, como va a cambiar de estado, change, que color va a retornar, getColor, la forma que va a tener, shape, y si esta activa, isActive.

1. ¿qué no puede hacer distinto (métodos finales)?

Tiene que aplicar todos los métodos que están por defecto en la interfaz, pero ya depende de que clase la implemente para ver como funcionaran.

1. ¿qué debe aprender a hacer (métodos abstractos)? f) Justifique sus respuestas.

Tiene que aprender a decidir sus comportamientos, decide, como va a cambiar de estado, change, que color va a retornar, getColor, la forma que va a tener, shape, y si esta activa, isActive.

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.

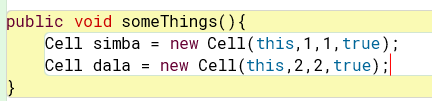
1. ¿Qué pasa con las células?

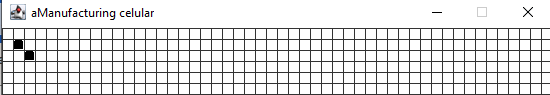
Se crearon, se colocaron de manera correcta en las posiciones indicadas, y quedaron pintadas en los cuadrados especificados.

b) ¿Por qué?

Dado que para crear una cell se necesita primero haber creado un aManufacturing.

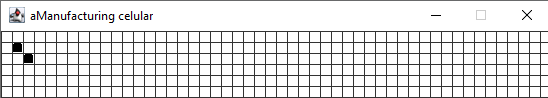
c) Capturen una pantalla significativa.





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.

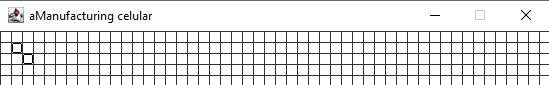


Al principio aparecerían activas.



Tic\_tac = 0

Al oprimir el botón por primera vez se activarán las células.

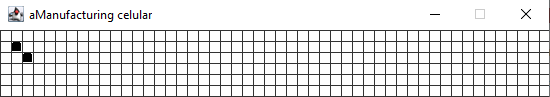


Tic\_tac = 1

Ya después de oprimirlo se considerará esta vez como el primer momento.



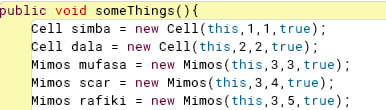
Tic\_tac = 2



Tic\_tac = 3

**Ciclo 2. Incluyendo a las células mimos**

1. Para implementar esta nueva célula. ¿cuáles métodos se sobre escriben (overriding)?
   1. Se debe sobrescribir el método decide, dado que su comportamiento va a ser distinto al de las células normales.
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.







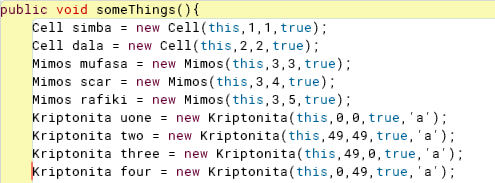


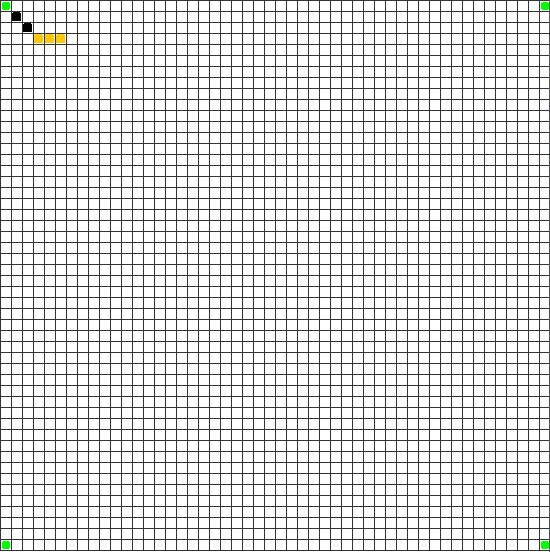


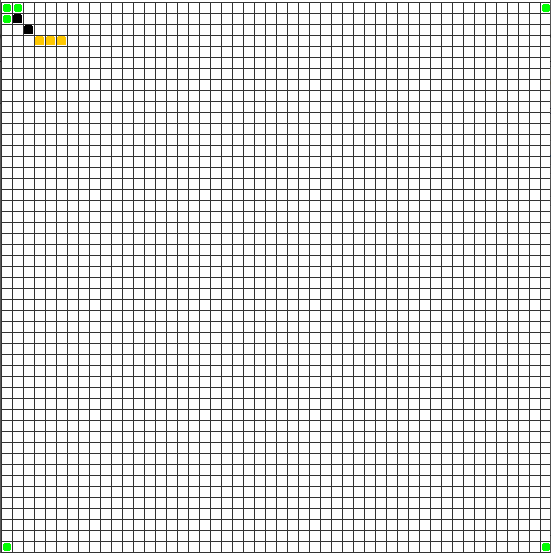


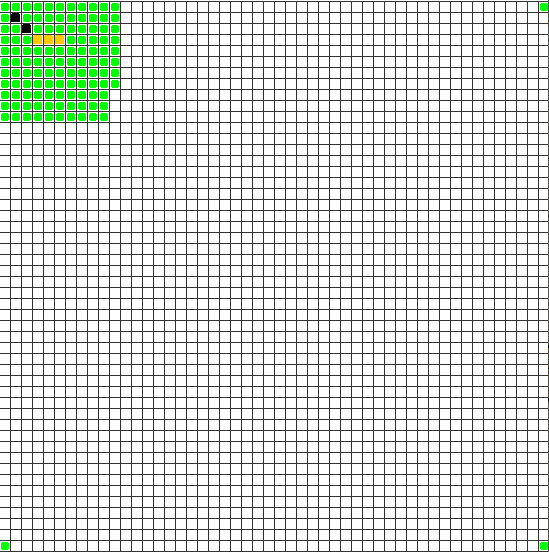
**Ciclo 3. Adicionando kriptonita**

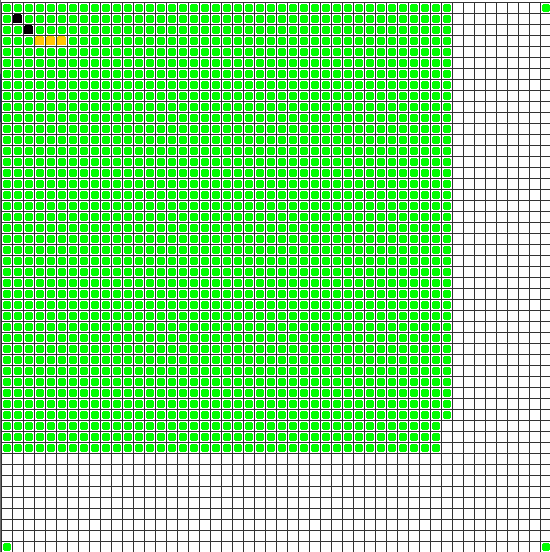
1. Para incluir Kriptonita en AManufacturing. a) ¿debe cambiar en el código del AManufacturing en algo? b) ¿por qué?
   1. Aparte de modificar el metodo Tic-tac, no se tendría que cambiar otra parte de AManufacturing.
2. Diseñen, construyan el método y prueben esta nueva clase. Incluyan el caso siguiente 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.





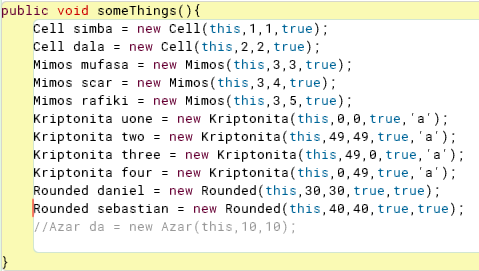


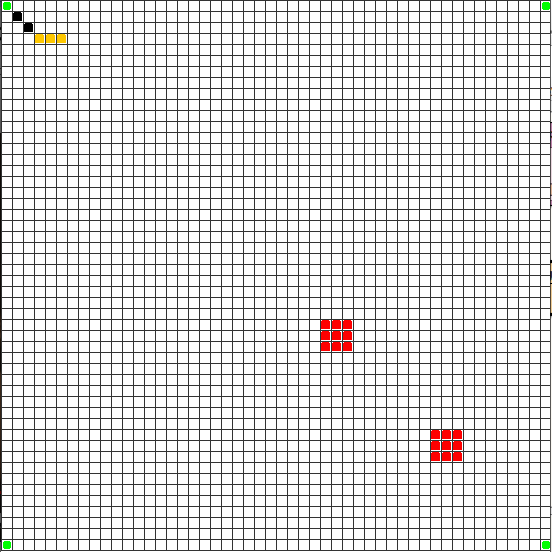


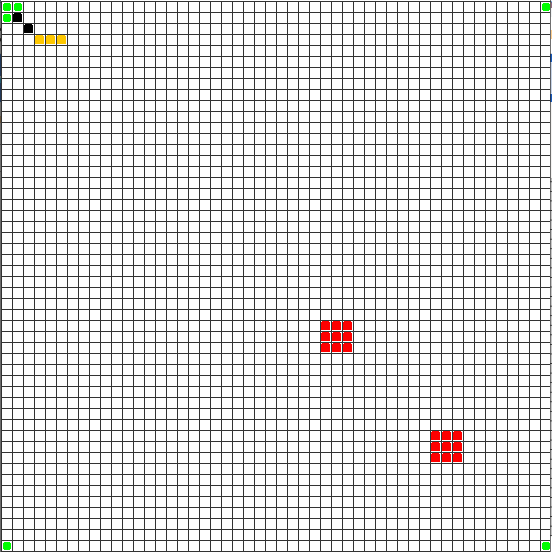


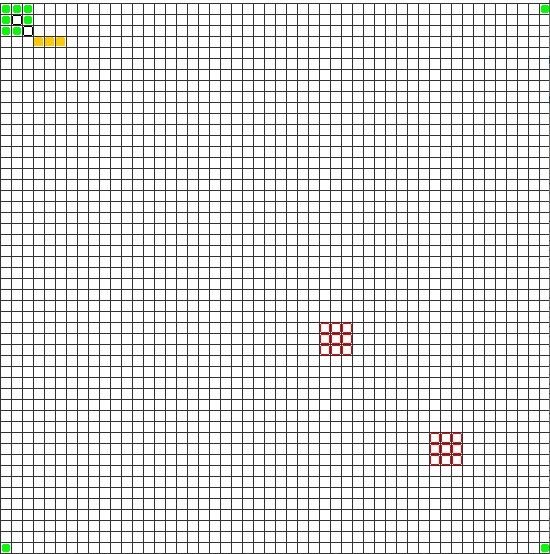
**Ciclo 4. Nueva Célula: Proponiendo y diseñando**

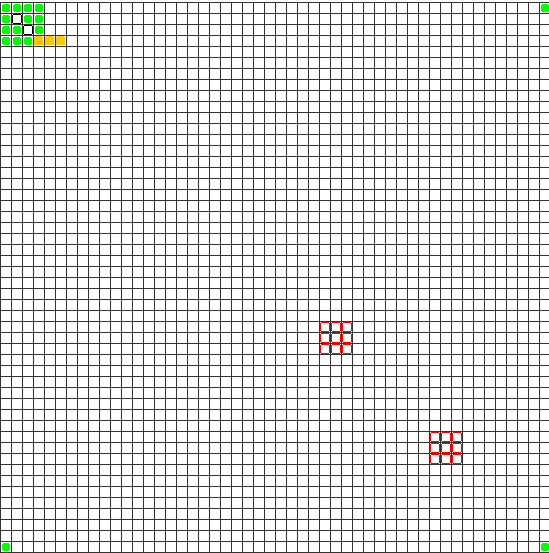
* 1. Propongan, describan e Implementen el nuevo tipo de células.
     1. La nueva célula se va a llamar Rounded, y cuando se crea va a crear nuevas Rounded alrededor de la que creamos al principio y se van a activar y desactivar de la misma forma que una célula normal.
  2. Incluyan una pareja de ellas 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.

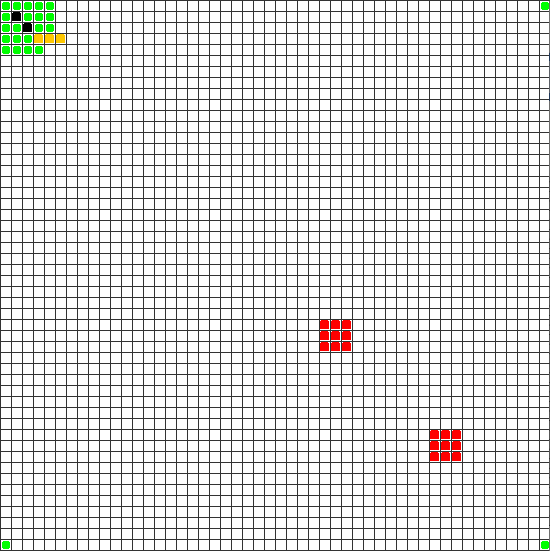






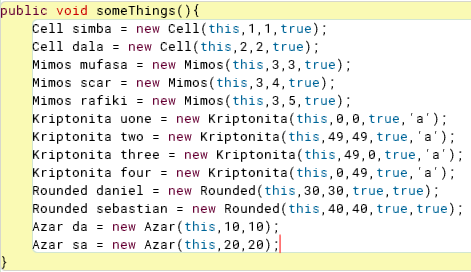


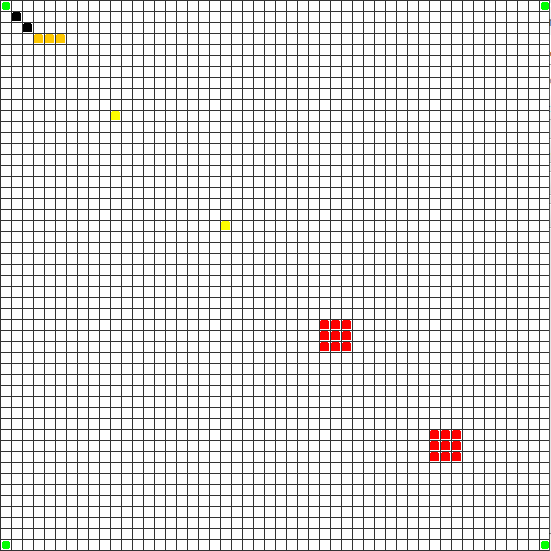


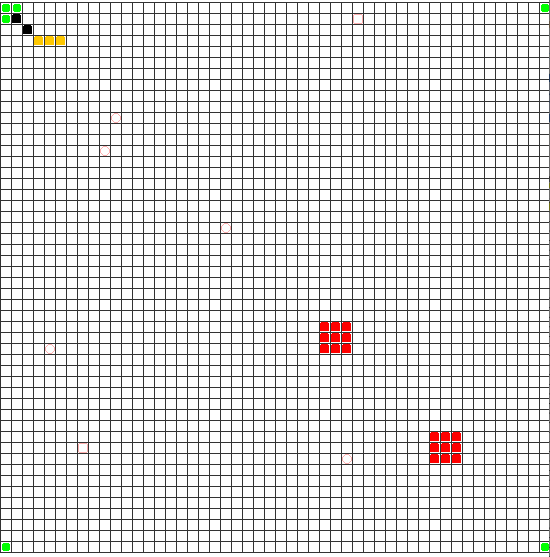


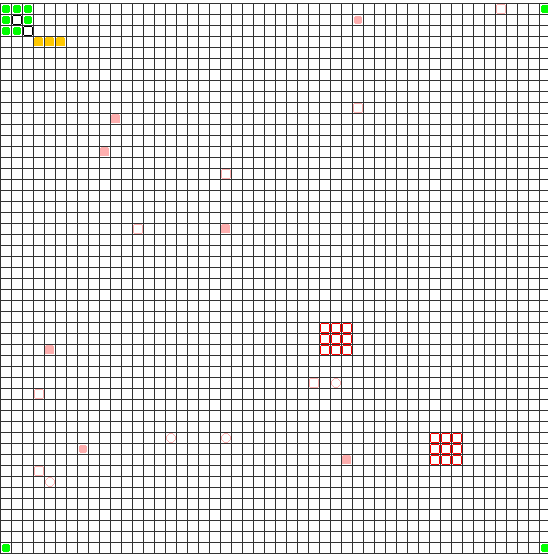
**Ciclo 5. Nuevas Cosas: Proponiendo y diseñando**

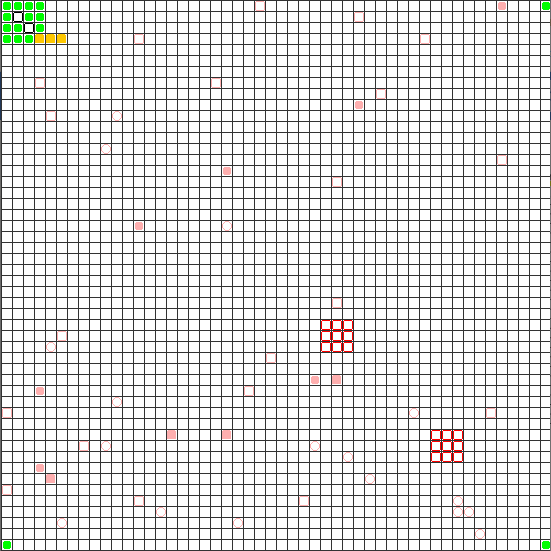
* 1. Propongan, describan e Implementen un nuevo tipo de Thing.
     1. La nueva Thing se va a llamar Azar. Va a reproducirse al azar en los espacios donde no se encuentre nada, va a ir cambiando entre un cuadrado y un círculo, y va a empezar con un color amarillo y después lo cambiara por un rosado.
  2. Incluyan un par de ellos con los 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.

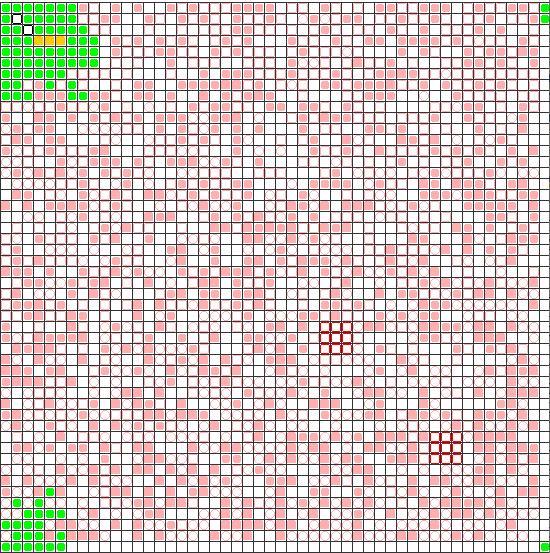






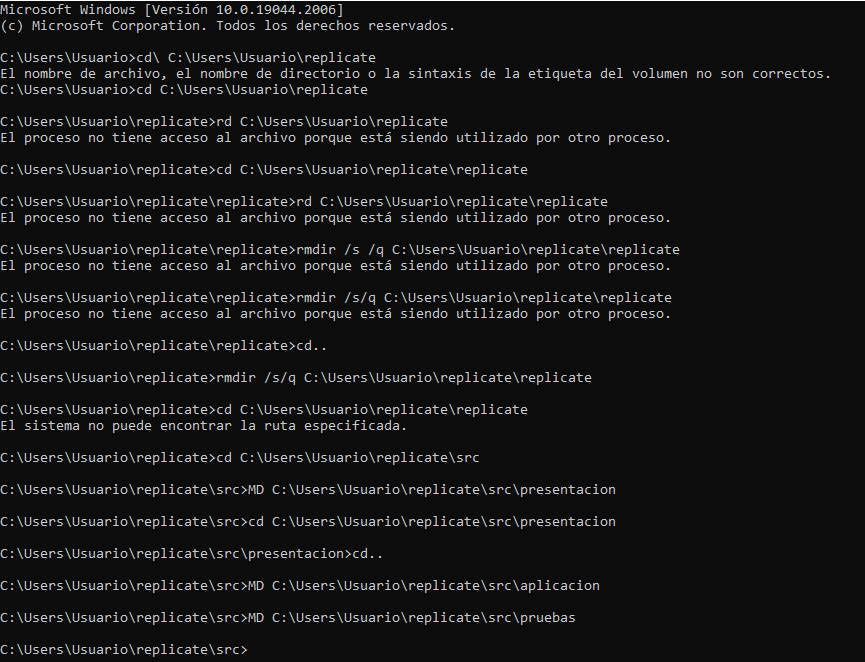






**DE BLUEJ A CONSOLA.**

**Comandos básicos del sistema operativo**

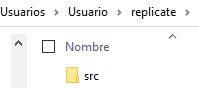
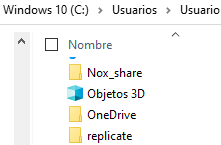
1. Investiguen los comandos para moverse en la estructura de directorios: crear, borrar, listar su contenido y copiar o eliminar un archivo. 
2. Organicen un nuevo directorio con la estructura propuesta para probar desde allí su habilidad con los comandos de consola. Consulten y capturen el contenido de su directorio replicate

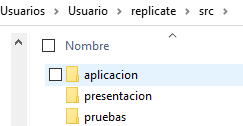
src

aplicacion

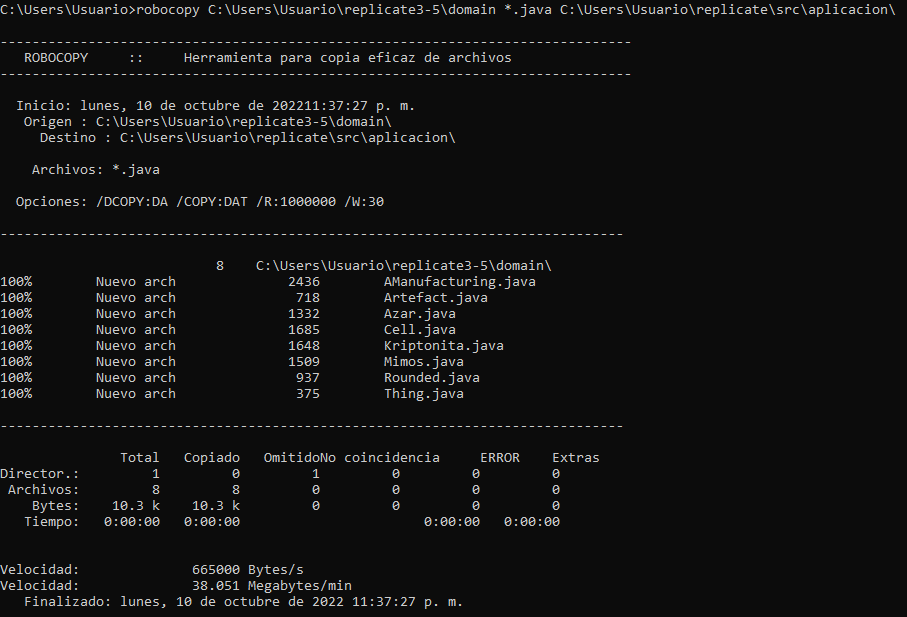
presentacion

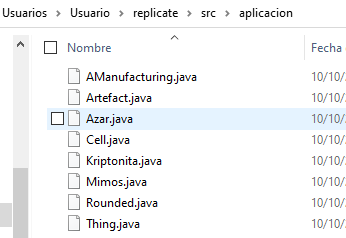
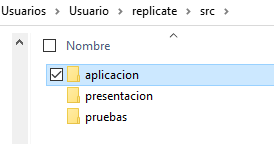
pruebas

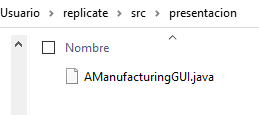




1. En el directorio copien únicamente los archivos \*.java del paquete de aplicación. Consulte y capture el contenido de src/aplicacion







**Estructura de proyectos java**

En 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.
     1. En la carpeta src deben ir todos los archivos .java del proyecto
     2. La carpeta bin contiene los archivos que se generan automáticamente a partir de los metadatos del lenguaje de definición de datos (DDL) durante un proceso de construcción. Los archivos de cada script se conservan en su propio directorio, cuyo nombre refleja la carpeta core/ddl.
  2. ¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué? Cópielos y consulte y capture el contenido del directorio que modificó.