**Laboratorio 03**

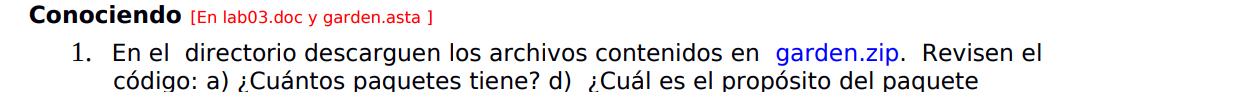
**Integrantes:**

**Sebastián Cardona Parra**

**Diego Alexander Cárdenas Beltrán**

**Universidad Escuela Colombiana de Ingeniería Julio Garavito**  
**Programación Orientada a Objetos**  
 **María Irma Rozo**

**15/03/2024**



1. Tiene 2 paquetes: presentation y Domain.
2. Mostrar visualmente la interfaz grafica para que el usuario pueda usar la aplicación, y no tenga que interactuar directamente con el paquete Dominio.
3. este ya tiene toda la lógica y sus funcionamientos.



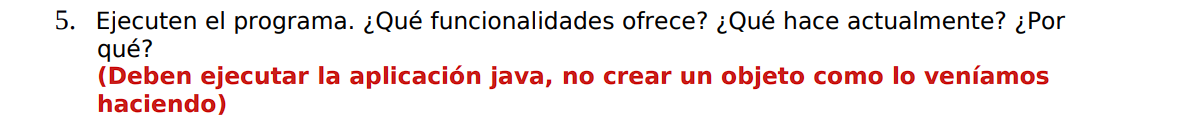
1. Tiene una interfaz, una clase abstracta y tres clases completas
2. Las clases abstractas implican que son hereditarios de otras clases concretas donde sus métodos abstractos se van a implementar en estas, las interfaces son implementadas por clases concretas es decir estas implementan todos sus métodos en otras palabras, un método en una clases abstracta o interfaz puede tener diferentes implementaciones en distintas clases concretas.

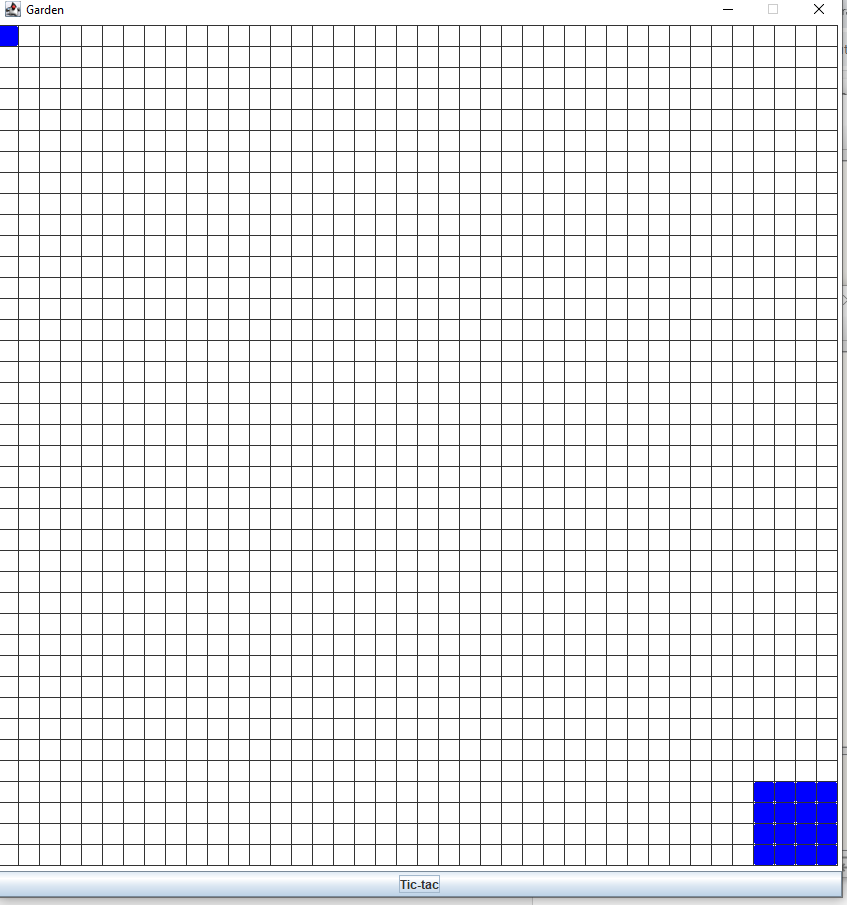


1. Tiene un archivo que contiene dos clases concretas.
2. Tiene 4 métodos públicos propios no heredados

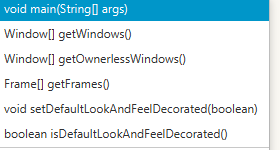


Se debe ejecutar el método main que está dentro del archivo GardenGUI

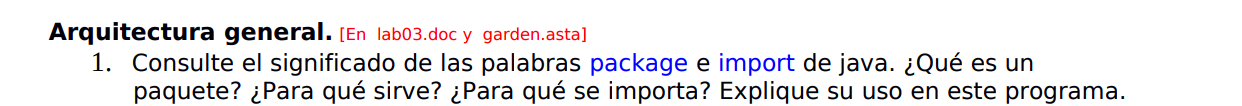




Actualmente ofrece las siguientes funcionabilidades:



Ofrece un botón llamado tic-tac, pero este aun no hace nada pues no está programado



Package es un modo de contener diversas clases concretas, abstractas o interfaces, que se usan como un solo archivo, al principio de cada archivo se debe poner la palabra clave **package** nombre\_paquete\_al\_que\_pertenece

Import es la manera para referenciar y usar otros paquetes distintos al propio se usa la palabra clave **Import** nombre\_paquete\_a\_usar



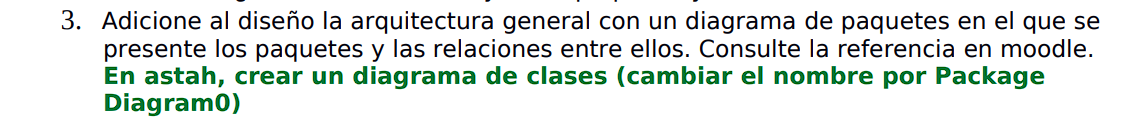
En el directorio Garden se encuentra todos los subdirectorios de paquetes.

Además, está el directorio doc, donde se encuentra todo el html de las documentaciones del proyecto o aplicación

La coincidencia entre los paquetes y directorios es que los paquetes se tratan como directorios, es decir ambos pueden agrupar distintas coas, ente las cuales archivos

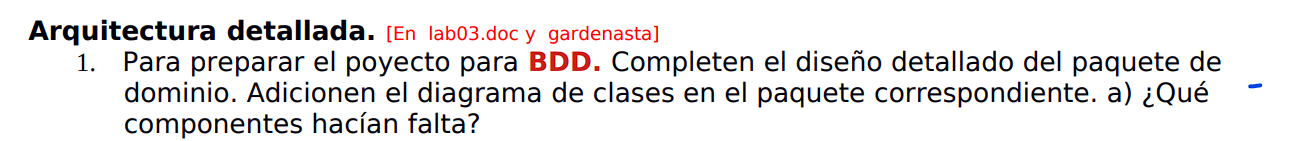
El paquete domain contiene contiene los .class que son los compilados los tipo java que son codigos y el .ctxt que es directamente usado por BlueJ, así mismo tiene el arvhico BlueJ donde se ve contenido todo lo anterior que és la lógica e la aplicación

El paquete presentation contiene los mismos tipos de archivos del anterior paquete, pero de los archivos que se encargan de trabajar como interfaz gráfica para el usuario



Gráfico, Gráfico en cascada

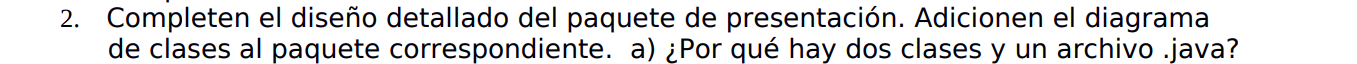
Descripción generada automáticamente



Hacía falta las clases concretas Flower y Water

Diagrama

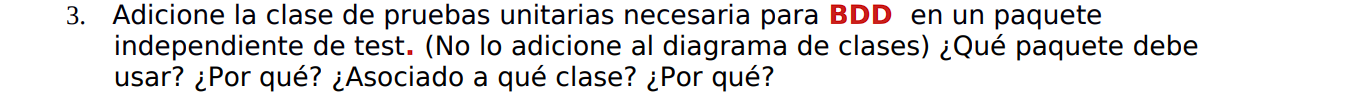
Descripción generada automáticamente



Diagrama

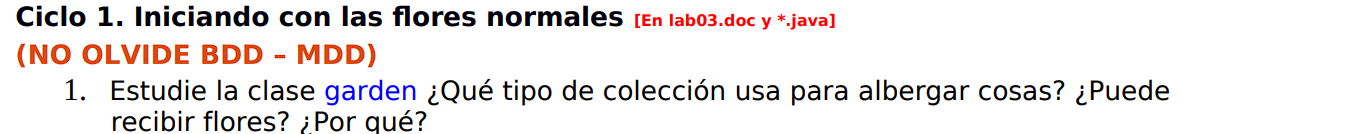
Descripción generada automáticamente

Debido a que estas das clases solo pueden interactuar entre ellas, uno es publica y la otra es privada esto hace que solo interactúen entre sí.

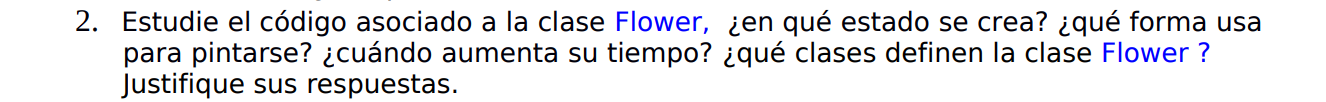
Se debe usar el paquete de domain pues en este está la lógica de la aplicación y a partir de esta es que se va a hacer la interfaz gráfica, esta prueba, debe ir relacionado con la clase de Garden, pues sobre esta se hará la interfaz y es esta clase la principal, sobre esto, si probamos esta clase, las demás clases estarán bien

Diagrama

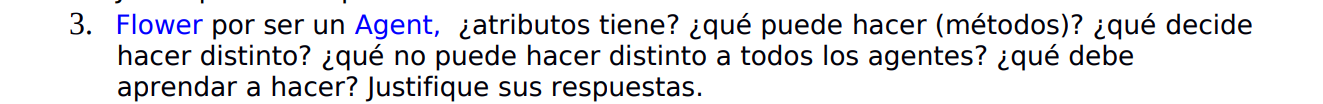
Descripción generada automáticamente



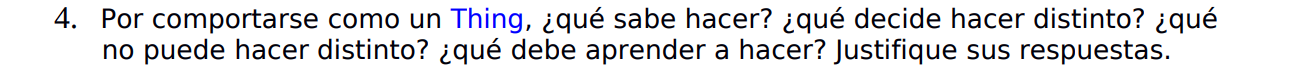
Se usa un arreglo de arreglos o en otras palabras una matriz, en ella se encuentran las cosas, además si se puede revivir flores, ya que, pese a que las flores pertenecen a una clase distinta, implementan Thing, es decir, se comportan como tales y se añaden con el método setThing



Una Flower se crea inicialmente en estado ALIVE, para pintarse usa el paquete presentation, específicamente la clase GardenGUI, y usa círculos para pintarse. Cada vez que la clase realiza una acción, aumenta su tiempo con el método turn de la clase Agent, Flower extiende la clase abstracta Agent e implementa la interfaz Thing, son estas las que definen a Flower



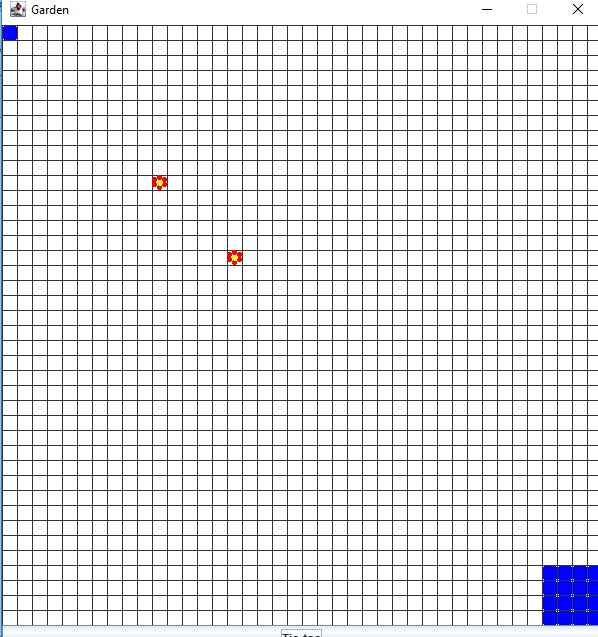
Flower tiene los atributos de state, time, puede hacer turn, que es aumentar su tiempo, puede moverse move, puede obtener el tiempo actual de esta getTime, puede establecer su tiempo setTime, y puede preguntarse si esta viva isAlive. Lo que puede y debe aprender a hacer distinto a los demás agentes es move, moverse de diferente manera, lo que no puede hacer distinto a los demás agentes son los demás métodos pertenecientes a Agent, lo que puede hacer distinto son los métodos de Thing, además de obtener el garden en donde está, validar si una posición es legal, calcular la distancia entre dos puntos

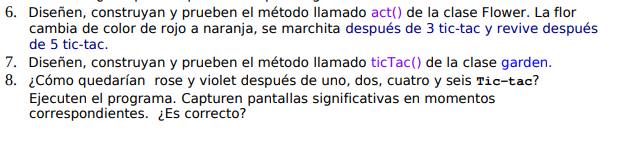


Una Flower por comportarse como Thing sabe actuar, sabe obtener su columna, su fila donde está, sabe establecer su color, sabe establecer su forma, y sabe preguntarse si está viva, lo que puede hacer distinto son los métodos propios de Agent, además de obtener el garden en donde está, validar si una posición es legal, calcular la distancia entre dos puntos, debe aprender a hacer todos los métodos de Thing, pues aún no están implementados a excepción de getColor, Shape e is que tienen valores e implementaciones por defecto.

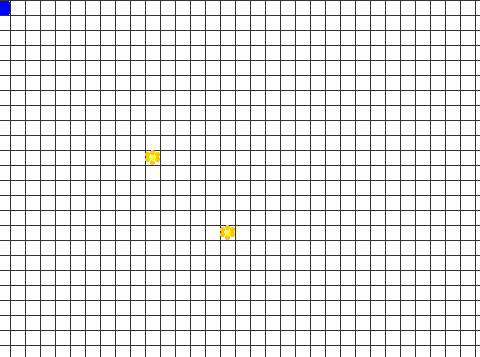


Al ejecutar el programa, nos percatamos que se crean las dos flores en el tablero en las posiciones específicadas

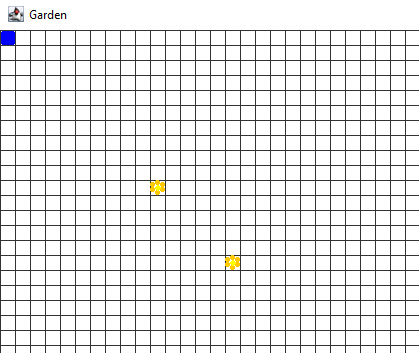




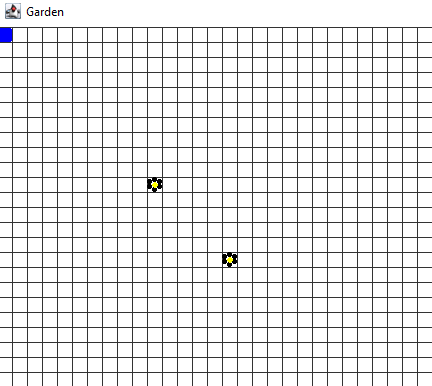
1 tictac:



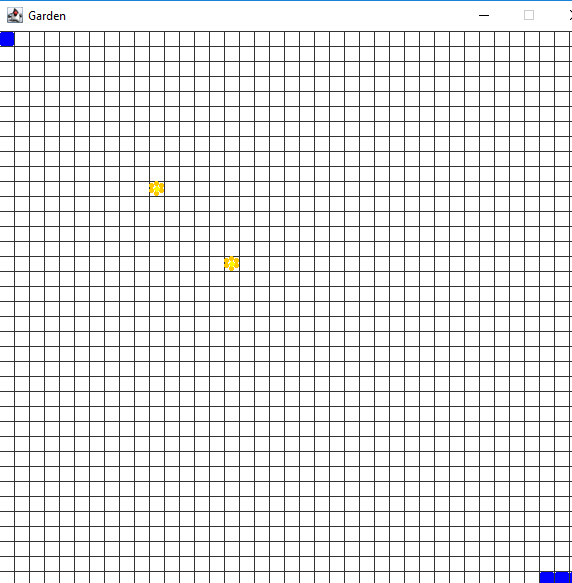
2 tictac:

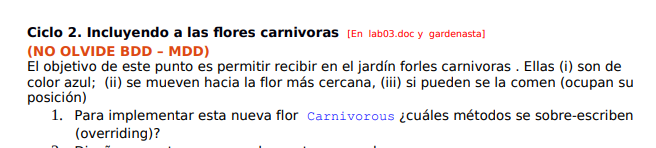


4 tictac:

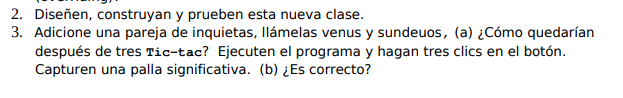


6 tictac:

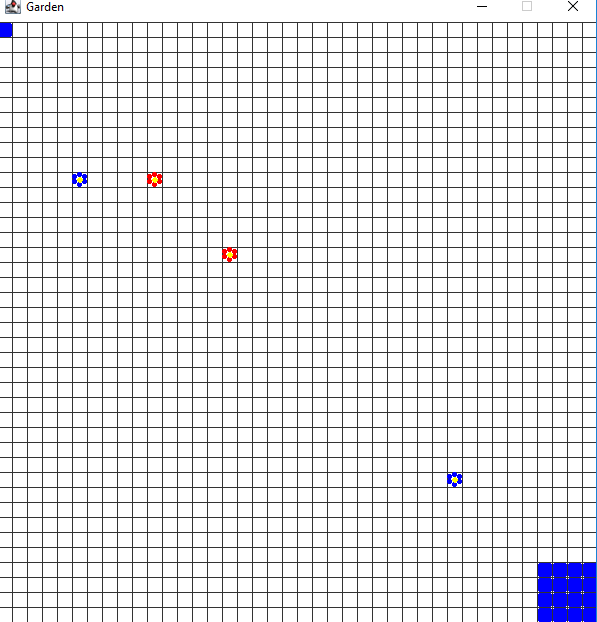




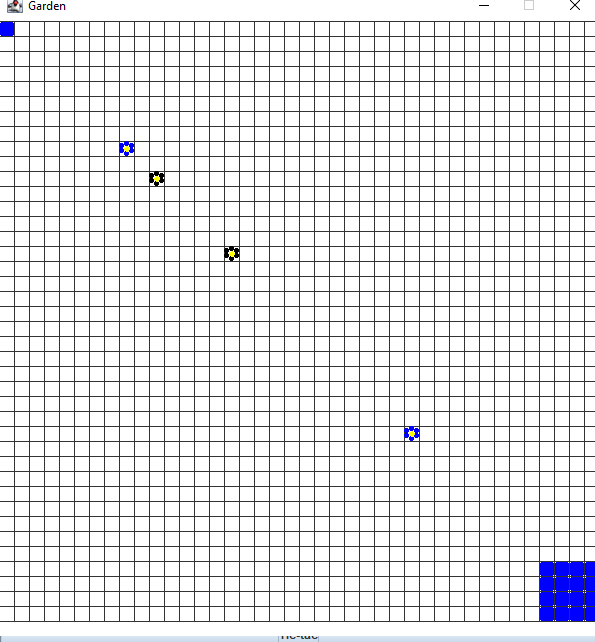
El método que tuvimos que sobre-escribir fue act, ya que el act de una flor normal no es el mismo al de una carnívora

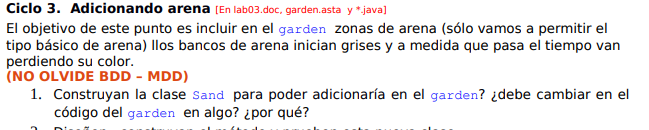


Inicio:

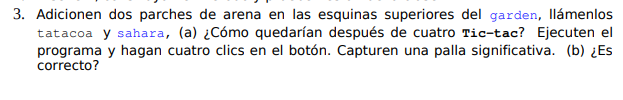


Fin:

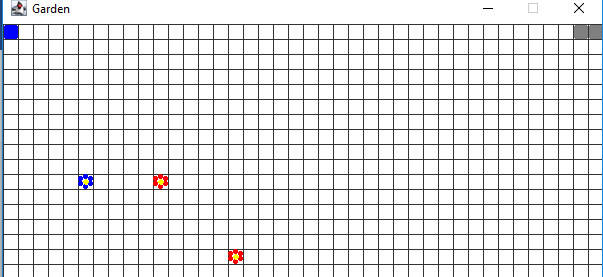




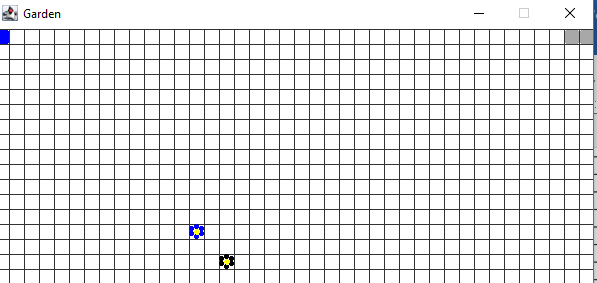
No debemos cambiar nada en Garden, debido a que lo que toca modificar es el act de sand y este no lo define garden

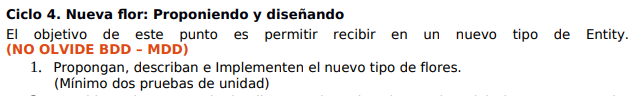


1 tictac:

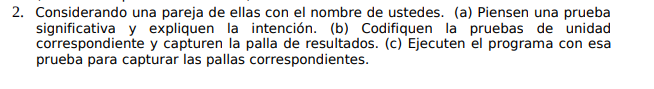


4tictac:



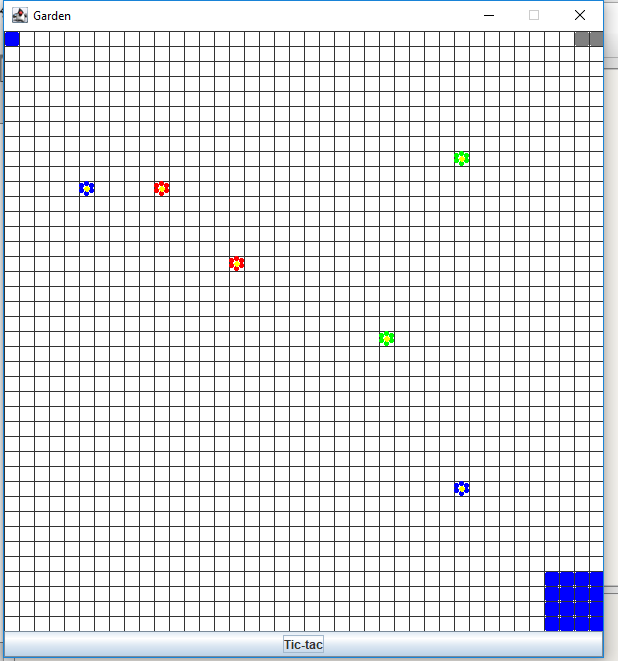


Nuestra nueva flor se llama Cane, es una flor que se dirige a el banco de arena más cercano e impide que siga perdiendo su color, es decir, una vez que el banco de arena tiene por lo menos una flor Cane como vecina, este deja de perder color, sino que lo recupera, es decir, Cane nutre la arena

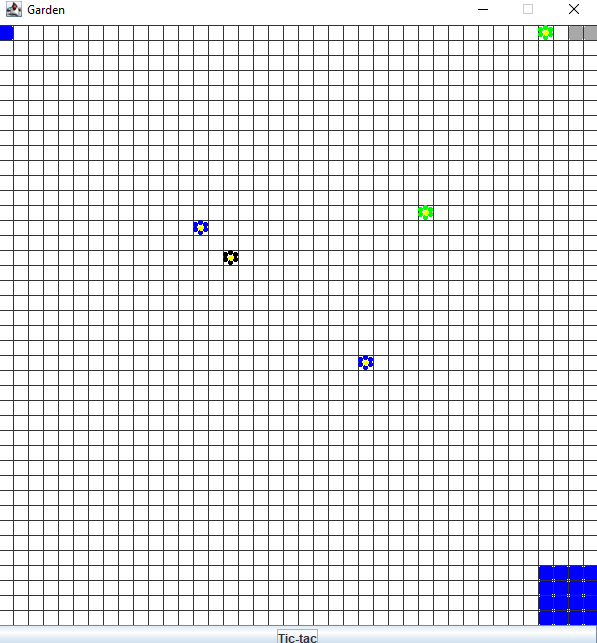


Nuestra prueba se tratará de crear las dos nuevas flores lejos de las arenas, al hacer varios tictacs, las arenas pierden su color, pero una vez las flores Cane llegan a la arena, esta se vuelve a nutrir, eso hasta que las carnívoras se la coman, así, nuevamente las arenas vuelven a perder color.

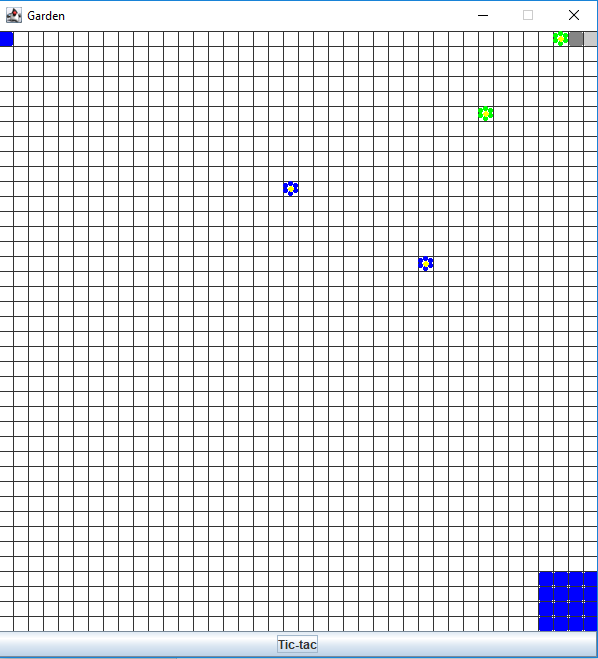
Estado inicial, las flores Cane son las verdes



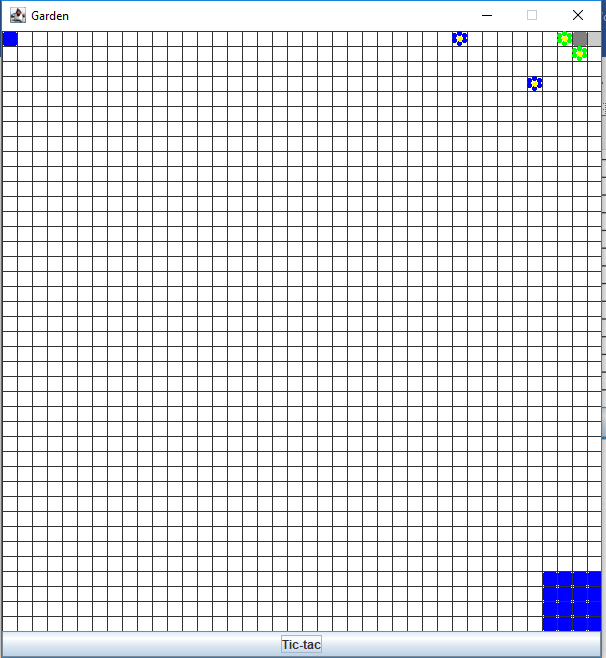
Estado conde una flor Cane está a punto de llegar donde está la arena, nótese que la arena ah perdido color:



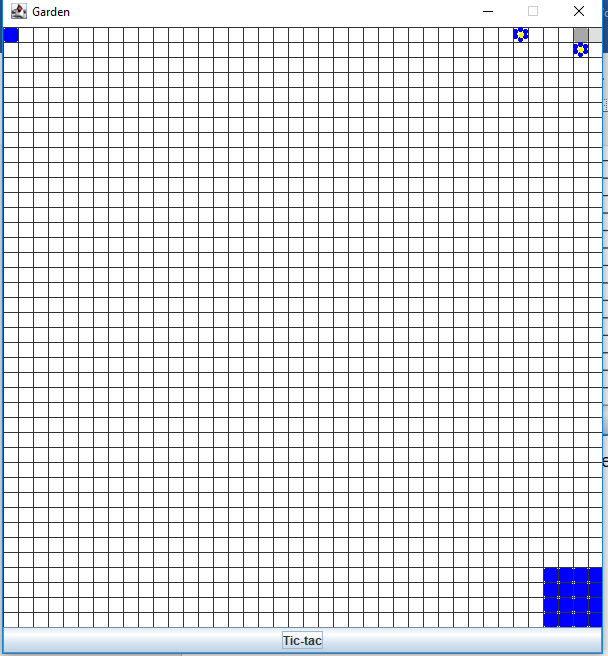
Estado donde la flor Cane ha estado nutriendo a la arena adyacente durante un rato, nótese que la arena que tiene a la flor Cane como vecina, vuelve a ganar color mientras que la otra sigue perdiendo

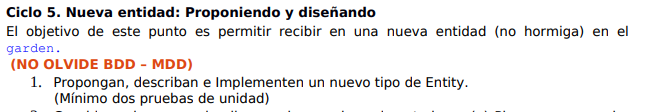


Estado donde ambas flores llegan y nutren ambas arenas

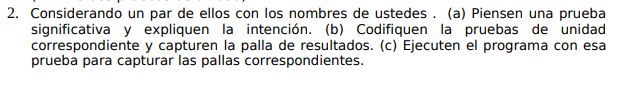


Estado donde las carnívoras se comen las flores y las arenas nuevamente pierden color



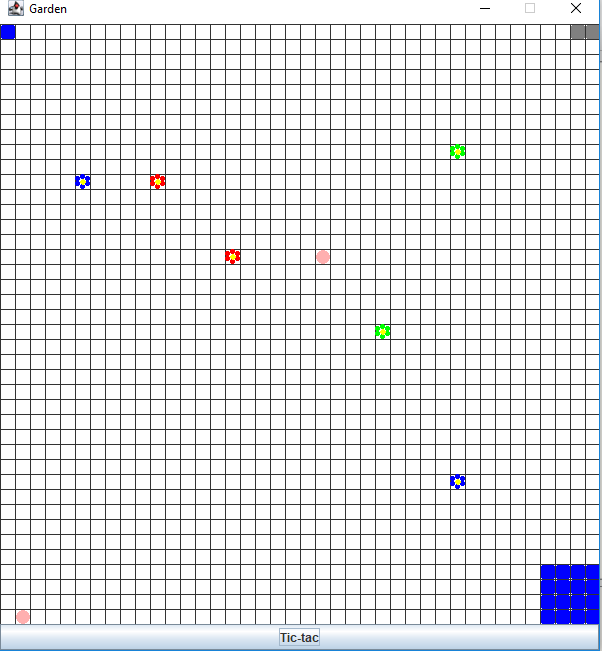


Creamos un nuevo Thing que es un animal (en forma de ovalo y rosado) que se encarga de seguir y comer las plantas carnívoras

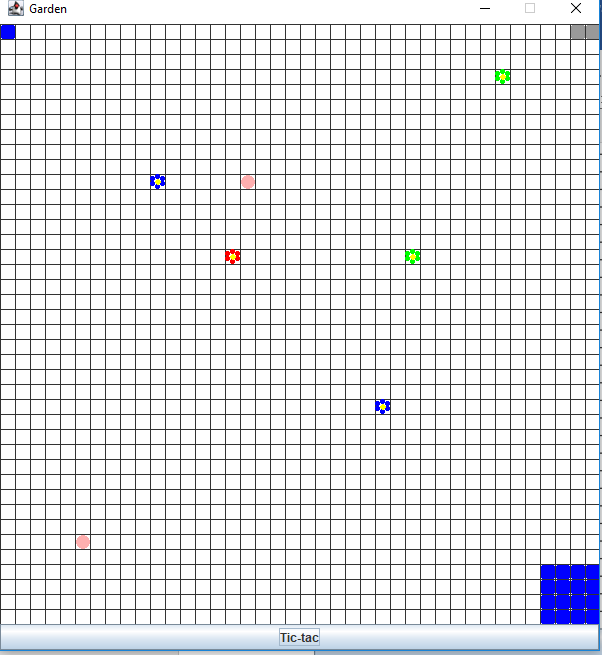


Pondremos dos hormigas en cierta posición y observaremos como se encargan de seguir y comerse solo las flores carnívoras, ignora las demás

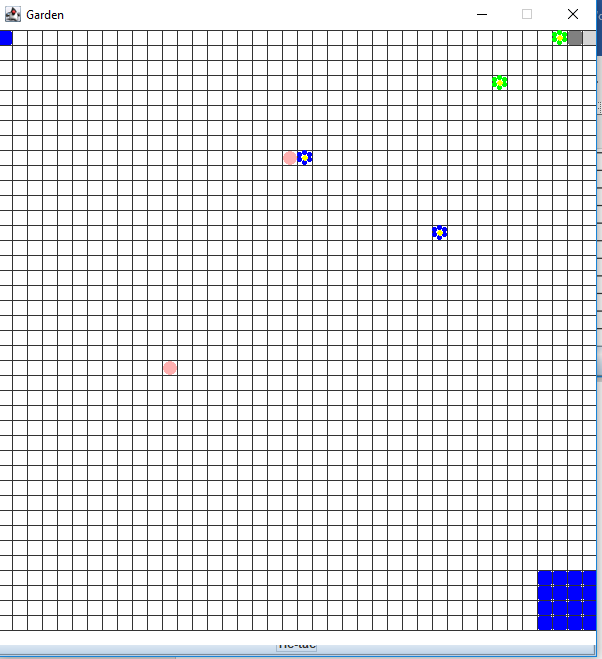
Estado inicial, las rosadas son las hormigas



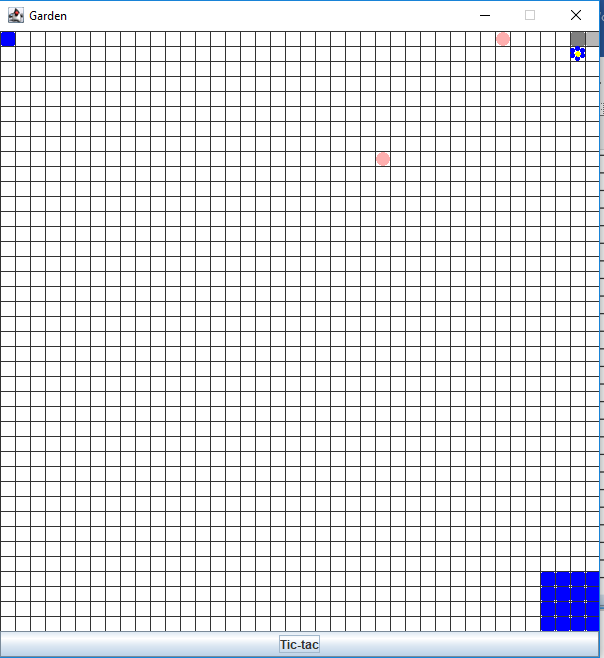
Estado donde vemos que las hormigas se acercan solamente a la carnívora más cercana



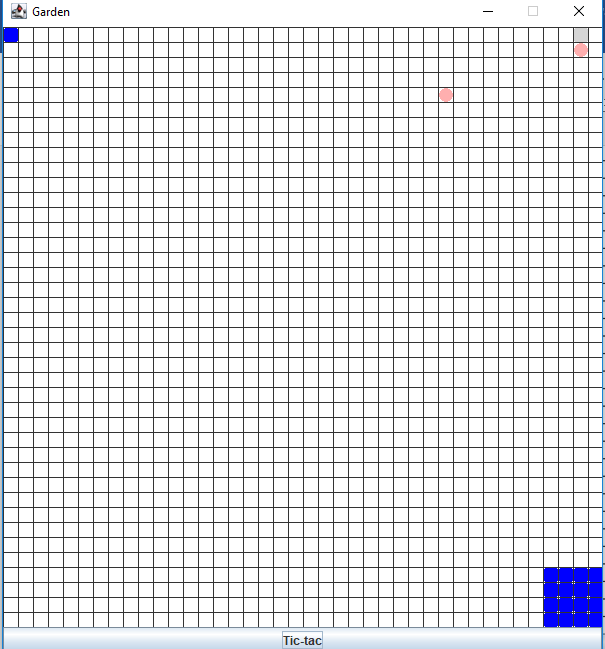
Estado donde la primera hormiga está a punto de comerse la carnívora

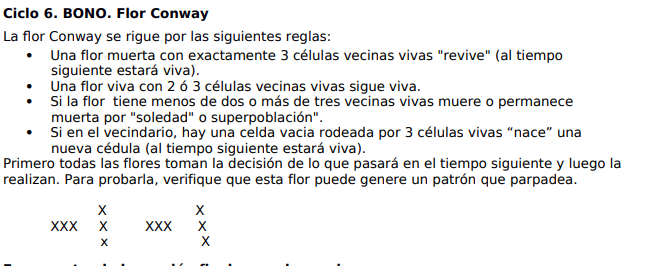


Estado donde la carnívora restante mató las flores restantes, pero las hormigas van tras ella

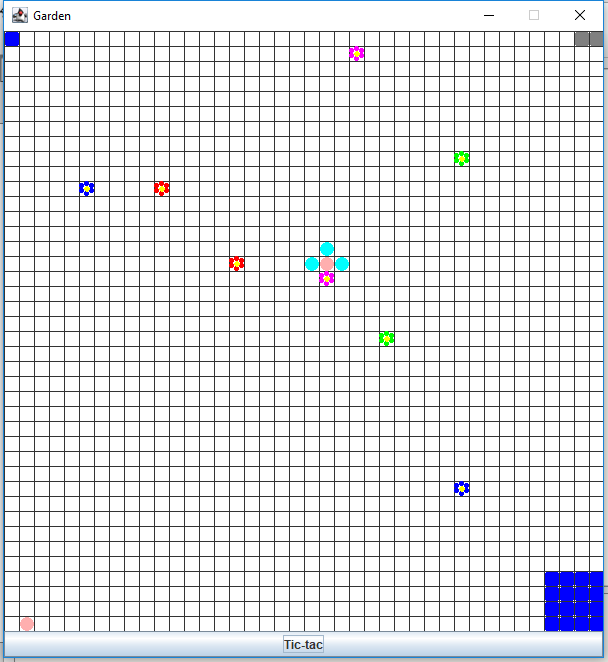


Finalmente, las hormigas se comen a todas las carnívoras

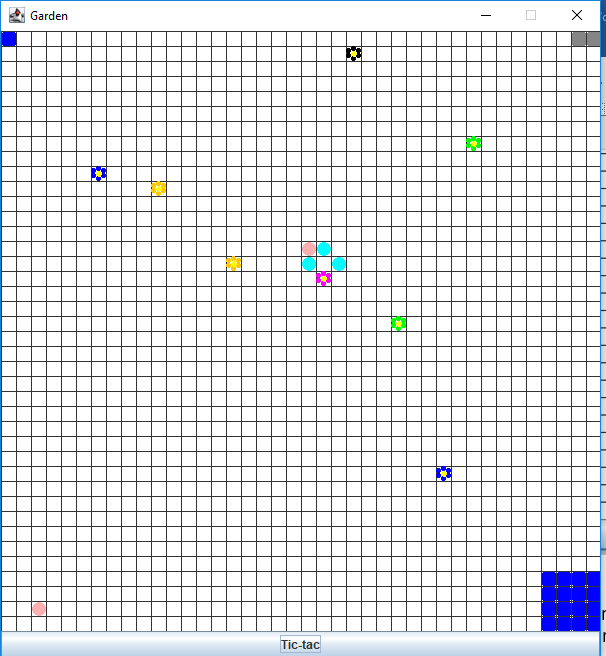




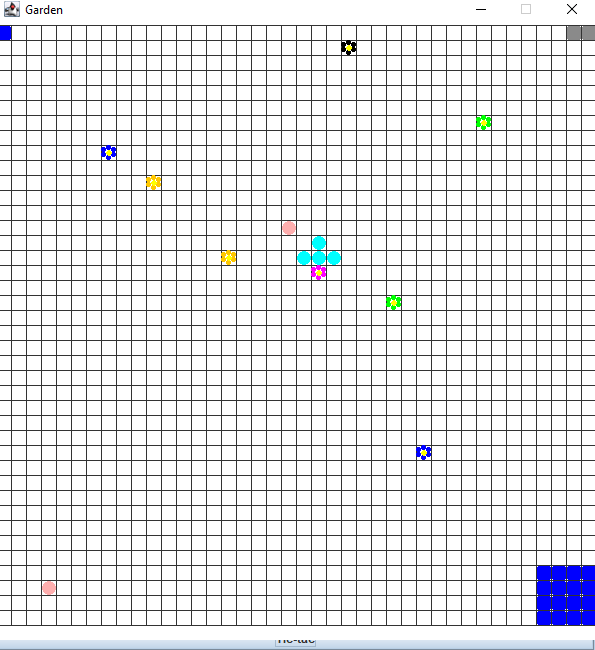
Las Células son los círculos celestes, las Conway son las magenta:

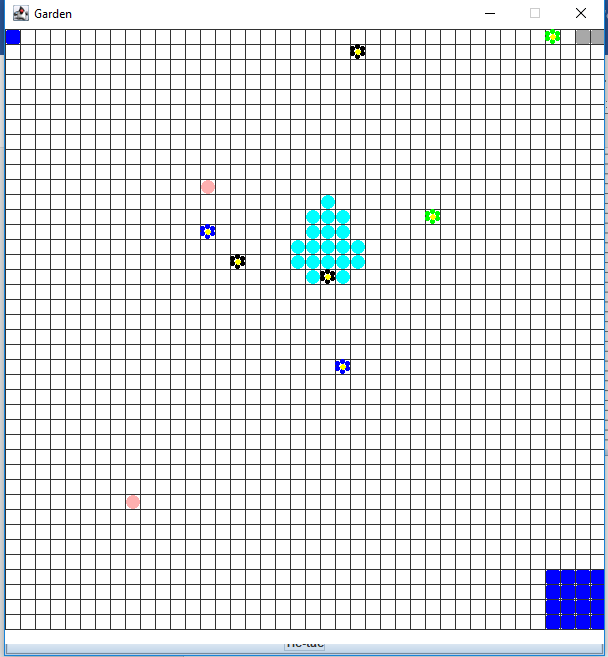


La magenta de abajo inicialmente tiene dos células vivas vecinas, por lo que sigue viva, pero la magenta de arriba no tiene ninguna, es decir, muere

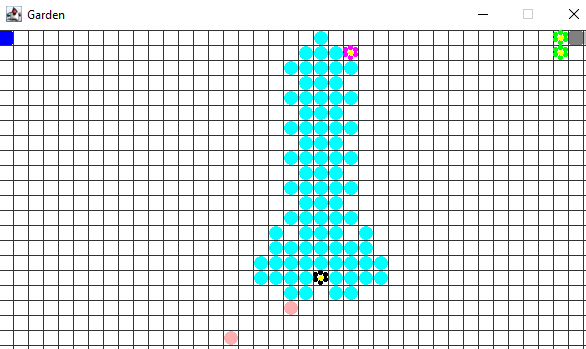


Las células vivas empiezan a esparcirse



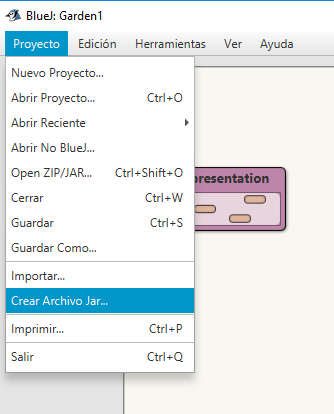


Llego un momento donde la Conway de abajo tenia más de tres células vivas, por lo cual muere



La conway de arriba ya tiene tres células vecinas vivas, es decir, revive





Selecciona esta opción y selecciona la clase GardenGUI



Ve hasta el directorio donde se encuentra el archivo .jar

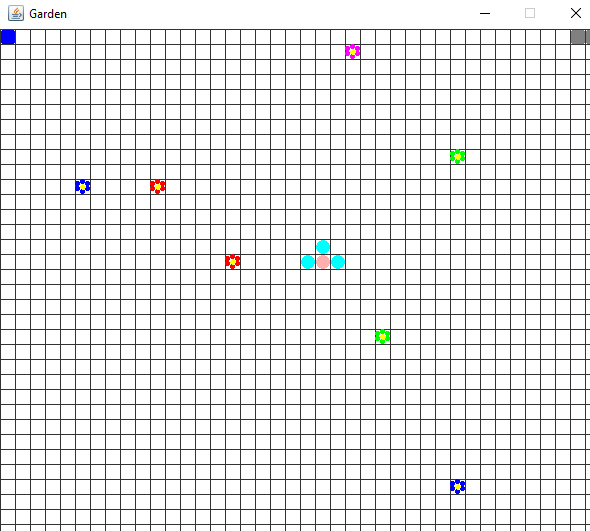
Ejecuta:

java -jar nombre\_del\_archivo.jar

sino debes especificar la ruta donde se encuentra el archivo

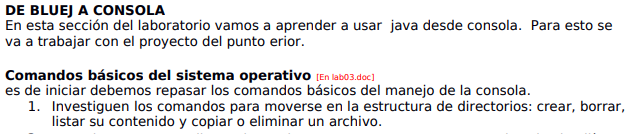
java -jar C:\ruta\_completa\archivo.jar

Cuando se ejecuta se abre el tablero

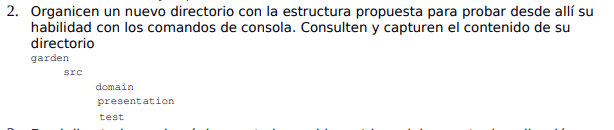




* + Los archivos JAR encapsulan todo el programa, lo que facilita su ejecución a través de diferentes sistemas operativos y entornos.
  + Un archivo JAR es fácil de compartir, simplificando la entrega del proyecto a usuarios finales o colaboradores y haciendo más comprensible su ejecución a la hora de la presentación
  + Permite que la aplicación se ejecute sin recompilar el código fuente
  + Se pueden empaquetar también imágenes y archivos dentro del .JAR, facilitando su transporte y que este no sea dependiente de su entorno donde se ejecute
  + Las bibliotecas pueden incluirse en el archivo JAR, gestionando la necesidad de que los usuarios instalen bibliotecas por separado.
  + Se posee una facilidad a la hora de actualizar la aplicación, ya que solo basta con reemplazar el archivo viejo JAR por el nuevo
  + Tienen bastante utilidad en los desarrollos web, ya que estos archivos son esenciales para empaquetar aplicaciones o bibliotecas



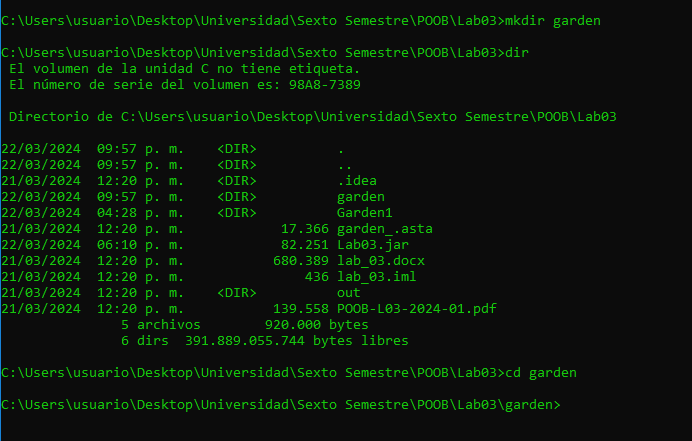
* + cd “directorio”: Permite cambiar al directorio especificado.
  + cd .. : Retrocede un nivel en la estructura de directorios.
  + cd /: Navega al directorio raíz del sistema.
  + mkdir nombre\_directorio: Crea un nuevo directorio.
  + rmdir nombre\_directorio: Elimina un directorio vacío.
  + rm -r nombre\_directorio: Elimina un directorio y su contenido
  + touch nombre\_archivo: Crea un archivo vacío.
  + rm nombre\_archivo: Elimina un archivo.
  + dir : listar los archivos y directorios del directorio actual
  + dir /s : lo mismo de dir, también lista los subdirectorios del directorio actual
  + copy “origen” “destino”: copia un archivo desde el origen hasta el destino
  + move “origen” “destino”: mueve archivo o directorio



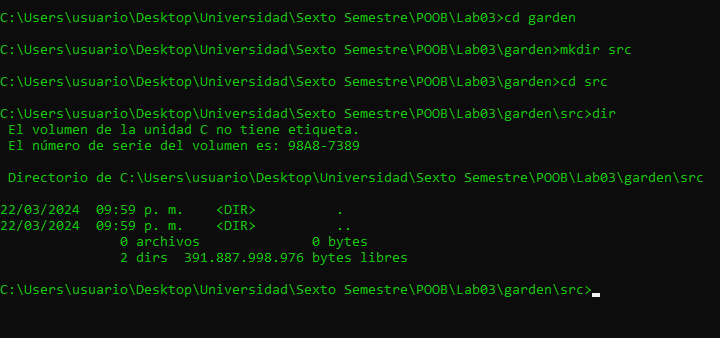
Se hará esta bitácora suponiendo el computador de Sebastián Cardona, cuya ubicación donde estará el directorio es:

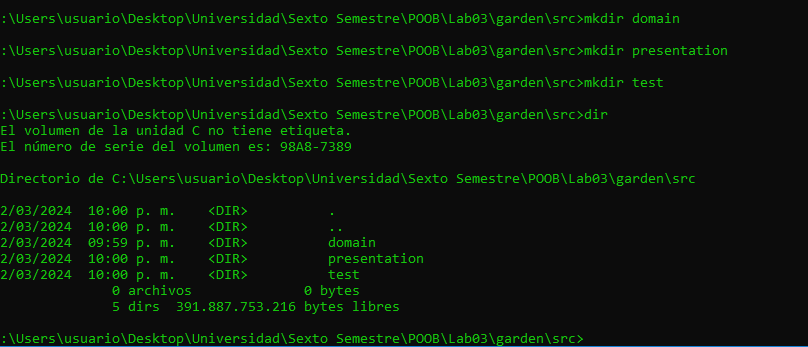
C:\Users\usuario\Desktop\Universidad\Sexto Semestre\POOB\Lab03

Creamos el directorio en la ubicación especificada garden y listamos para ver si se creó, posteriormente, entramos por medio del cd garden a dicho directorio:

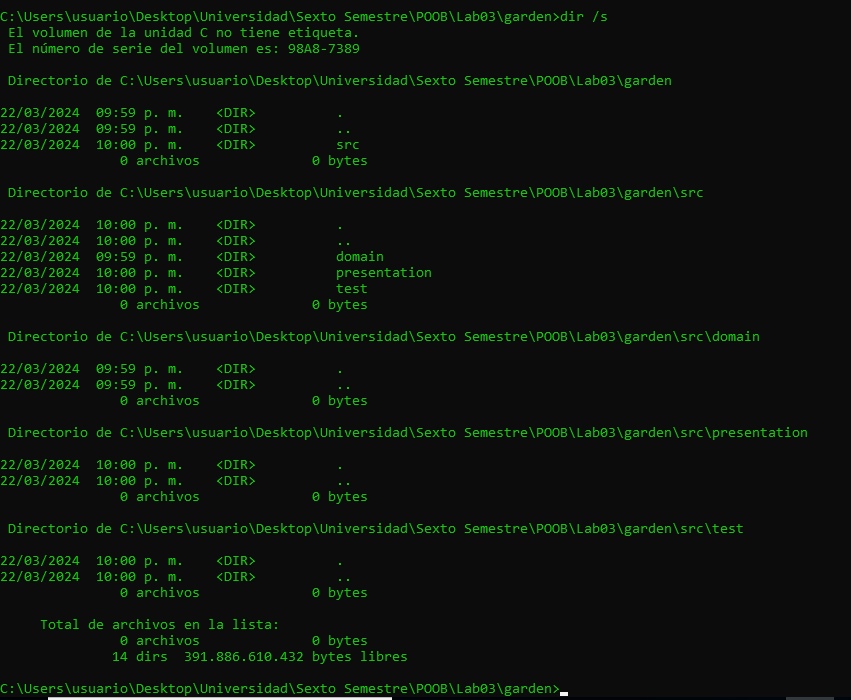


Creamos las demás carpetas y subdirectorios





Si listamos todo con dir /s

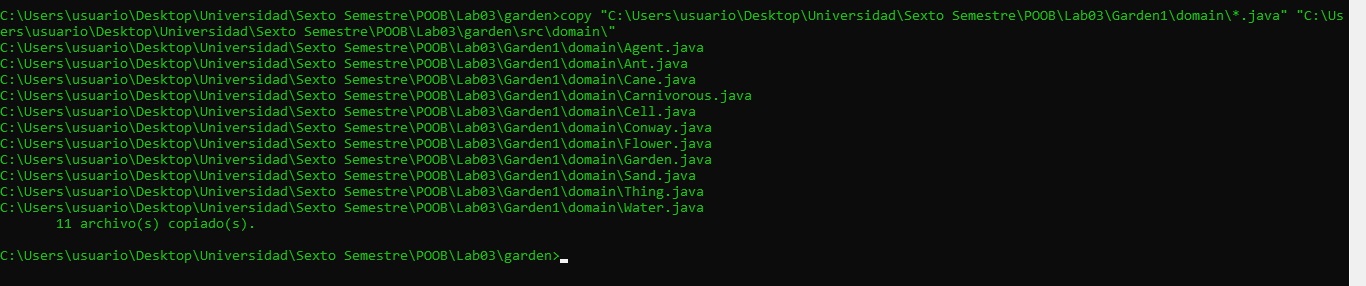


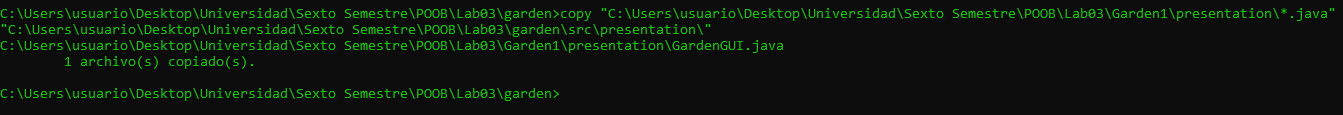


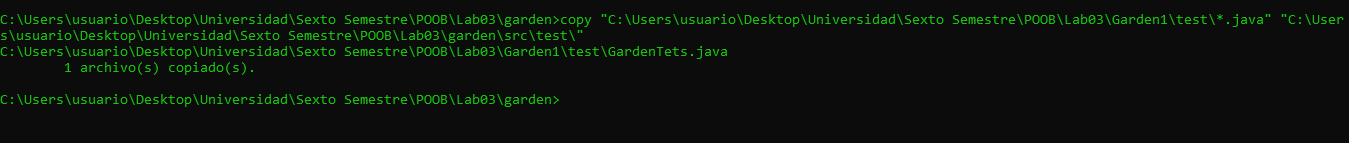
Usaremos:

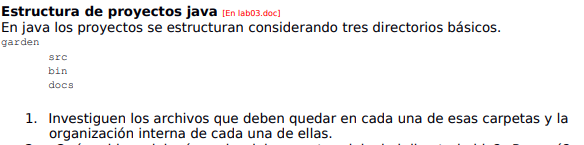
copy "C:\Users\usuario\Desktop\Universidad\Sexto Semestre\POOB\Lab03\Garden1\domain\\*.java" "C:\Users\usuario\Desktop\Universidad\Sexto Semestre\POOB\Lab03\garden\src\domain\"

Para copiar todo del directorio original domain hasta el nuevo directorio domain que creamos, repetimos con presentation y test









src: allí se deben albergar todos los archivos .java, es decir, el código fuente

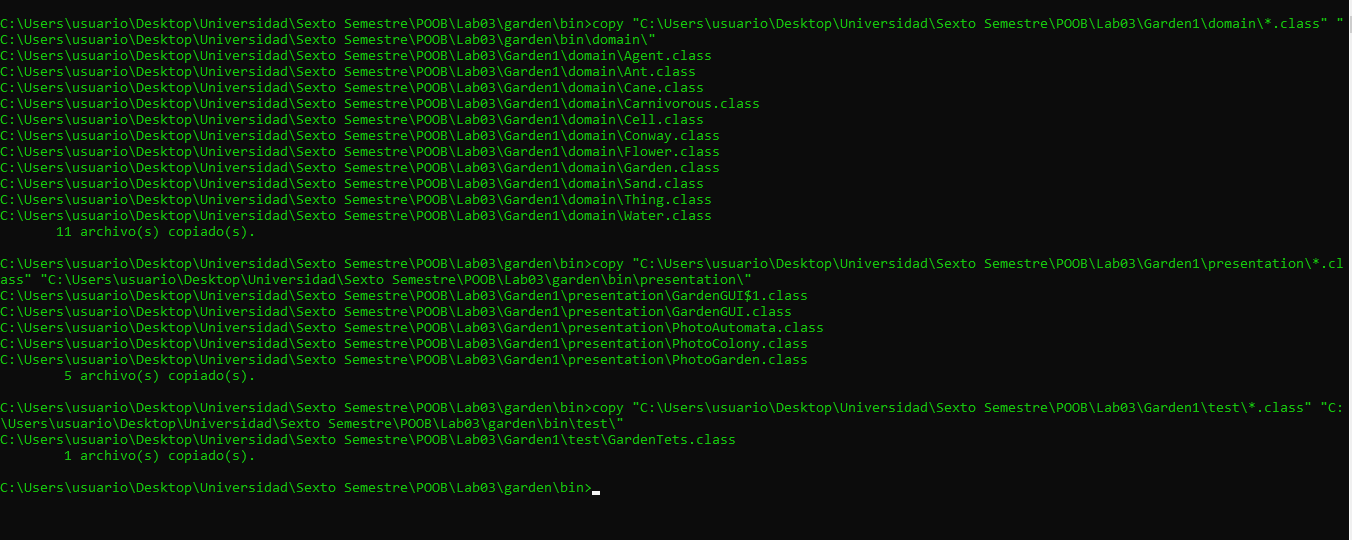
bin: Allí se deben almacenar los códigos compilados, es decir los archivos de tipo .class

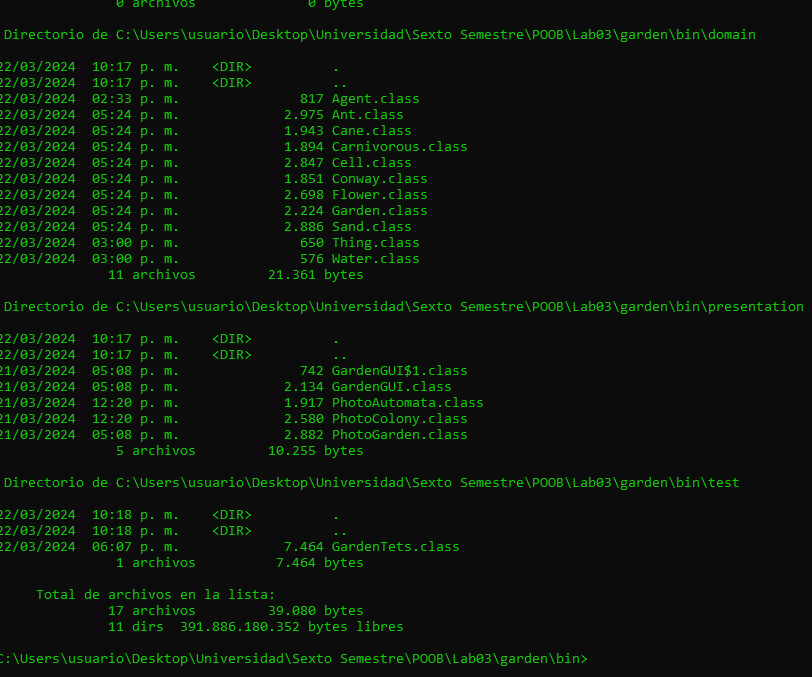
docs: Allí se alberga la documentación del código fuente como archivos .html

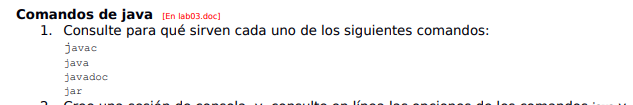


Debo copiar todos los archivos .class del directorio original

Para hacerlo se hace algo similar al punto anterior de copiar:







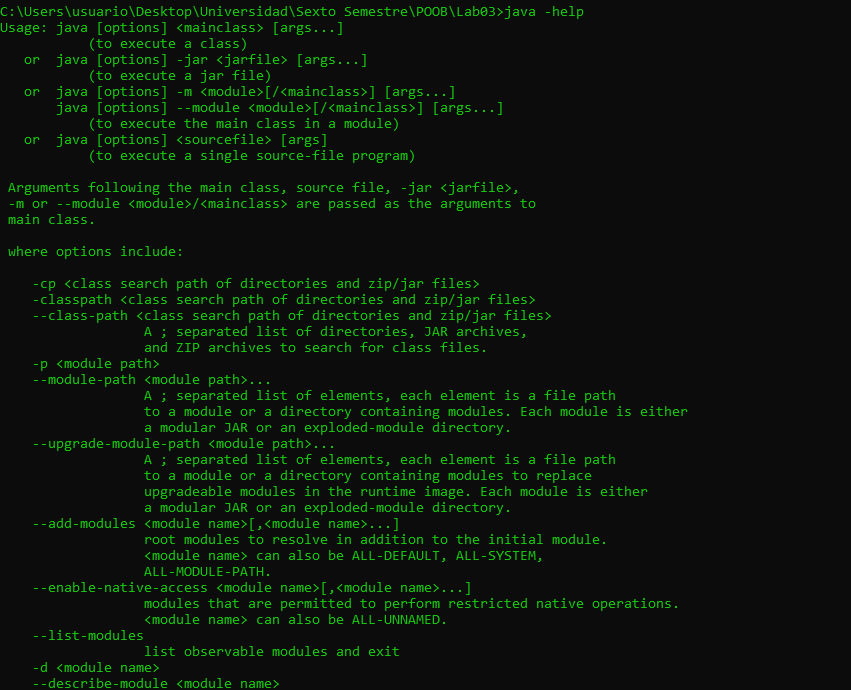
javac: Compilar archivos .java y genera el archivo .class

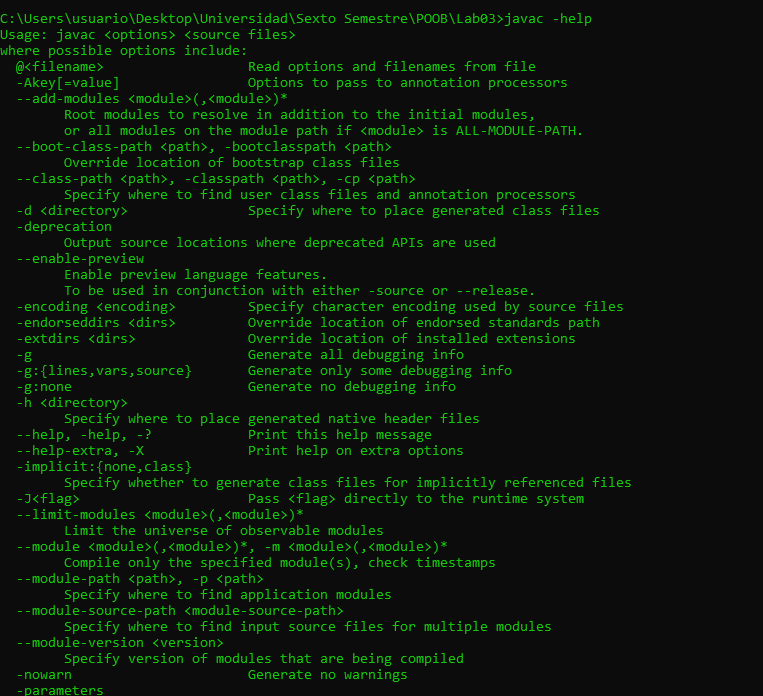
java: Ejecutar programas compilados (.class)

javadoc: Genera la documentación del código a partir de comentarios existentes en los archivos .java

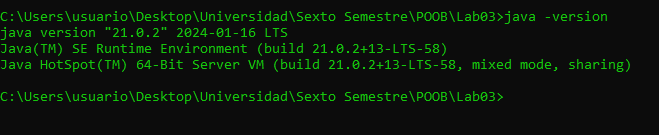
jar: crear y manipular archivos .jar que son archivos comprimidos y contienen múltiples archivos compilados y recursos

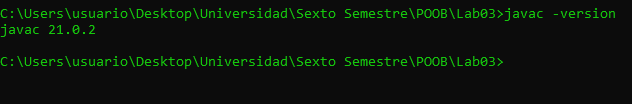


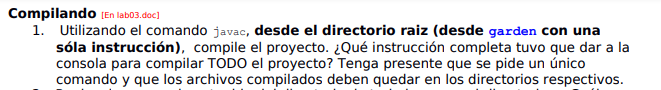






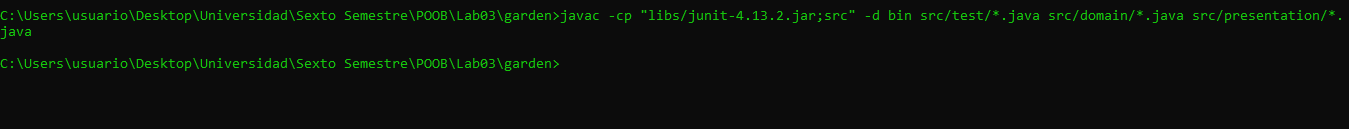




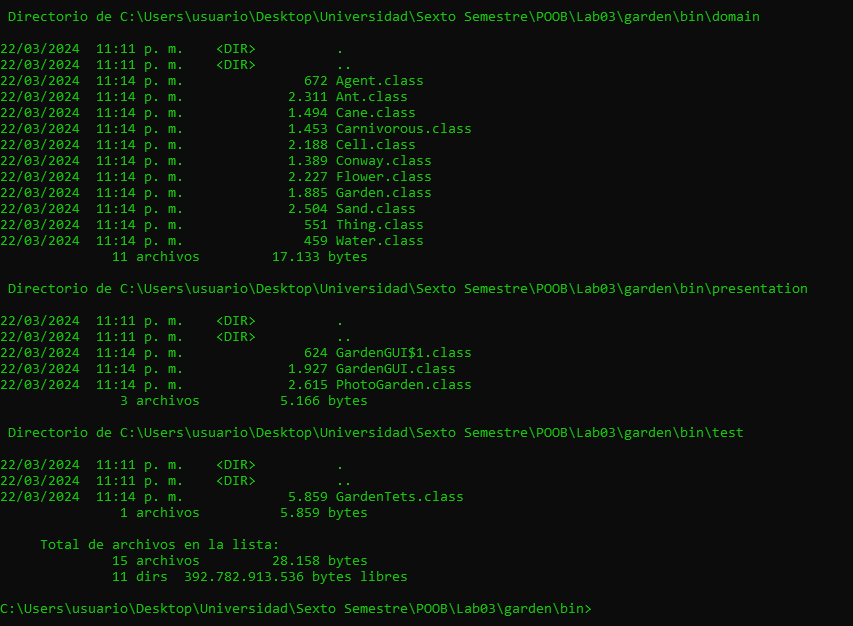


Toca tener cuidado al compilar las pruebas, ya que aquí se usa el frameWork de Junit, por lo que es importante descargar y poner en un directorio llamado libs dentro de garden el archivo .jar llamado junit-4.13.2.jar. de esta manera se usa el siguiente comado para compilar el proyecto desde garden

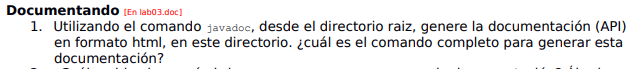
javac -cp "libs/junit-4.13.2.jar;src" -d bin src/test/\*.java src/domain/\*.java src/presentation/\*.java



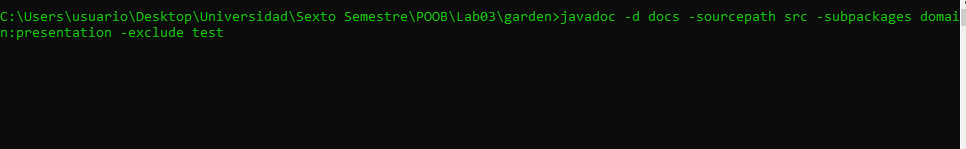


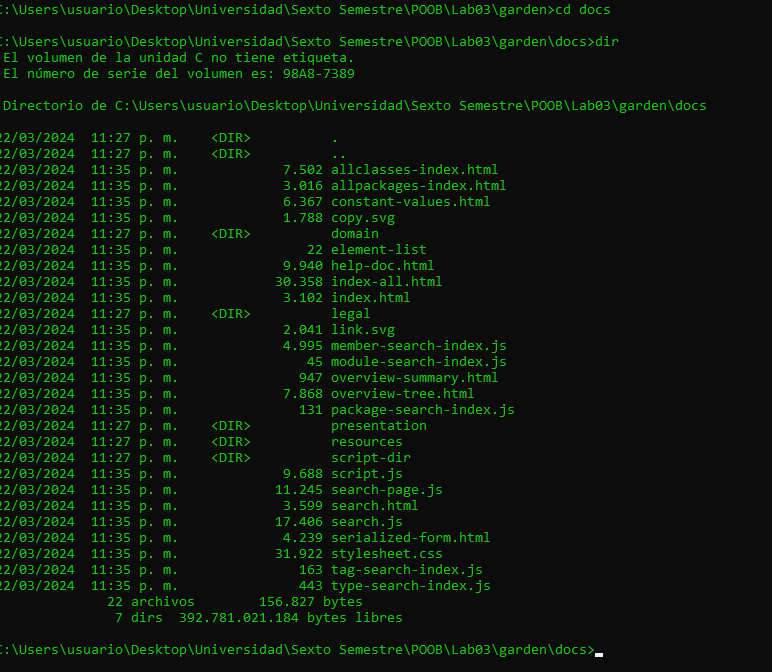


Se ubican dentro de bin



javadoc -d docs -sourcepath src -subpackages domain:presentation -exclude test



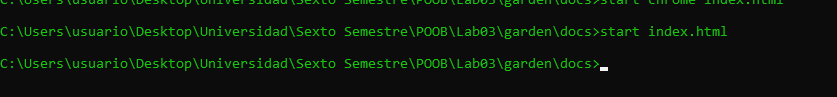


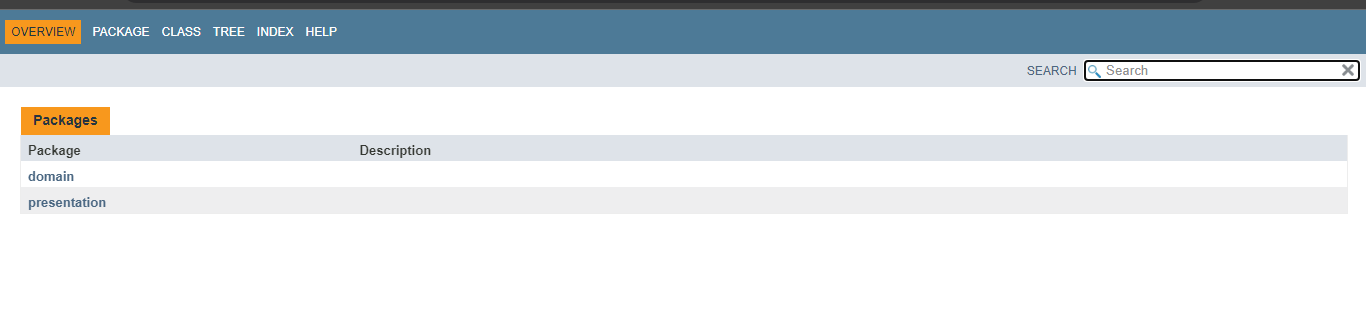


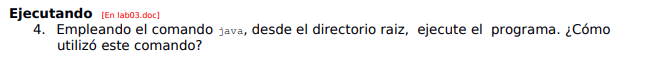
El archive que debemos abrir, es index.html

Nos ubicamos en el directorio docs:

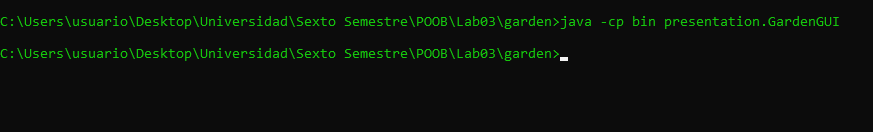
start index.html

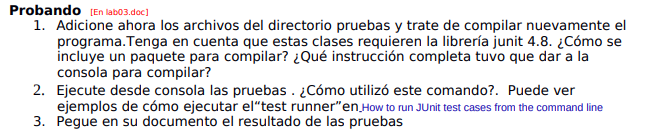






java -cp bin presentation.GardenGUI





1. ya está hecho: se dio el comando:

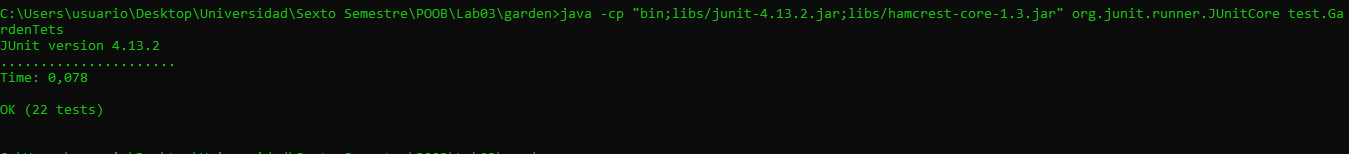
javac -cp "libs/junit-4.13.2.jar;src" -d bin src/test/\*.java src/domain/\*.java src/presentation/\*.java

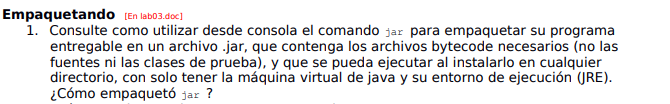
1. En este caso debemos hacer algo similar para ejecutar las pruebas, debemos instalar y poner en el directorio libs el archivo hamcrest-core-1.3.jar

Se usa el siguiente comando:

java -cp "bin;libs/junit-4.13.2.jar;libs/hamcrest-core-1.3.jar" org.junit.runner.JUnitCore test.GardenTets

1. Pantalla al ejecutar





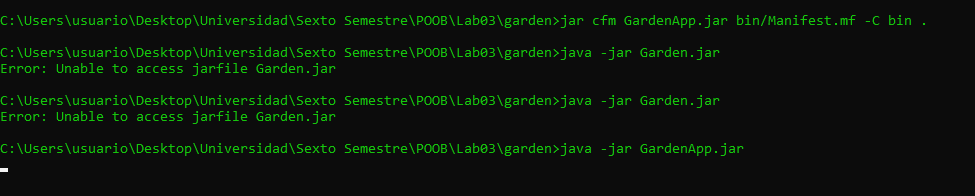
Primero creamos un archivo .mf en un editor de texto que contenga la siguiente linea:

Main-Class: presentation.GardenGUI

Guardala y colocala en el el directorio bin, esto es para al hacer el jar, indicar cual es la clase que debe ejecutar

Después ejecuta:

jar cfm GardenApp.jar bin/Manifest.mf -C bin .





Para ejecutar pon el siguiente comando

java -jar GardenApp.jar

**RETROSPECTIVA**

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

El tiempo total invertido a este laboratorio fue de 20 horas

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

Completado a finalidad, pues trabajamos hasta el último momento para dejar todo listo

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

Mejora la calidad del software: Las prácticas de XP, como la programación en parejas, las revisiones de código continuas y las pruebas unitarias, ayudan a mejorar la calidad del software al identificar y solucionar problemas temprano en el proceso de desarrollo.

Entrega de valor temprana: XP se centra en la entrega de software funcional en iteraciones cortas y frecuentes. Esto permite que los clientes obtengan valor rápidamente y proporciona oportunidades para retroalimentación temprana.

Flexibilidad y adaptabilidad: XP es una metodología ágil que se adapta a los cambios de requisitos. Las prácticas como la planificación de versiones frecuentes permiten ajustar el software a medida que evolucionan las necesidades del cliente.

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

Fue un laboratorio demasiado largo, el tiempo jugó a nuestra contra puesto que debíamos cumplir con otras obligaciones, pero nos encargamos de organizarnos desde el principio y hacer todas las dudas que nos surgían en los foros

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

Hicimos bien el trabajo a pares, y el desarrollo del contenido especificado para el código