**Laboratorio 3**

**Santiago Gualdron y Laura Sofia Gil**

**CONOCIENDO**

1. En el directorio descarguen los archivos contenidos en garden.zip. Revisen el código:

a) ¿Cuántos paquetes tiene?

El código contiene 2 paquetes presentación y dominio

b) ¿Cuál es el propósito del paquete presentación?

El propósito general del paquete es una interfaz gráfica para el usuario donde tenemos 1 paquete que es go up (siendo un paquete para devolverse al menú principal) y una clase Garden Gui, la cual tiene como propósito visualizar y controlar lo que puede estar pasando en el jardin.

c) ¿Cuál es el propósito del paquete dominio?

El propósito del paquete domain es una representación del código detrás que tiene la interfaz gráfica, que tiene 5 clases que se relacionan entre sí (Flower, Agent,thing , wáter ,garden) y un paquete go up para devolvernos al menú principal donde se encuentran los dos paquetes (domain y presentation).

2. Revisen el paquete de dominio,

a)¿Cuáles son los diferentes tipos de componentes de este paquete?

Los diferentes componentes son el botón go up, las clases Flower, Garden, Water y una interfaz llamada Thing que esta implementada por la clase Water y por Flower, también una de las clases es una abstracta ( la cual es Agent) y asi mismo esta extendida por la misma clase Flower

b) ¿Qué implica cada uno de estos tipos de componentes?

Lo que implica cada uno de estos componentes Thing que es una interfaz la cual hace una serie de métodos abstractos y propiedades constantes que permiten el uso de herencia múltiple que en sintesis nos permite que la misma interfaz sea implementada por dos clases diferentes, lo que significa que diferentes clases pueden comenzar desde la misma estructura, la clase abstracta que es Agent y es extendida por la clase Flower y las otras 3 clases Flower, Garden, Water que se relacionan entre si, derivando diferentes funcionalidades. En este caso implica que al menos la clase flower si o si implemente todos los metodos de la interfaz y todos los métodos abstractos de la clase abstracta que está intentando o al menos pretende extender.

3. Revisen el paquete de presentación,

a) ¿Cuántos componentes tiene?

Los diferentes componentes son el paquete go up y la clase GardenGUI que se deriva de la herencia de Jframe la cual es una interfaz gráfica para añadir distintos objetos con los que podrá interactuar o no el usuario, a su vez encontramos la clase PhotoGarden que se encuentra dentro de la misma clase de GardenGUI.

b) ¿Cuántos métodos públicos propios (no heredados) ofrece?

Los métodos privados son 4 y hay 2 públicos, hay que tener en cuenta que todos los métodos que son ofrecidos por JFrame también son métodos que están implícitos a pesar de que no sean físicamente visibles, además también está en existencia una clase adicional que tiene como nombre PhotoGarden que extiende JPanel, es decir es una herencia de JPanel y tiene acceso completo a todos sus métodos.

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

Se debe ejecutar la clase de GardenGUI en este caso el método main dado que nos proporciona la capacidad de ejecutar correctamente la aplicación. Al ejecutar el programa, se comienza la ejecución desde este método. Este procedimiento implica la creación de una instancia de la clase GardenGUI y su posterior visibilidad, lo que permite la presentación de la interfaz gráfica de la simulación.

5. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ofrece las funcionalidades de crear y ejecutar correctamente (gráficamente hablando) la posibilidad de crear el tablero y el botón tic tac que nos permitirá el ejecutar correctamente el movimiento, sin embargo, lo único que hace por ahora es ejecutar un nuevo GUI, preparar cada uno de los interfaces gráficos que son necesarios para que se vea así. El botón de tic tac sigue siendo solo eso un botón, dado que no le hemos proporcionado ninguna funcionalidad

**ARQUITECTURA GENERAL**

1. Consulte el significado de las palabras package e import de java. ¿Qué es un paquete? ¿Para qué sirve? ¿Para qué se importa? Explique su uso en este programa.

-Package (paquete): Un package en Java es un mecanismo de organización de clases y recursos relacionados en una estructura jerárquica. Los paquetes se utilizan para evitar conflictos de nombres y para organizar el código de una manera lógica y modular. Cada clase en Java pertenece a un paquete, ya sea explícitamente especificado o al paquete predeterminado si no se especifica ninguno.

-Import (importar): Un import se utiliza en Java para incluir clases o miembros de clases de otros paquetes en el código actual. Permite que el compilador de Java sepa dónde encontrar las clases que estás utilizando en tu código. Sin la declaración import, se tendría que especificar el nombre completo de la clase cada vez que se use, lo que sería incómodo y propenso a errores.

En el programa proporcionado, se emplean paquetes y declaraciones de importación para organizar y acceder a las clases y bibliotecas necesarias. Los paquetes ayudan a agrupar clases relacionadas, mientras que las declaraciones de importación permiten el uso de clases y bibliotecas de Java de manera más sencilla sin requerir nombres completos. Esto mejora la organización y la legibilidad del código.

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

En la carpeta de directorio se encuentra todo el contenido de la visualización de los paquetes principales que tenemos, donde tenemos 3 subdirectorios domain, presentación y doc donde cada uno de estas carpetas tiene sus respectivos archivos. Dentro de la carpeta de "Garden," se halla un archivo Astah que contendrá los paquetes correspondientes a "domain" y "presentation." Al abrirlo, se revelan varios tipos de archivos y un proyecto en BlueJ. Además, en la carpeta "doc," se guarda toda la documentación integral del proyecto.

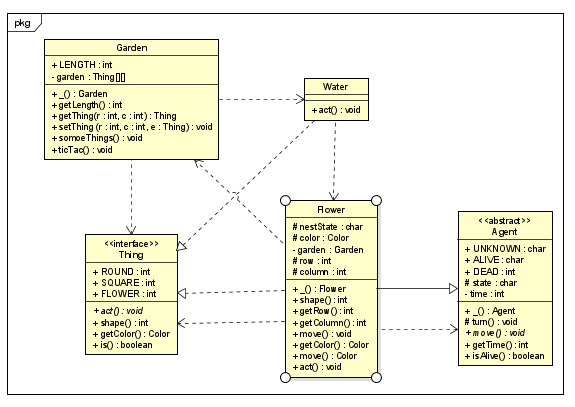
3. Adicione al diseño la arquitectura general con un diagrama de paquetes en el que se presente los paquetes y las relaciones entre ellos. Consulte la referencia en moodle. En astah, crear un diagrama de clases (cambiar el nombre por Package Diagram0)

Diagrama

Descripción generada automáticamente

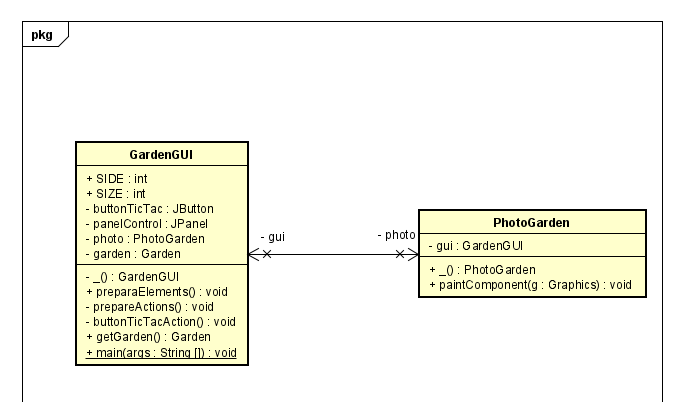
**ARQUITECTURA DETALLADA.**

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?



Los componentes que hacían falta en el diagrama de clases son: las clases concretas Garden, Flower y Water, la clase abstracta Agent y la interfaz Thing.

2. Completen 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?



Tenemos dos clases porque la clase GardenGUI es la clase principal de la interfaz de usuario de la aplicación y se encarga de crear y gestionar la ventana de la interfaz de usuario, los botones y otros componentes visuales. Crea en si un jardin, mientras que PhotoGarden está diseñada para manejar la representación visual de la simulación en un panel dentro de la interfaz gráfica. Esta clase es responsable de dibujar la simulación en la pantalla y puede incluir métodos para pintar entidades y actualizar la visualización.

El archivo .java en el paquete "presentation" puede incluir otras clases que se emplean en el contexto de la interfaz de usuario. Es una práctica común combinar múltiples clases en un archivo .java cuando estas clases están relacionadas entre sí o son pequeñas y se utilizan conjuntamente.

3. Adicione la clase de pruebas unitarias necesaria para BDD en un paquete independiente de test. (No lo adicione al diagrama de clases) ¿Qué paquete debe usar? ¿Por qué? ¿Asociado a qué clase? ¿Por qué?

La clase de pruebas unitarias debería estar asociada al paquete dominio y específicamente a la clase Garden. Dado que Flower es la clase responsable dominio donde realiza toda la lógica conjunta de la aplicación, y las pruebas unitarias deberían realizarse para verificar el comportamiento de la aplicación. Al crear una clase de pruebas asociada a Flower, se debería verificar que la aplicación funcione según lo esperado en diferentes situaciones.

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

1. Estudie la clase garden ¿Qué tipo de colección usa para albergar cosas? ¿Puede recibir flores? ¿Por qué?

La clase Garden utiliza un arreglo bidimensional (Thing[][] garden) para almacenar las cosas (objetos de tipo Thing) en el jardín. La matriz bidimensional garden de la clase Garden no puede recibir directamente instancias de Flower u otras clases que implementen la interfaz Thing. Esto se debe a que en Java, una matriz bidimensional almacena referencias a objetos en lugar de los objetos mismos. Para agregar una instancia de Flower o cualquier otra entidad que implemente Thing a la matriz, se debe seguir un proceso que implica primero crear la instancia y luego asignar esa instancia a una celda específica de la matriz. Para llevar a cabo esto, se puede utilizar el método setThing(int r, int c, Thing e) dentro de la clase Garden que acepte las coordenadas de la celda y la entidad que se desea asignar a esa celda en particular.

2. Estudie el código asociado a la clase Flower, ¿en qué estado se crea? ¿qué forma usa para pintarse? ¿cuándo aumenta su tiempo? ¿qué clases definen la clase Flower ? Justifique sus respuestas.

-Se crea bajo el estado de un protected char nextState donde se declara que el agente está vivo al momento de instanciarlo.

- La forma que usa la flor para pintarse se determina mediante el método shape(), que la clase Flower implementa de la interfaz Thing.

- No se proporciona en el código una lógica específica para aumentar el tiempo de la flor, por lo que no se puede determinar cuándo ocurre este aumento en la implementación actual.

- Agent: La clase Agent parece ser una clase base o superclase de la cual Flower hereda, ya que Flower extiende Agent. Probablemente, la clase Agent define comportamientos generales para las entidades del jardín.

-Thing: La clase Thing parece ser una interfaz que Flower implementa. Esta interfaz podría definir métodos comunes que todas las cosas (Things) en el jardín deben tener, como shape(), getRow(), getColumn(), etc.

3. Flower 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

aprendar a hacer? Justifique sus respuestas.

-Los atributos y métodos específicos de Agent no están explícitamente definidos en el código proporcionado para Flower, pero se asume que Flower hereda estos atributos y métodos de su clase base Agent.

-Flower puede decidir hacer cosas distintas basándose en su comportamiento específico como una flor, como implementar métodos relacionados con la floración, polinización, etc.

-Flower no puede realizar acciones completamente fuera del contexto de un agente genérico, como operaciones de entrada/salida directamente.

-Flower debe aprender a implementar la lógica para los métodos move() y act() para tener un comportamiento completo como un agente dentro del sistema.

Los métodos

-Constructor Flower(Garden garden, int row, int column): Inicializa el estado del agente como ALIVE y el tiempo como 0, además de inicializar su posición en el jardín.

-Método turn(): Incrementa en uno el tiempo de vida del agente cada vez que se llama.

-Método move(): Este método es abstracto en Agent, por lo que Flower debe implementarlo según su comportamiento específico de movimiento.

-Métodos getTime() y isAlive(): Hereda estos métodos de Agent para obtener el tiempo de vida actual del agente y verificar si está vivo o no.

Qué decide hacer distinto Flower:

Como Flower es una flor, podría decidir implementar su propio comportamiento de movimiento en el método move(), como moverse hacia la luz solar o hacia un suelo fértil. Además, podría implementar métodos específicos como bloom() para simular el proceso de floración o pollinate() para simular la polinización.

Qué no puede hacer distinto a todos los agentes:

Flower, al heredar de Agent, tiene ciertas limitaciones en cuanto a lo que no puede hacer de manera distinta a todos los agentes. Por ejemplo, no puede cambiar la lógica básica de incrementar el tiempo de vida en el método turn(), ya que este es un comportamiento común a todos los agentes. Tampoco puede modificar directamente el estado UNKNOWN, ALIVE o DEAD sin modificar la lógica base de Agent.

Qué debe aprender a hacer Flower:

Flower debe aprender a implementar el método move() según el comportamiento específico que se espera de una flor en el sistema. Además, puede aprender a implementar otros métodos específicos de una flor, como bloom() o pollinate(), para tener un comportamiento completo y realista dentro del contexto del sistema.

4. Por comportarse como un -Thing, ¿qué sabe hacer? ¿qué decide hacer distinto? ¿qué

no puede hacer distinto? ¿qué debe aprender a hacer? Justifique sus respuestas.

La interfaz Thing define ciertos métodos de una clase que la implementa debe proporcionar. En este caso, Flower debe implementar los siguientes métodos:

* act(): Define el comportamiento general de la entidad en cada paso de la simulación. En el contexto de Flower, este método debe definir las acciones que realiza una flor en cada paso.
* shape(): Retorna un valor que indica la forma de la entidad. En este caso, Flower devuelve Thing.FLOWER para indicar que tiene la forma de una flor.
* getColor(): Retorna el color que se utiliza para representar gráficamente la entidad. En Flower, el color es Color.red.
* is(): Retorna un booleano, en este caso esta predefinido como “true”, que sirve para confirmar el estado.

Como un Thing, la clase Flower puede hacer de diferente los métodos de act(), ya que este coordina el movimiento de las flores, además tiene la diferencia de shape(), puesto que al estar en Flower, retornamos la figura de la flor.

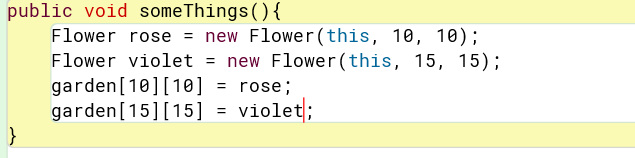
Como un Thing, la clase Flower no cambia con respecto al método getColor(), al retornar el color que tiene predefinito el objeto, además del método is(), ya que este ni siquiera esta empleado inicialmente en la clase Flower.

La clase Flower, debe aprender a implementar los métodos act(), arreglar su return y reliazacion del método, con respecto al método getColor(); al igual que el método shape(), están predefinidos en la interfaz, pero cambian a la hora de ser implementados en la clase; y por ultimo implementar el método is(), donde puede cambiar el return, por el estado visible o no de la flor.

5. Ahora vamos a crear dos flores en diferentes posiciones (10,10) (15,15) llámelas

rose y violet usando el método someThings() . Ejecuten el programa, ¿Qué pasa con

las flores?¿Por qué? Capturen una palla significativa.



Gráfico

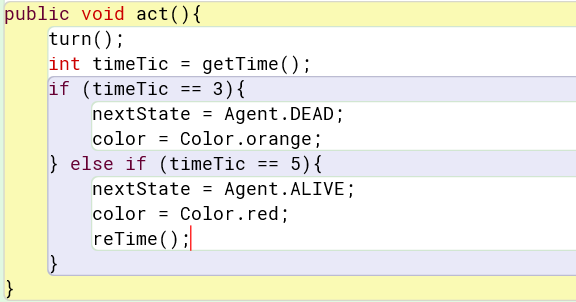
Descripción generada automáticamente

Se crearon 2 flores con las coordenadas especificadas (10,10) y (15, 15) en el tablero (Garden) donde también se guardaron en el arraylist bidimensional.

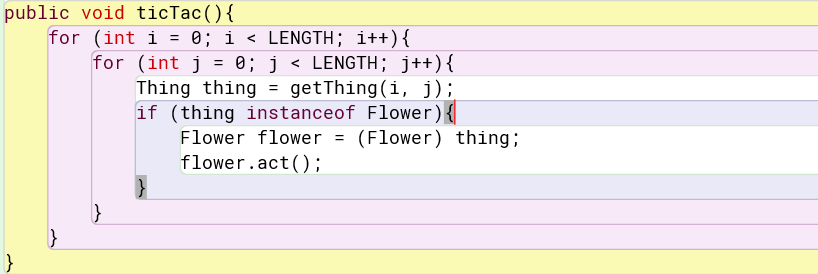
6. Diseñen, construyan y prueben el método llamado act() de la clase Flower. La flor

cambia de color de rojo a naranja, se marchita después de 3 tic-tac y revive después

de 5 tic-tac.



7. Diseñen, construyan y prueben el método llamado ticTac() de la clase garden.

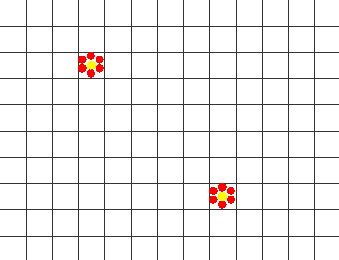


8. ¿Cómo quedarían rose y violet después de uno, dos, cuatro y seis Tic-tac?

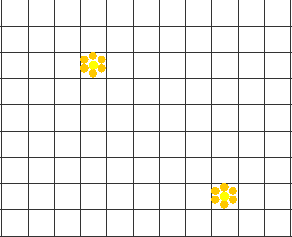
Ejecuten el programa. Capturen pantallas significativas en momentos

correspondientes. ¿Es correcto?

Después de 1 u 2 Tic-tac



Después de 4 Tic-tac



Después de 6 Tic-tac

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

**Ciclo 2. Incluyendo a las flores carnivoras**

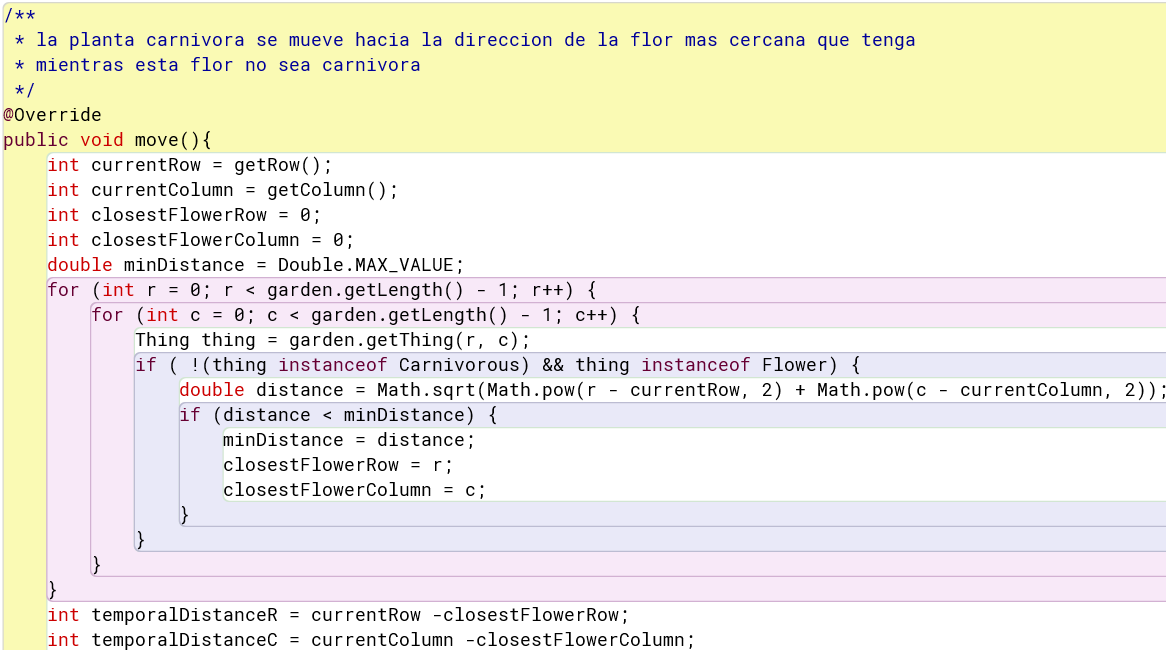
El objetivo de este punto es permitir recibir en el jardín forles carnivoras . Ellas (i) son de color azul; (ii) se mueven hacia la flor más cercana, (iii) si pueden se la comen (ocupan su posición)

1. Para implementar esta nueva flor Carnivorous ¿cuáles métodos se sobre-escriben (overriding)?

Para crear esta nueva Clase de Planta Carnivora, extendida de Flower, se puede hacer sobreescritura de los métodos act() y move().

1. Diseñen, construyan y prueben esta nueva clase.

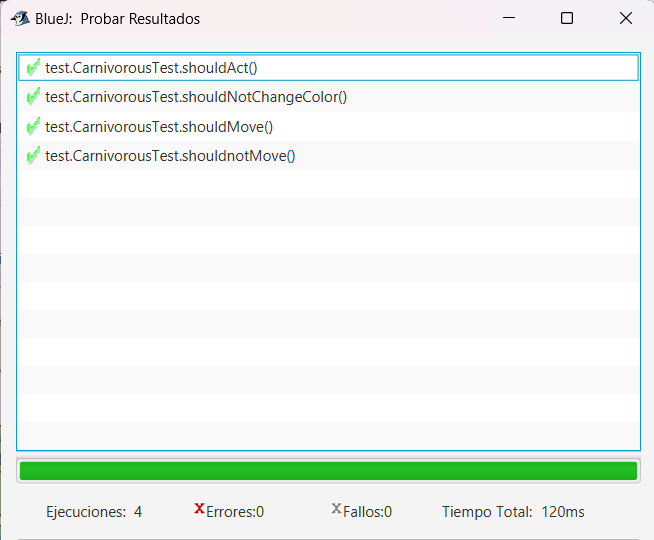
Métodos de la clase Carnivorous: move() y act()



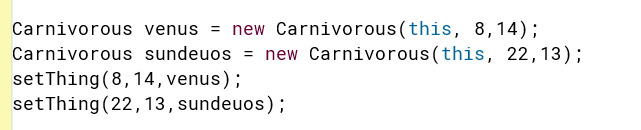
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Pruebas de unidad:



1. Adicione una pareja de inquietas, llámelas venus y sundeuos, (a) ¿Cómo quedarían después de tres Tic-tac? Ejecuten el programa y hagan tres clics en el botón. Capturen una palla significativa. (b) ¿Es correcto?



Después de 0 Tic-tac

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Después de 3 Tic-tac

Gráfico, Gráfico de dispersión

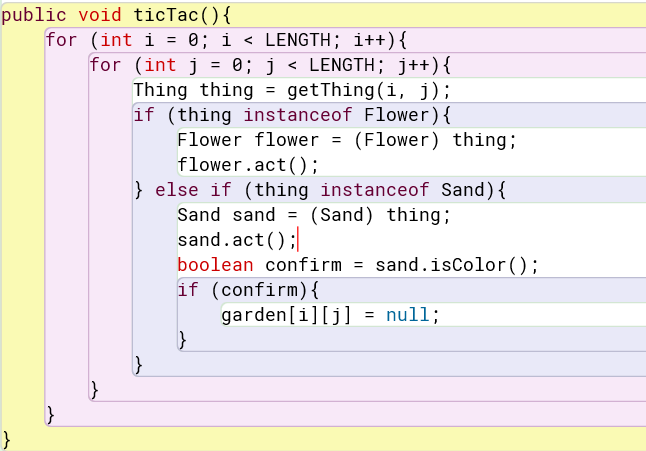
Descripción generada automáticamente

**Ciclo 3. Adicionando arena**

El objetivo de este punto es incluir en el garden zonas de arena (sólo vamos a permitir el tipo básico de arena) los bancos de arena inician grises y a medida que pasa el tiempo van perdiendo su color.

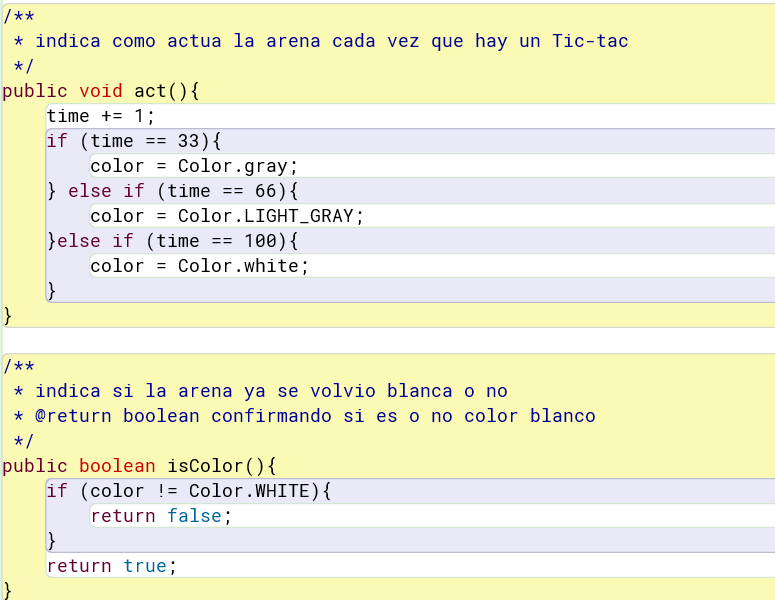
1. Construyan la clase Sand para poder adicionarla en el garden ¿debe cambiar en el código del garden en algo? ¿por qué?

Para el código en Garden, se debe colocar un nuevo condicional el cual identifique el objeto Thing de Sand, si no se coloca este condicional, Sand pasaría desapercibido y no podría cambiar de color cada cierto tiempo.



1. Diseñen , construyan el método y prueben esta nueva clase.

Método act() y isColor() de Sand, los métodos diferentes de esta clase.



Pruebas de unidad:

Interfaz de usuario gráfica, Texto, Aplicación

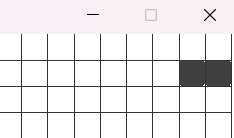
Descripción generada automáticamente

3. Adicionen dos parches de arena en las esquinas superiores del garden, llámenlos tatacoa y sahara, (a) ¿Cómo quedarían después de cuatro Tic-tac? Ejecuten el programa y hagan cuatro clics en el botón. Capturen una palla significativa. (b) ¿Es correcto?

Texto

Descripción generada automáticamente

Al hacer 4 Tic-tac en los montículos de Sand, no se genera un cambio significativo, puesto que para que la arena cambie de color debe pasar mínimo 33 Tic-tac



Sand después de 33 Tic-tac

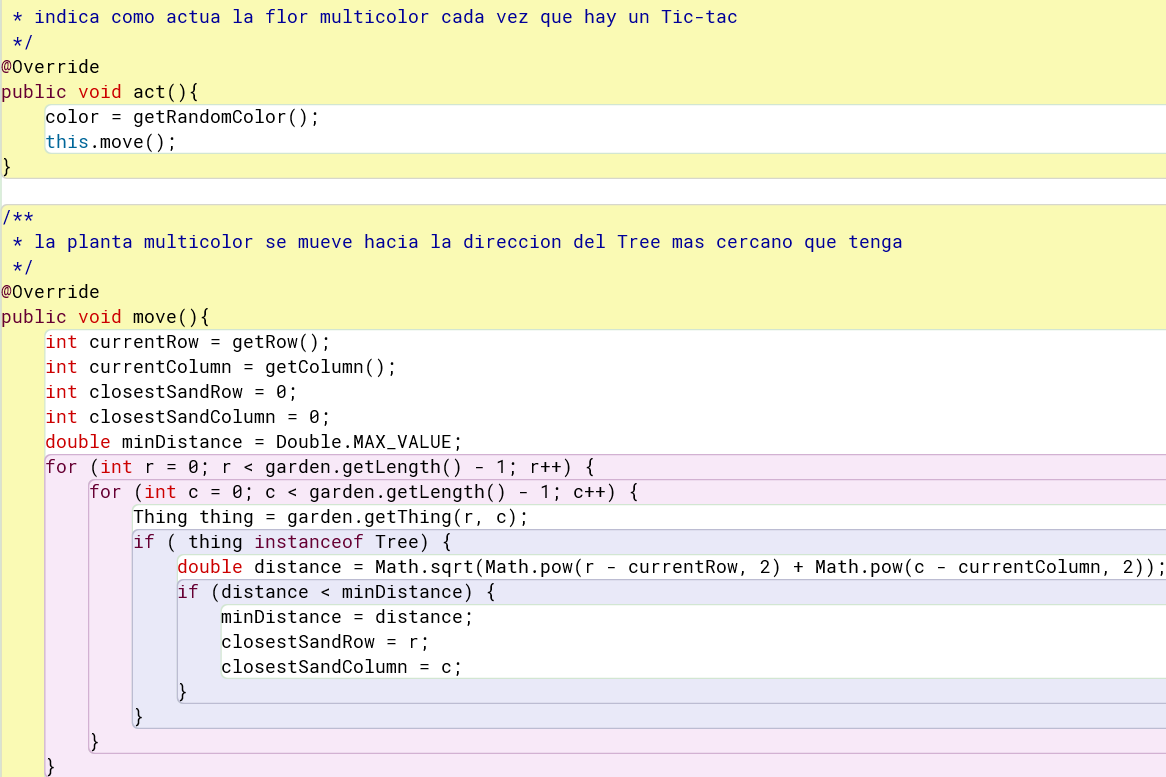


**Ciclo 4. Nueva flor: Proponiendo y diseñando**

1. Propongan, describan e Implementen el nuevo tipo de flores. (Mínimo dos pruebas de unidad)

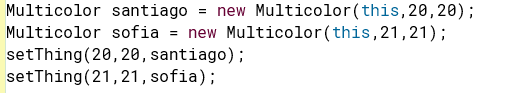
Nuestro nuevo tipo de flor se llama multicolor la cual hace por cada tic tac un cambio de color y se dirige a la entidad Tree.

Acá se muestran los códigos de la implementación de multicolor en: act() y move():

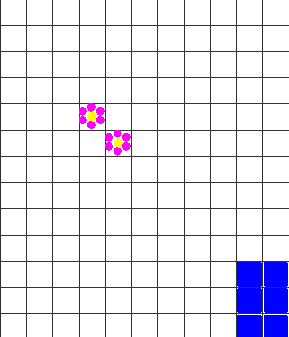
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

2. Considerando una pareja de ellas con el nombre de ustedes. (a) Piensen una prueba significativa y expliquen la intención. (b) Codifiquen las pruebas de unidad correspondiente y capturen la palla de resultados. (c) Ejecuten el programa con esa prueba para capturar los pantallazos correspondientes.



Flores multicolor después de 1 Tic-tac:



Flores multicolor después de 2 Tic-tac:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Flores multicolor después de 3 Tic-tac:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Pruebas de unidad:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Ciclo 5. Nueva entidad: Proponiendo y diseñando**

1. Propongan, describan e Implementen un nuevo tipo de Entity. (Mínimo dos pruebas de unidad).

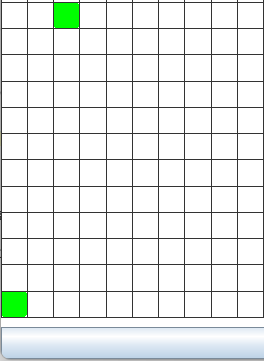
Nuestra nueva entidad es un árbol que cambia de color (naranja) cada 5 tic tac y renace cada 10(vuelve a verde)

2. Considerando un par de ellos con los nombres de ustedes . (a) Piensen una prueba significativa y expliquen la intención. (b) Codifiquen la pruebas de unidad correspondiente y capturen la palla de resultados. (c) Ejecuten el programa con esa prueba para capturar las pallas correspondientes.

Texto

Descripción generada automáticamente

Imágenes de los Trees después de 1 Tic-tac



Los Trees después de 50 Tic-tac:

Imagen que contiene biombo, edificio

Descripción generada automáticamente

Pruebas de unidad:

Patrón de fondo

Descripción generada automáticamente

**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.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

2. Consulte el comando java para ejecutar un archivo jar. ejecútenlo ¿qué pasa?

El comando utlizado es “java -jar nombre-del-archivo.jar, pasa que se abre instantáneamente el archivo.

3. ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.

Los archivos JAR en Java son ventajosos debido a su capacidad para empaquetar, distribuir y ejecutar aplicaciones de manera eficiente. Ofrecen organización, gestión de dependencias, portabilidad y seguridad, lo que simplifica el desarrollo y despliegue de aplicaciones Java.

**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 directorios:

* mkdir nombre\_del\_directorio: Crea un nuevo directorio con el nombre especificado.

Borrar directorios y archivos:

* rm -r directorio (Unix/Linux) o rmdir /s /q directorio (Windows): Borra un directorio y su contenido de manera recursiva.
* rm archivo (Unix/Linux) o del archivo (Windows): Borra un archivo.

Listar el contenido de un directorio:

* ls (Unix/Linux) o dir (Windows): Muestra el contenido del directorio actual.
* ls ruta/al/directorio (Unix/Linux) o dir ruta\al\directorio (Windows): Muestra el contenido de un directorio específico.

Navegar por los directorios:

* cd directorio: Cambia al directorio especificado.
* cd ..: Regresa al directorio padre.
* cd ~: Cambia al directorio de inicio del usuario.
* pwd (Unix/Linux) o echo %cd% (Windows): Muestra la ruta actual del directorio.

Copiar archivos:

* cp origen destino (Unix/Linux) o copy origen destino (Windows): Copia un archivo desde la ubicación de origen a la ubicación de destino.

Mover o renombrar archivos:

* mv origen destino (Unix/Linux) o move origen destino (Windows): Mueve o renombra un archivo o directorio.

Eliminar archivos:

* rm archivo (Unix/Linux) o del archivo (Windows): Borra un archivo.
* rmdir directorio (Unix/Linux) o rmdir /s /q directorio (Windows): Borra un directorio vacío.

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

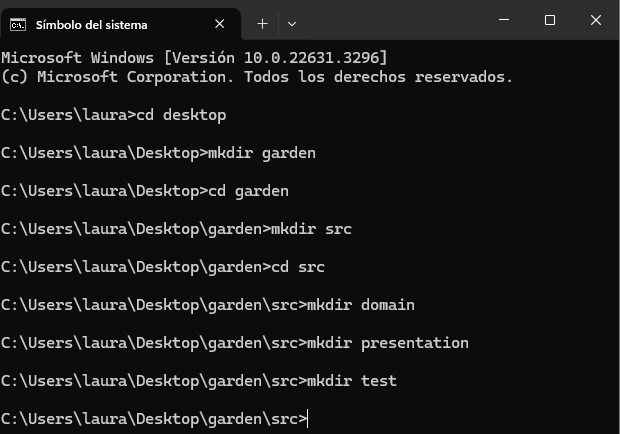
garden

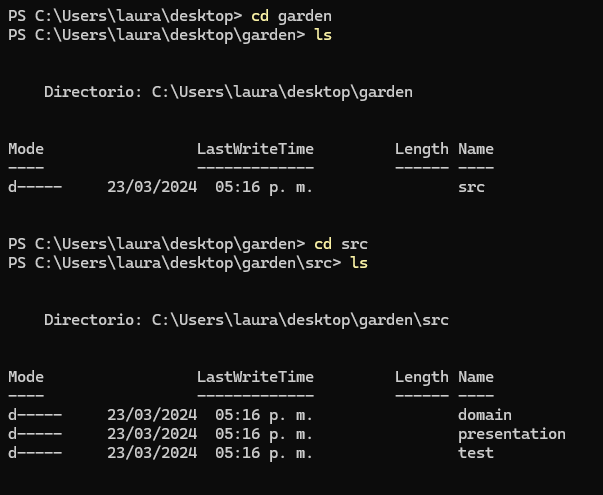
src

domain

presentation

test

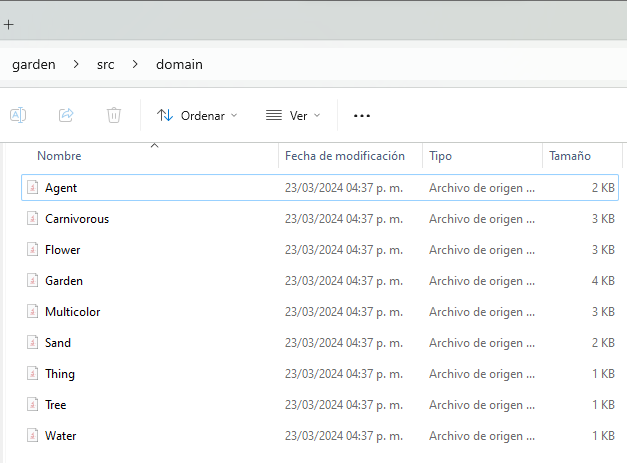




3. En el directorio copien únicamente los archivos \*.java del paquete de aplicación .

Consulte y capture el contenido de src/domain.





**Estructura de proyecto en java**

En java los proyectos se estructuran considerando tres directorios básicos.

garden

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.

src (Source Code):

La carpeta "src" debe contener todo el código fuente de tu proyecto Java. Los archivos en esta carpeta serán archivos Java (.java) que contienen las clases y el código fuente del proyecto.

La organización interna de "src" depende del proyecto y puede seguir la estructura de paquetes Java.

bin (Binary):

La carpeta "bin" generalmente se utiliza para almacenar los archivos binarios generados a partir del código fuente. Estos archivos binarios pueden incluir archivos de clases compiladas (.class) y archivos ejecutables (por ejemplo, archivos JAR en el caso de Java).

La organización interna de "bin" suele ser más simple y puede contener archivos como este:

bin/

miapp.jar

"miapp.jar" es un archivo ejecutable que contiene las clases compiladas de tu aplicación

docs (Documentation):

La carpeta "docs" se utiliza para almacenar documentación relacionada con el proyecto. Esto puede incluir documentación técnica, documentación de usuario, diagramas, manuales y cualquier otro tipo de documentación relevante.

La organización interna de "docs" puede variar según las necesidades del proyecto, pero a menudo se utiliza una estructura de carpetas para categorizar diferentes tipos de documentación. Por ejemplo:

docs/

tecnica/

arquitectura.md

diagramas/

diagrama\_flujo.png

usuario/

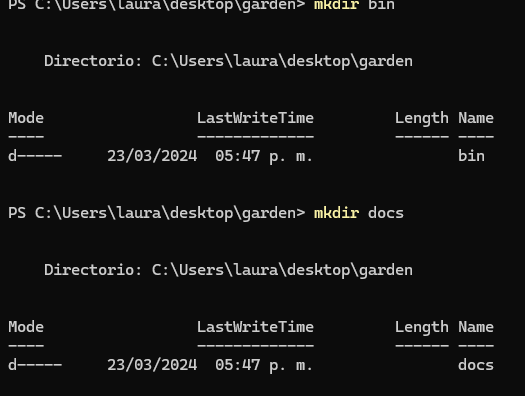
manual\_usuario.pdf

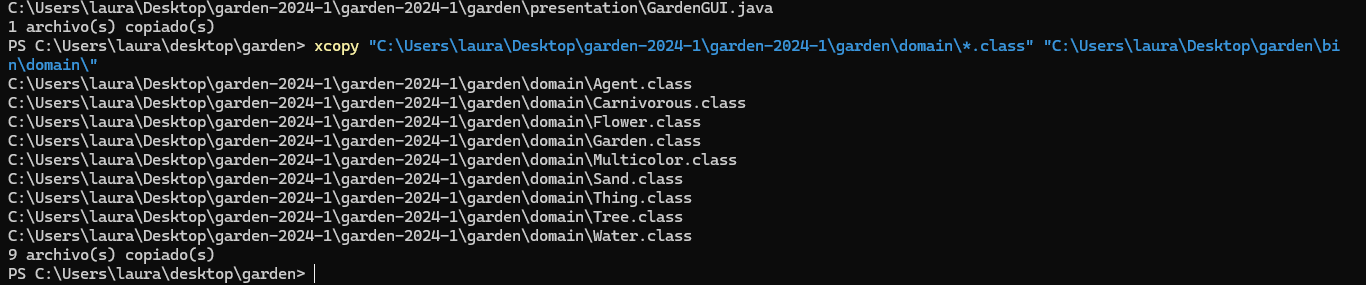
changelog.md

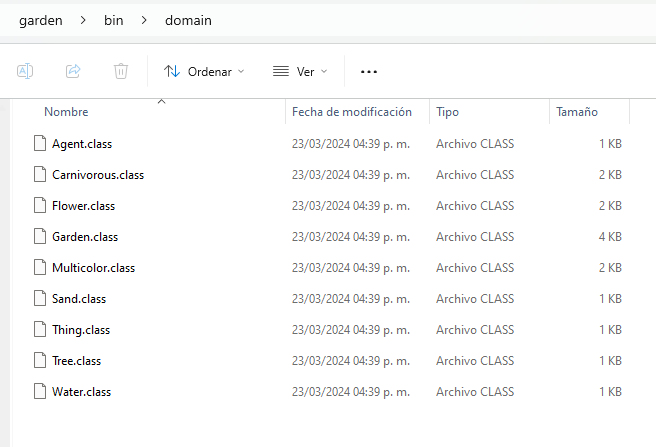
2. ¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué?

Cópielos y consulte y capture el contenido del directorio que modificó.

La JVM solo necesita los archivos compilados (.class) para ejecutar el programa, ya que estos contienen el código ejecutable en formato bytecode.



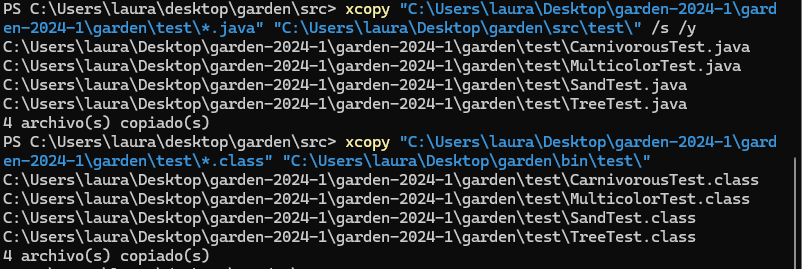






Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Comandos de java**

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

**javac:** javac es el compilador de Java. Se utiliza para traducir el código fuente escrito en Java (archivos con extensión .java) en archivos de clase bytecode (archivos con extensión .class) que pueden ser ejecutados en la máquina virtual Java (JVM).

Ejemplo de uso: javac MiClase.java compilará un archivo de código fuente Java llamado MiClase.java en un archivo de clase MiClase.class.

**java:** java es el intérprete de Java y se utiliza para ejecutar aplicaciones Java. Toma el archivo de clase bytecode generado por javac y lo ejecuta en la JVM.

Ejemplo de uso: java MiClase ejecutará la clase MiClase previamente compilada.

**javadoc:** javadoc es una herramienta que genera documentación a partir de comentarios en el código fuente Java. Los comentarios deben estar formateados de acuerdo con las convenciones de Javadoc. La herramienta procesa los comentarios y genera documentación en formato HTML.

Ejemplo de uso: javadoc MiClase.java generará documentación para la clase MiClase basada en los comentarios del código fuente.

**jar**: jar (Java Archive) es una herramienta que se utiliza para crear, manipular y administrar archivos JAR, que son archivos comprimidos que pueden contener clases Java, recursos y otros archivos. Los archivos JAR son comunes para distribuir bibliotecas y aplicaciones Java.

Ejemplo de uso:

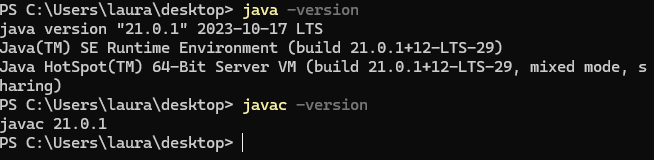
jar -cf MiArchivo.jar MiClase.class creará un archivo JAR llamado MiArchivo.jar que contendrá la clase MiClase.

jar -xf MiArchivo.jar descomprimirá un archivo JAR para acceder a sus contenidos.

2. Cree una sesión de consola y consulte en línea las opciones de los comandos java y javac. Capture las pallas.

3. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.

2.3



La primera solo muestra información detallada sobre la versión de Java.

La segunda solo muestra el número de versión.

**Compilando**

1. Utilizando el comando javac, desde el directorio raiz (desde garden 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.

1. Con las siguientes instrucciones obtenemos que :

javac -d ../bin domain/Water.java

javac -d ../bin domain/Garden.java

javac -d ../bin domain/Flower.java

javac -d ../bin domain/Sand.java

javac -d ../bin domain/Tree.java

javac -d ../bin domain/Multicolor.java

javac -d ../bin domain/Carnivorous.java

javac -d ../bin domain/Thing.java

javac -d ../bin domain/Agent.java

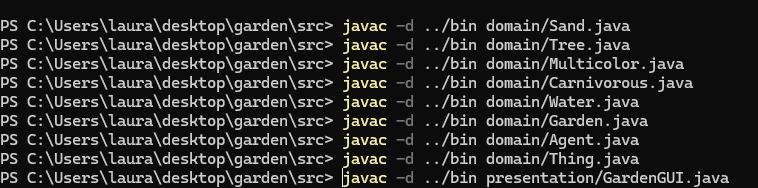
javac -d ../bin presentation/GardenGUI.java

javac -d ../bin test/MulticolorTest.java

javac -d ../bin test/CarnivorousTest.java

javac -d ../bin test/SandTest.java

javac -d ../bin test/TreeTest.java



2. Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles nuevos archivos aparecen ahora y dónde se ubican?

Aparecen 2 nuevos directorios domain y presentation las cuales cada uno tienen archivos de tipo .class, estas se ubican en el directorio bin.

**Documentando**

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?

javadoc -d ./ -sourcepath src -subpackages paquete.principal

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

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

Para navegar por la documentación generada en formato html, se debe abrir el archivo llamado index.

Interfaz de usuario gráfica

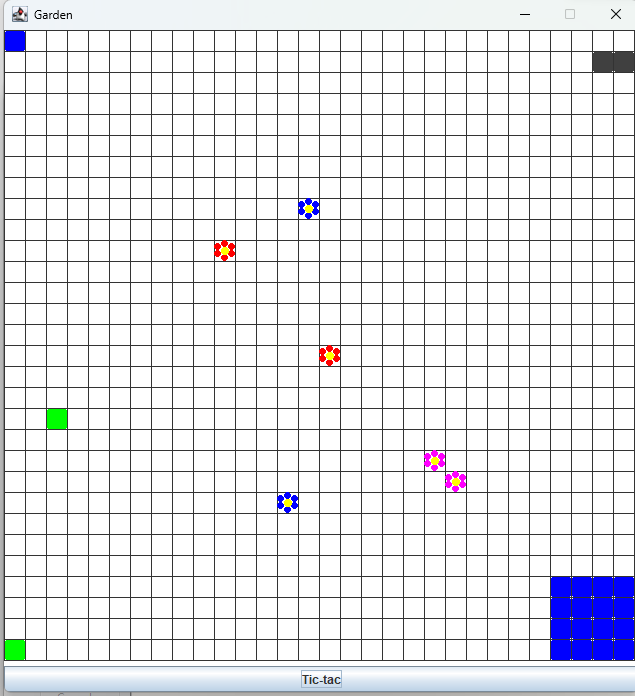
Descripción generada automáticamente con confianza media

**Ejecutando**

4. Empleando el comando java, desde el directorio raiz, ejecute el programa. ¿Cómo

utilizó este comando?





**Probando**

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?

Para incluir un paquete externo, como JUnit 4.8 en este caso, al compilar un programa en Java, se necesita agregar la librería JUnit al classpath durante el proceso de compilación. Esto se puede hacer utilizando la opción -cp o -classpath seguida de la ruta hacia el archivo JAR de JUnit.



2. Ejecute desde consola las pruebas . ¿Cómo utilizó este comando?. Puede ver ejemplos de ­­cómo ejecutar el “test runner” en How to run JUnit test cases from the command line

java -cp .:/usr/share/java/junit.jar org.junit.runner.JUnitCore [test class name]

­­­

**Empaquetando**

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 ? Para empaquetar un archivo jar, vamos a seguir el siguiente ejemplo:

```

mi\_programa/

clases/

MiClase.class

recursos/

archivo.txt

``

a. Abrir la consola el directorio que contiene la carpeta `mi\_programa`.

b. Utiliza el comando jar que se encuentra en la carpeta bin del directorio JDK para crear un archivo JAR.

- ruta\_al\_jdk :es la ruta al directorio del JDK que contiene la carpeta bin. Esto puede variar según el sistema operativo y cómo esté configurado el JDK.

- cvf: indica que estás creando un nuevo archivo jar

- nombre\_del\_archivo.jar: nombre del archivo a pasar a jar

--C directorio: indica que se debe cambiar al directorio especificado antes de agregar los archivos.

- El último punto `.` indica que se deben incluir todos los archivos y subdirectorios en el directorio actual.

```

ruta\_al\_jdk\bin\jar cvf nombre\_del\_archivo.jar -C directorio .

```

2. ¿Cómo se ejecuta el proyecto empaquetado?

Una vez que se haya empaquetado el proyecto en un archivo JAR, se ejecuta en la consola de comandos utilizando el comando java.

Siguiendo el ejemplo anterior:

a. Navegar hasta el directorio donde se encuentra el archivo JAR. Para hacerlo, se utiliza el comando cd para cambiar de directorio.

```

cd ruta/al/directorio/donde/esta/el/mi\_programa.jar

```

b. Ejecutar el proyecto empaquetado en el archivo JAR utilizando el siguiente comando:

```

java -jar mi\_programa.jar

```

c. Presiona "Enter" y Java ejecutará la aplicación desde e

l archivo JAR.

**RETROSPECTIVA**

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

Cada uno trabajo 15 horas

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

Nos falta realizar un punto de consola debido a que no ejecutaba en la consola

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

Son importantes para trabajar eficientemente, intentado emplear el menor tiempo posible

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

Nuestro mayor logro fue poder comunicarnos en la asignación de tareas, y nuestro problema fue optimizar y entender bien los puntos del laboratorio. Para resolverlo, fue hablar detenidamente de los ejercicios.

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

Hicimos bien como equipo la comunicación y el desarrollo de los puntos, y nos comprometemos a mejorar el rendimiento del tiempo.