**LABORATORIO Nro. 3 POOB**

**Camilo Murcia y Jeisson Casallas**

**Conociendo [En lab03.doc y colony.asta ]**

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

**a)** ¿Cuántos paquetes tiene?

Hay dos paquetes, Presentation y Domain.

**b)** ¿Cuál es el propósito del paquete presentación?

Es la que alberga la parte grafica de la simulación, en este caso la interfaz donde se ve reflejada la simulación de la colonia de hormigas.

**c)** ¿Cuál es el propósito del paquete dominio?

La clase dominio es la que alberga los objetos que va a contener la simulación, como lo son las hormigas, la comida, las colonias junto con sus clases abstractas e interfaces.

**2. Revisen el paquete de dominio**

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

Están los componentes de interface, clases abstractas y clases “normales” o concretas

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

Las clases concretas permiten crear objetos, se pueden usar todo tipo de atributos, y para sus métodos, también se pueden crear de cualquier tipo, siempre y cuando no sean abstractos.

Las clases abstractas no permite crear objetos, se pueden usar todo tipo de atributos, y para sus métodos, se pueden crear cualquier tipo, sin ninguna restricción.

Las interfaces no permiten crear objetos, sus variables solo pueden ser static, y de igual forma, sus métodos, se pueden crear sin ninguna restricción.

**3. Revisen el paquete de presentación,**

**a)** ¿Cuántos componentes tiene?

Solo tenemos una clase concreta la cual es ColonyGUI.

**b)** ¿Cuántos métodos ofrece?

Dentro de la clase ColonyGUI, tenemos 8 métodos, dividido en dos clases, ColonyGUI y PhotoColony

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

Para ejecutar el programa debemos compilar y ejecutar el método main(), y está en la clase ColonyGUI.

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

Al ejecutar el programa mediante el método de main(), vemos que hay una funcionalidad, que es tic tac, pero actualmente no hace ninguna acción

**Arquitectura general. [En lab03.doc y colonyAstah]**

**1. Consulte el significado de las palabras package e import de java.**

**¿Qué es un paquete?**

Un paquete en Java es un mecanismo de organización que agrupa un conjunto de clases y otros tipos de recursos relacionados en una estructura jerárquica. Los paquetes son directorios en el sistema de archivos que contienen archivos de código fuente (.java) y archivos compilados (.class).

**¿Para qué sirve?**

Los paquetes sirven para organizar y estructurar el código fuente de un programa Java en un sistema modular. Esto ayuda a evitar conflictos de nombres entre clases, facilita la reutilización de código y hace que el código sea más mantenible y comprensible. También proporcionan control de acceso, ya que las clases en un paquete pueden ser públicas (accesibles desde cualquier parte del código) o tener un acceso más restringido (paquete o privado).

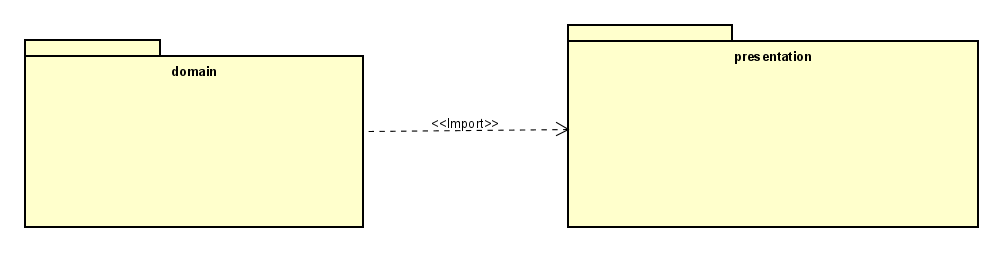
**¿Para qué se importa? Explique su uso en este programa.**

Se importan clases o miembros de otros paquetes para simplificar la escritura del código y mejorar la legibilidad. En el caso de este programa, se Importa el paquete de Domain a Presentation para que el programa pueda ser ejecutado desde la clase ColonyGUI.

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

El directorio de Colony está contenido por los Subdirectorios domain y presentation, los cuales vienen siendo los paquetes, que a su vez son directorios pues contienen también los archivos que permiten la ejecución de estos paquetes implementados en BlueJ

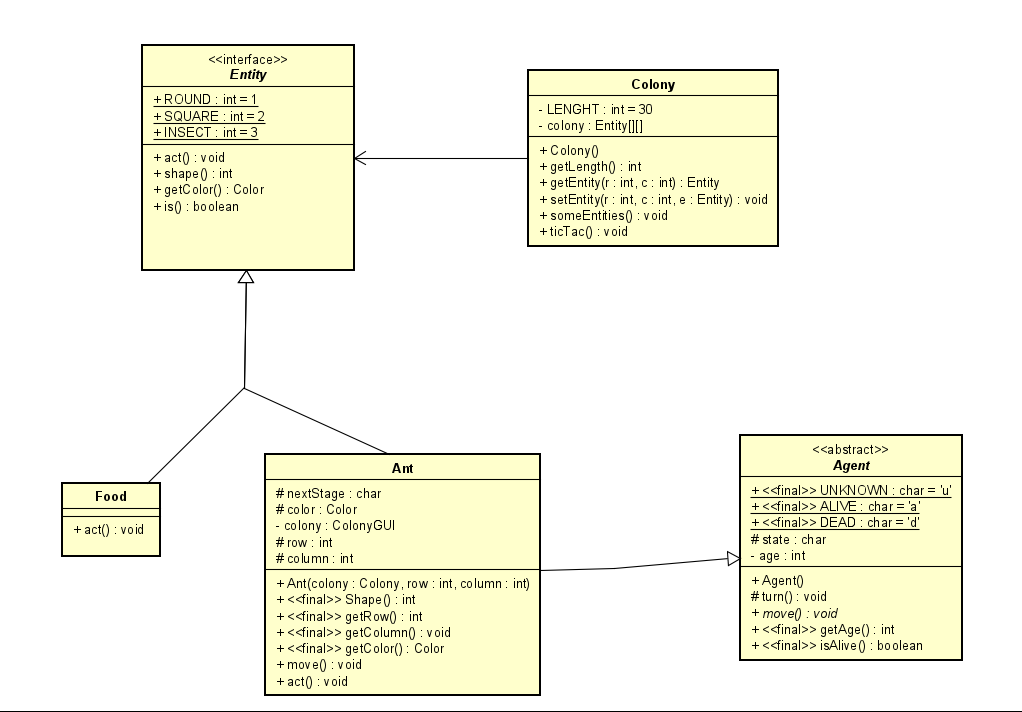
**3. Adicione al diseño la arquitectura general con un diagrama de paquetes en el que se presente los paquetes y las relaciones entre ellos. Consulte la referencia en moodle. En astah, crear un diagrama de clases (cambiar el nombre por Package Diagram0)**



**Arquitectura detallada. [En lab03.doc y colonyAstah]**

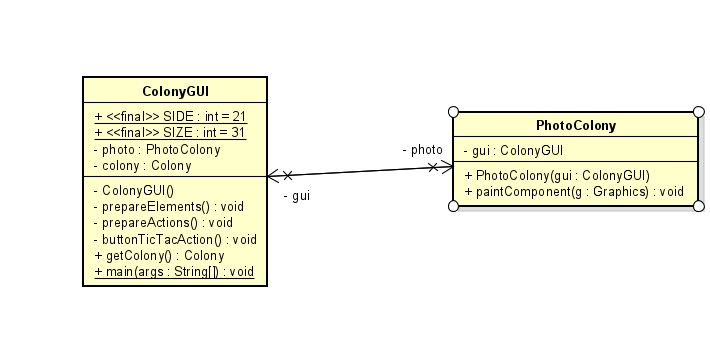
**1. Para preparar el proyecto para BDD. Completen el diseño detallado del paquete de dominio. Adicionen el diagrama de clases en el paquete correspondiente. a) ¿Qué componentes hacían falta?**

* Hacía falta la clase Food, Ant, Colony y Agent, pues ya se nos estaba dando la Interfaz Entity.



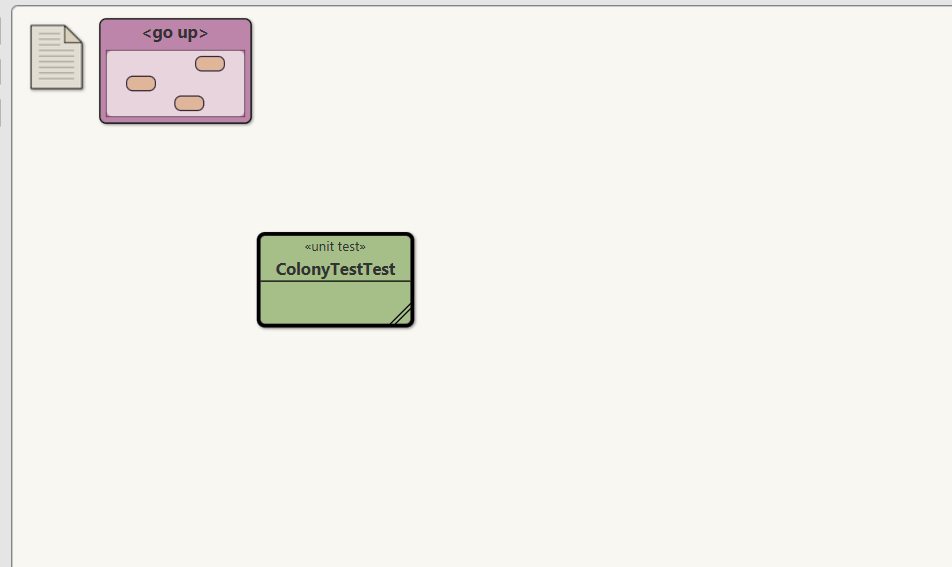
**2.** **Completen el diseño detallado del paquete de presentación. Adicionen el diagrama de clases al paquete correspondiente. a) ¿Por qué hay dos clases y un archivo java?**

Se tienen dos clases debido a que estas estan contenidas en un solo archivo java, esto nos ayuda con la encapsulación de codigo y la organización, de tal forma que sean mas accesibles, sin embargo, cada una tiene una funcionalidad diferente para la aplicación, donde “ColonyGUI” es donde se presenta la interfaz de usuario, como botones y paneles, mientras que “PhotoColony” lo que hace es dibujar y mostrar la representación gráfica de la colonia dentro de la interfaz.



**3. Adicione en las fuentes la clase de pruebas unitarias necesaria para BDD. (No lo adicione al diagrama de clases) ¿En qué paquete debe estar? ¿Por qué? ¿Asociado a qué clase? ¿Por qué?**

Esta debe ser creado como un nuevo paquete a parte de presentation y domain, en donde se van a realizar todas las pruebas, esta va principalmente asociada al paquete domain, pues aca es donde tenemos todos nuestros objetos para los cuales requerimos realizar las pruebas.



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

**(NO OLVIDE BDD – MDD)**

**1. Estudie la clase Colony ¿Qué tipo de colección usa para albergar entidades? ¿Puede recibir hormigas? ¿Por qué?**

* La clase colony utiliza una matriz para albergar entidades, esta matriz representa posiciones en una cuadricula en donde se pueden tener hormigas o comida las cuales vienen siendo subclases de la clase Entity por lo cual pueden ser albergadas.

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

* La clase “Ant” se crea en el estado inicial ‘ALIVE’ debido al nextState que hereda de la clase Agent.
* Para pintarse se usa el metodo shape, donde se retorna Un Entity.INSECT esto es lo que nos indica que se representa como un insecto.
* No sé especifica un metodo que nos diga donde aumente la edad, por lo que no se maneja en esta clase.
* La clase ant esta definida por la clase “Agent” y “Entity” siendo esta una subclase de la primera y siendo implementada por medio de una interfaz en la segunda mencionada.

**3. Ant 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.**

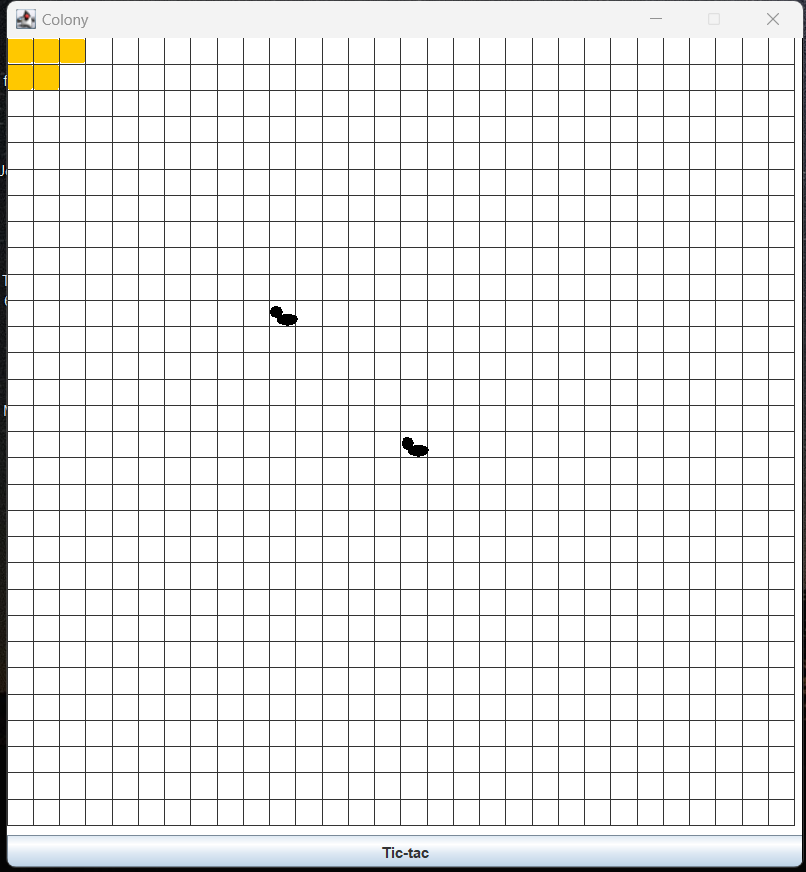
* Ant tiene los atributos State y age heredados de Agent
* Puede realizar metodos como turn(), getAge(),isAlive(), move() estos heredados de Agent.
* Lo distinto que puede hacer es el move() y el act(), debido a que estan dados como otro metodo diferente al que heredan.
* Lo que no puede hacer distinto es el turn(), getAge(), isAlive() pues estos son heredados para todas las subclases.
* Debe aprender a implementar su comportamiento, moverse por la colonia y el cómo interactuar con la comida o realizar actividades específicas en su colonia, según sean dados sus objetivos en la simulación.

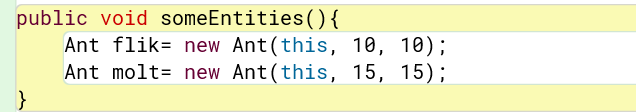
**4. Por comportarse como un Entity, ¿qué sabe hacer? ¿qué decide hacer distinto? ¿qué no puede hacer distinto? ¿qué debe aprender a hacer? Justifique sus respuestas.**

* Los metodos que debe saber hacer son act(), shape(),getColor() y is().
* Lo que puede hacer distinto es act(), shape() y getColor() debido a que este tiene otro metodo que lo reemplaza en la clase Ant.
* Lo que no puede hacer distinto es el is(), pues aplica para todas las subclases de Entity
* Debe aprender a definir su comportamiento como hormiga, debido a que no está definido en la implementación. Además del cómo interactuar con la comida o realizar actividades específicas en su colonia, según sean dados sus objetivos en la simulación.

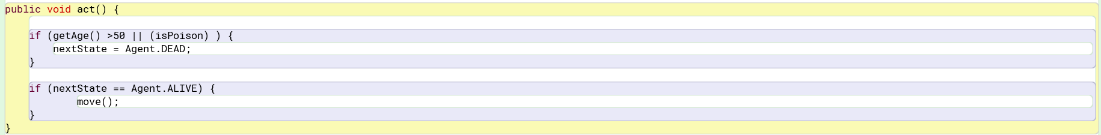
**5. Ahora vamos a crear dos hormigas en diferentes posiciones (10,10) (15,15) llámelos flik y molt usando el método someEntities() . Ejecuten el programa, ¿Qué pasa con las hormigas?¿Por qué? Capturen una palla significativa.**

Al ejecutar el programa podemos visualizar a las dos hormigas en el tablero, justo en las posiciones que le indicamos al programa por medio de la clase Colony, en el metodo someentities.



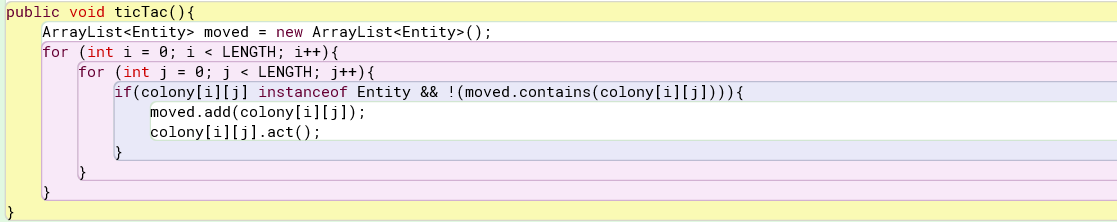


**6. Diseñen, construyan y prueben el método llamado act() de la clase Ant. La hormiga se mueve aleatoriamente y muere después de 50 tic-tac.**



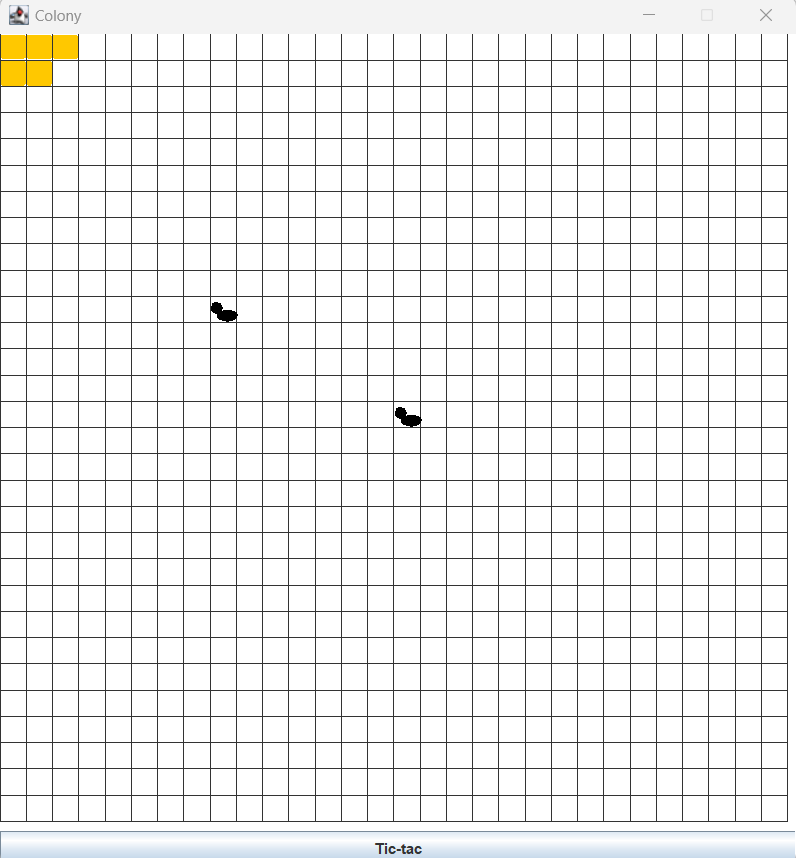


**7. Diseñen, construyan y prueben el método llamado ticTac() de la clase Colony.**



**8. ¿Cómo quedarían flik y molt después de uno, dos y tres Tic-tac? Ejecuten el programa y hagan tres clics en el botón. Capturen una palla significativa. ¿Es correcto?**

Es correcto, debido a que se movieron 3 veces en direcciones aleatorias.



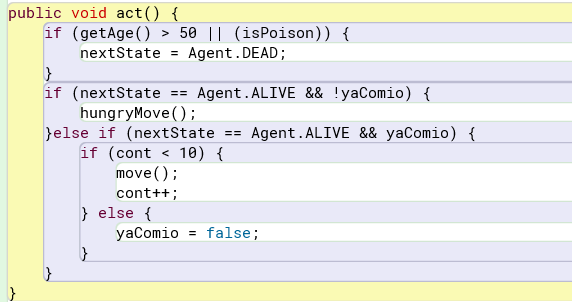
**Ciclo 2. Incluyendo a las hormigas hambrientas [En lab03.doc y colonyasta]**

**(NO OLVIDE BDD – MDD)**

El objetivo de este punto es permitir recibir en la colonia hormigas hambrientas. Las sociables (i) son de color verde; (ii) se mueven hacia la fuente de comida, (iii) una vez comen se mueven aleatoriamente hasta que vuelven a sentir hambre (10 tic-tac).

**1. Para implementar esta nueva hormiga Hambrienta ¿cuáles métodos se sobreescriben (overriding)?**

* Solo se sobreescribe el metodo act() debido a que este ahora tiene la condición de los 10 tics



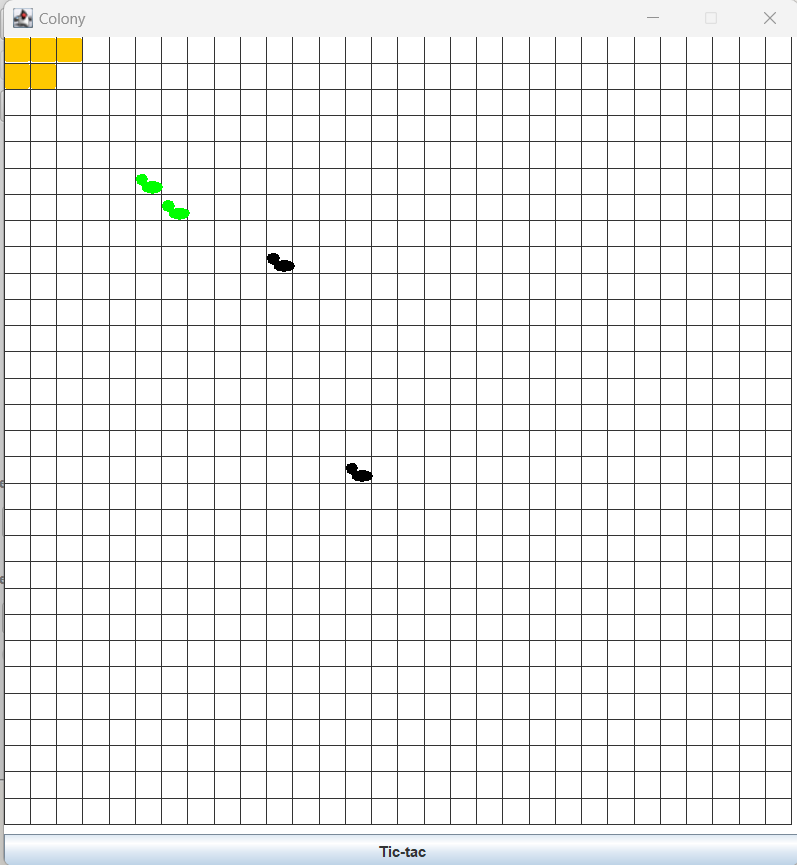
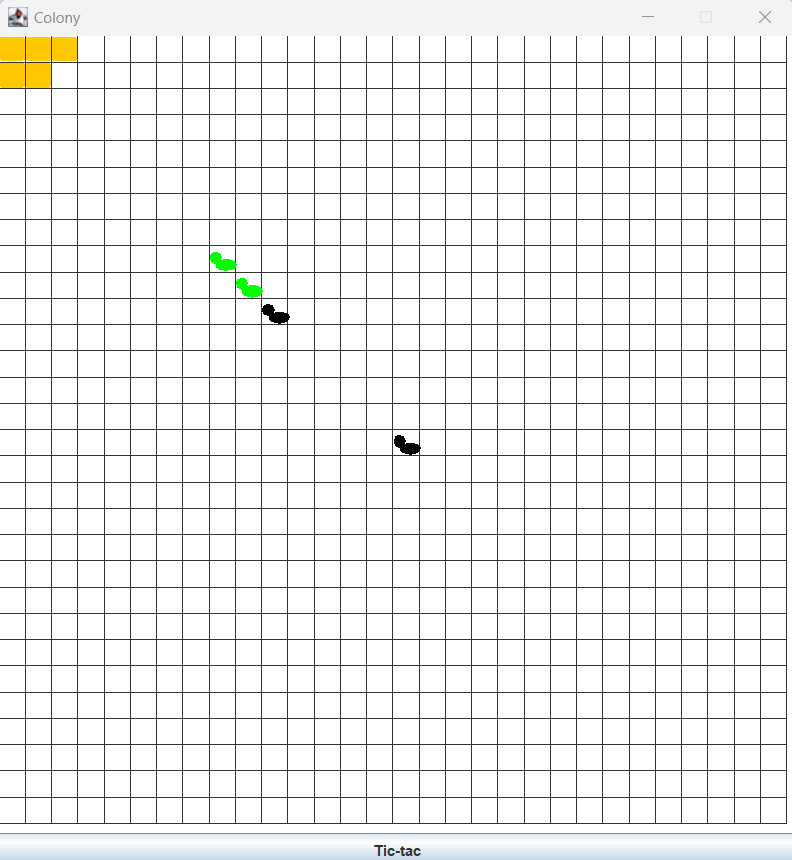
**2. Diseñen, construyan y prueben esta nueva clase.**

* Diseño en el BlueJ y en diagrama de clases del astah.

**3. Adicione una pareja de inquietas, llámelas rachel y monica,**

**(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 3 Tic-Tac las hormigas hambrientas quedan mucho más cercanas a la comida.
* Es correcto pues las hormigas deben buscar la comida.

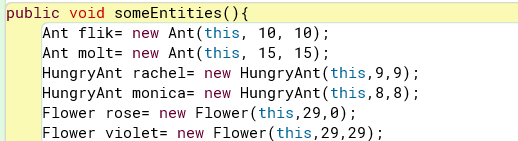


**Ciclo 3. Adicionando flores [En lab03.doc, colony.asta y \*.java]**

El objetivo de este punto es incluir en el Colony flores (sólo vamos a permitir el tipo básico de flores) las flores se abren (rojas) y se cierran (verdes). **(NO OLVIDE BDD – MDD)**

**1. Construyan la clase Flower para poder adicionaría en el Colony? ¿debe cambiar en el código del Colony en algo? ¿por qué?**

* Lo único que se debe modificar en la clase Colony, es agregar las nuevas entidades para que las flores se vean reflejadas en el tablero.

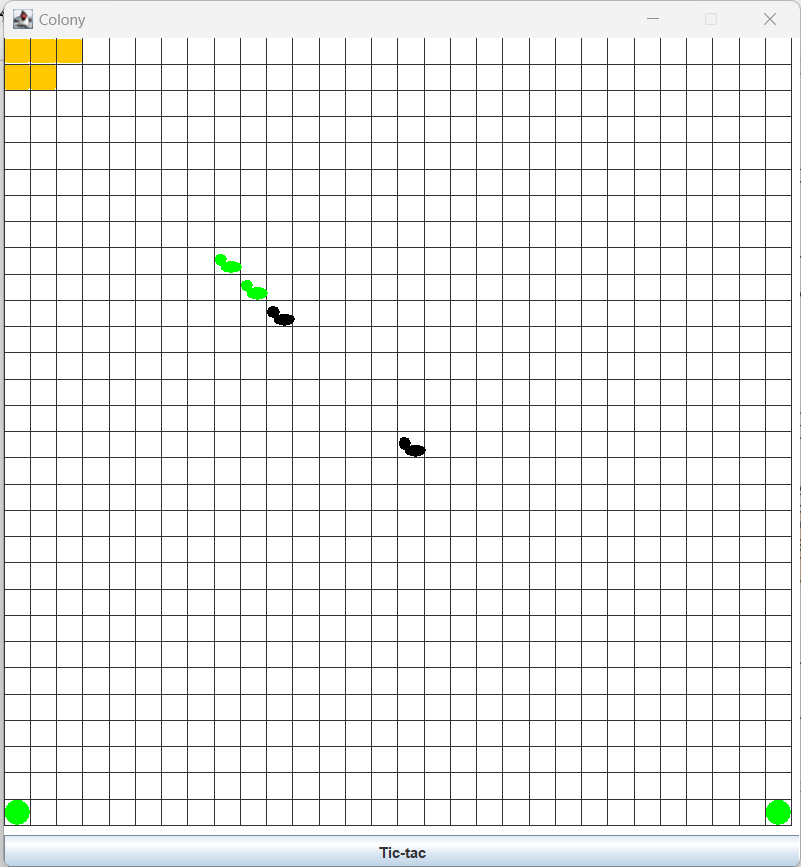


**2. Diseñen, construyan el método y prueben esta nueva clase.**

* Diseño en el blueJ y en el astah.

**3. Adicionen dos cerca en las esquinas inferiores del Colony, llámenlos rose y violet, (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?**

* Debido a que las flores cambian de color con cada Tic-Tac, al hacerlo 4 veces van a quedar del mismo color, para que queden rojas se debe hacer Tic-Tac un numero impar de veces, por lo que es correcto.



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

El objetivo de este punto es permitir recibir en un nuevo tipo de célula **(NO OLVIDE BDD – MDD)**

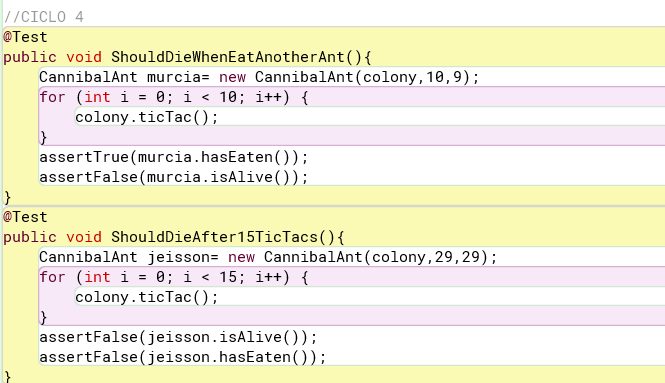
**1. Propongan, describan e Implementen el nuevo tipo de hormigas. (Mínimo dos pruebas de unidad)**

* El nuevo tipo de hormiga implementado, son las hormigas caníbales, estas son de color rojo y lo que hacen es moverse hacia la dirección en la que haya otras hormigas y comérselas, una vez lo hacen, estas mueren, o en caso de que se hagan 15 tics también morirán.

**2. Considerando una pareja de ellas con el nombre de ustedes. (a) Piensen una prueba significativa y expliquen la intención.**

* La hormiga se come a otra y esta muere.
* Pasan los 15 tics y si no ha comido a nadie muere.

**(b) Codifiquen las pruebas de unidad correspondiente y capturen la palla de resultados**

**.**

**(c) Ejecuten el programa con esa prueba para capturar las pallas correspondientes.**



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

El objetivo de este punto es permitir recibir en una nueva entidad (no hormiga) en el Colony.

**(NO OLVIDE BDD – MDD)**

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

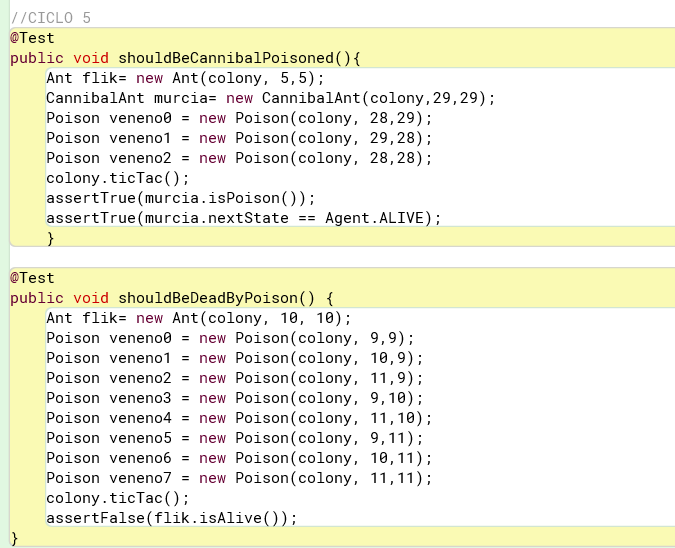
* La nueva entidad implementada es Poison, la cual viene siendo un veneno el cual si pisan accidentalmente las hormigas estas automáticamente van a morir, sin embargo, si la pisa una hormiga Canibal, esta no se verá afectada.

**2. Considerando un par de ellos con los nombres de ustedes.**

**(a) Piensen una prueba significativa y expliquen la intención.**

* La hormiga caníbal pisa el veneno y su estado no cambia a pesar de que este envenanada.
* La hormiga cambia su estado a muerta una vez pisa el veneno.

**(b) Codifiquen las pruebas de unidad correspondiente y capturen la palla de resultados.**



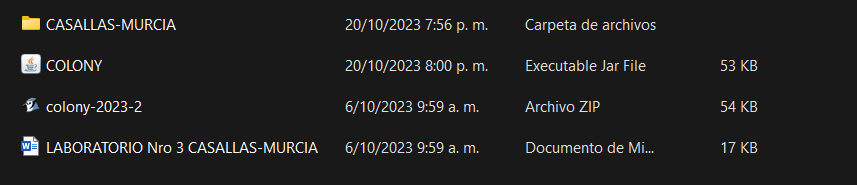
**(c) Ejecuten el programa con esa prueba para capturar las pallas correspondientes.**

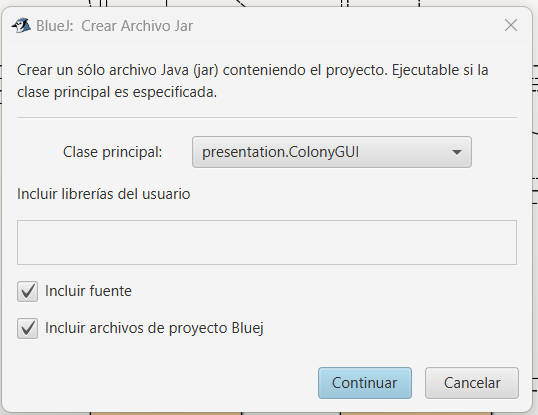


**Empaquetando la versión final para el usuario.**

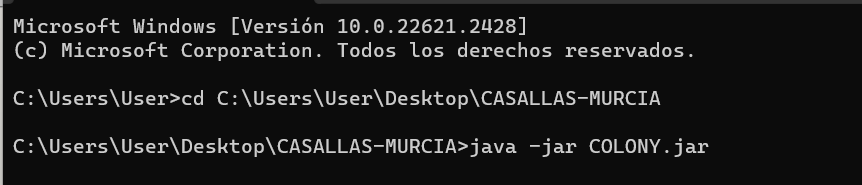
**[En lab03.doc, colony.asta , \*.java, colony.jar]**

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? R



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

La ventaja que tiene crear el archivo .jar es ejecutar nuestro programa sin necesidad de tener que acceder al paquete de BlueJ, si no ya vamos a tener el ejecutable del simulador como un archivo separado, lo que facilita el proceso de ver su funcionamiento.

**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 erior.

**Comandos básicos del sistema operativo [En lab03.doc]**

es 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.**

**1. Moverse en la estructura de directorios:**

* cd /ruta/al/directorio # Cambia al directorio especificado
* cd .. # Retrocede un directorio
* cd # Cambia al directorio de inicio del usuario

**2.Crear un directorio:**

mkdir (nombre\_del\_directorio)

**3.Borrar un directorio:**

**rmdir (Remove Directory):**

* rmdir nombre\_del\_directorio

**rm (Remove):**

* rmdir nombre\_del\_directorio

**4.Listar el contenido de un directorio:**

* ls # Lista el contenido del directorio actual
* ls /ruta/al/directorio # Lista el contenido de un directorio específico
* ls -l # Lista con detalles, incluyendo permisos y tamaños de archivo

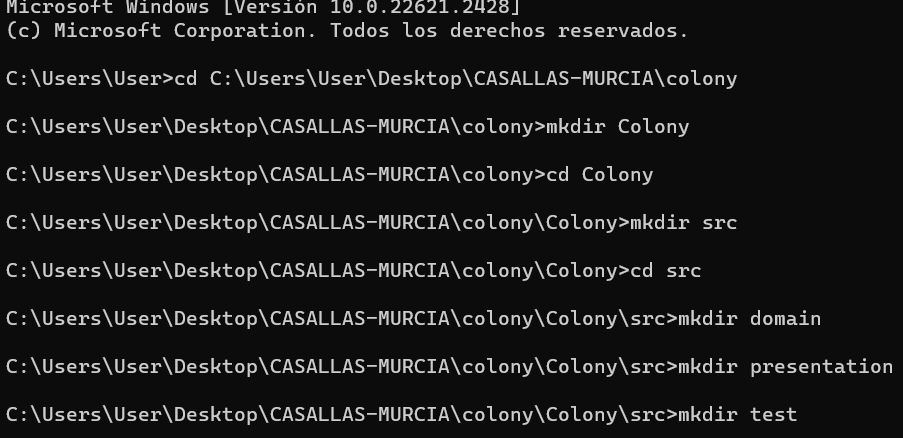
**5. Copiar un archivo:**

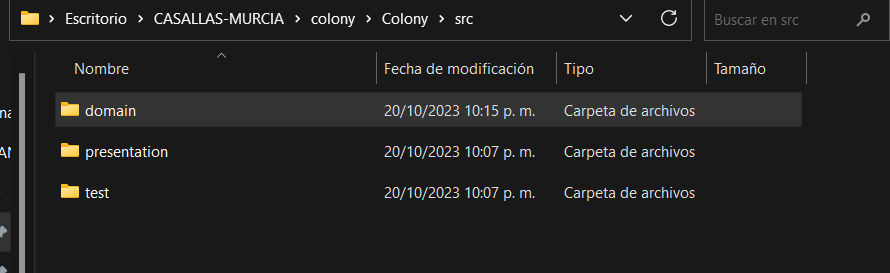
* cp archivo\_original destino # Copia un archivo a una ubicación específica
* cp -r directorio\_original destino # Copia un directorio de forma recursiva

**6.Eliminar un archivo:**

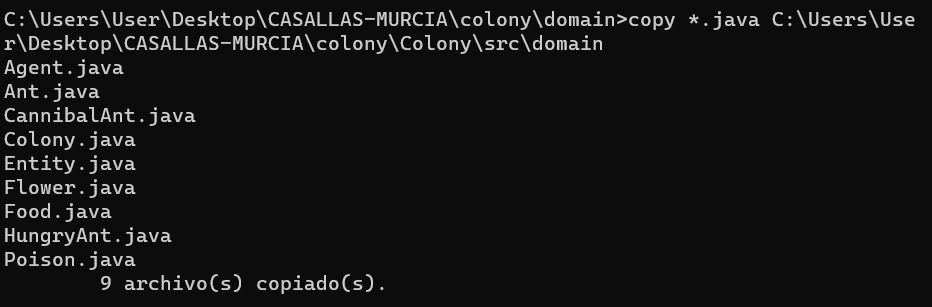
* rm archivo\_a\_eliminar

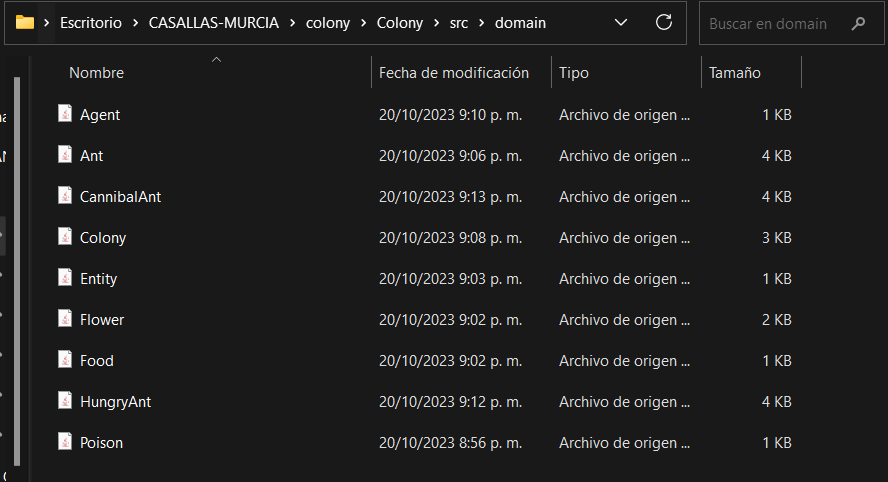
**2. Organicen un nuevo directorio con la estructura propuesta para probar desde allí su habilidad con los comandos de consola. Consulten y capturen el contenido de su directorio**





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





**Estructura de proyectos java [En lab03.doc]**

En java los proyectos se estructuran considerando tres directorios básicos.

colony

src

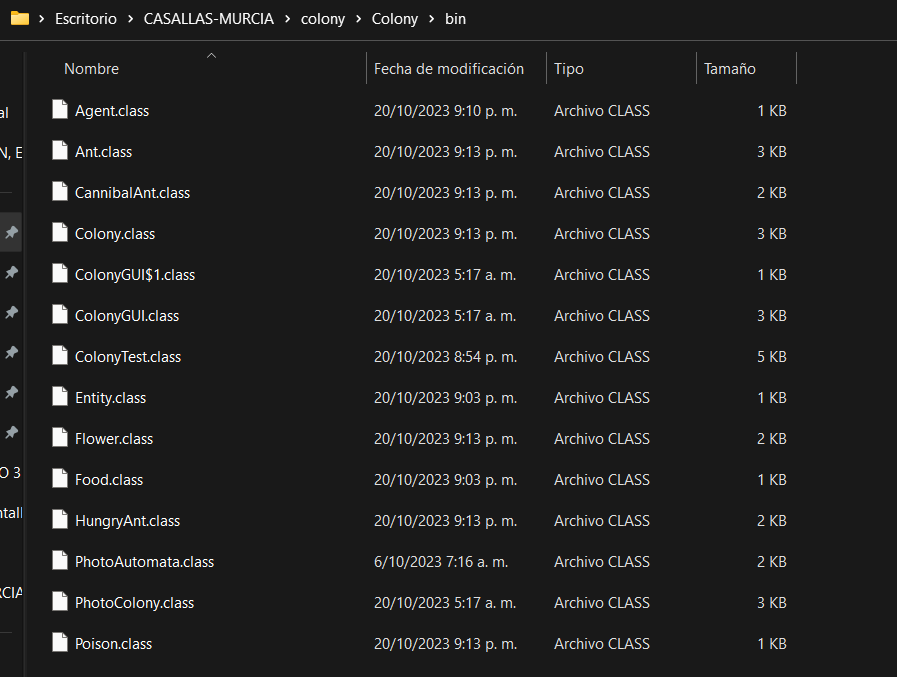
bin

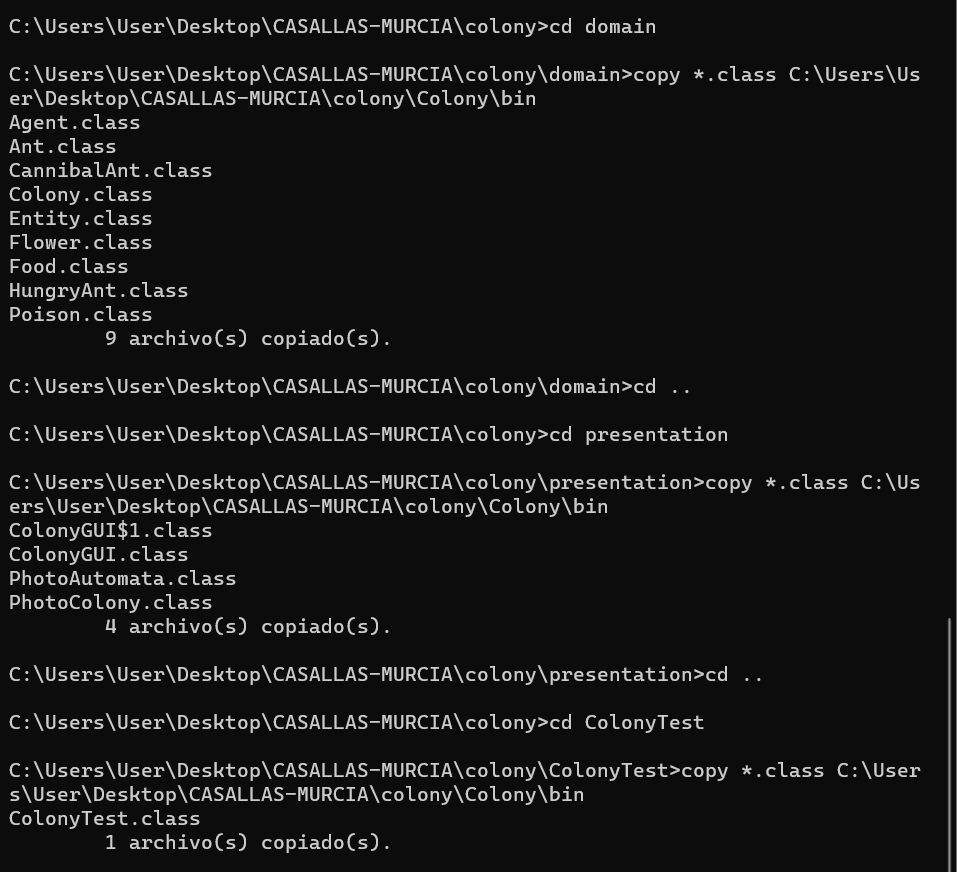
docs

**1. Investiguen los archivos que deben quedar en cada una de esas carpetas y la organización interna de cada una de ellas.**

* El directorio "src" se utiliza para almacenar el código fuente de tu proyecto. Aquí es donde se colocan todos los archivos fuente de Java que componen tu aplicación. Es decir, donde van los .java
* El directorio "bin" se utiliza para almacenar los archivos binarios generados a partir del código fuente. Estos archivos binarios suelen ser archivos compilados ".class" y otros recursos compilados.
* El directorio "docs" se utiliza para almacenar documentación relacionada con el proyecto, como archivos README, documentos de diseño, manuales de usuario, o documentación técnica. Incluso Archivos UML.

**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:** El comando javac es el compilador de Java. Se utiliza para compilar archivos fuente de Java (archivos con extensión .java) en archivos bytecode de Java (archivos con extensión .class) que pueden ser ejecutados por la máquina virtual de Java (JVM).

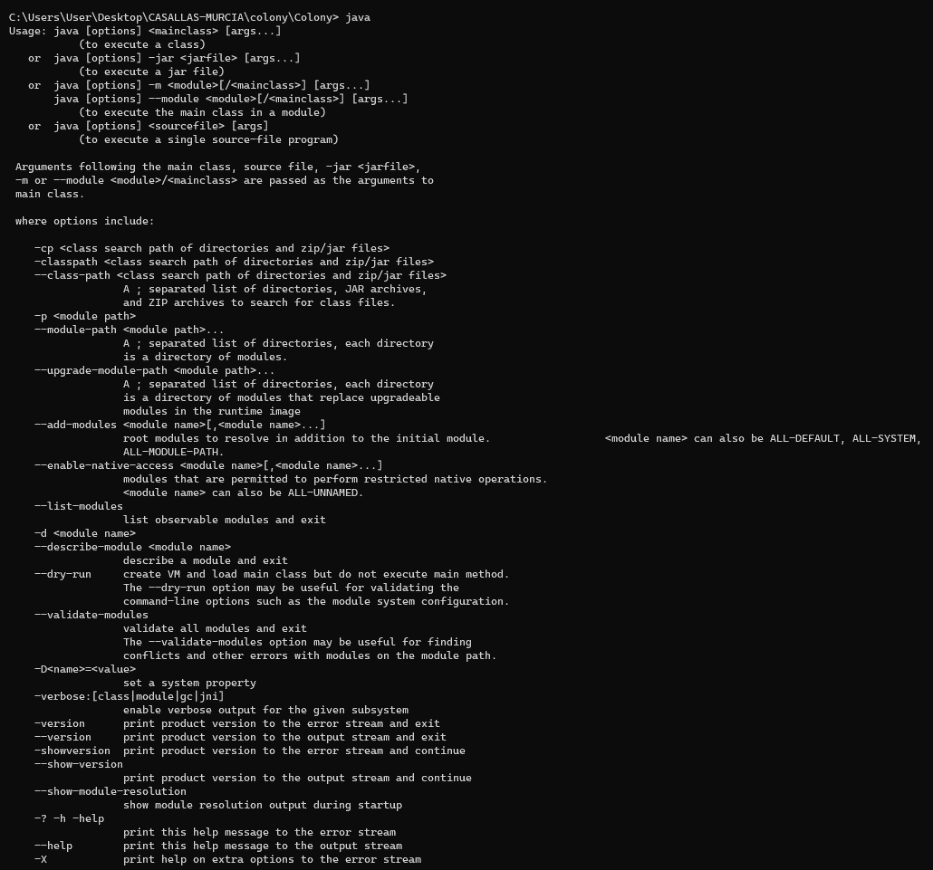
**Java:** El comando java es el intérprete de Java. Se utiliza para ejecutar programas Java compilados previamente (archivos .class) en la máquina virtual de Java (JVM).

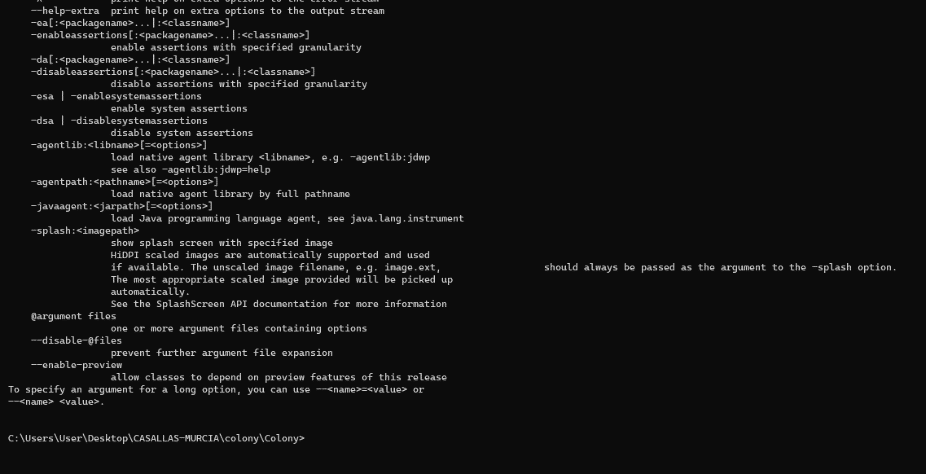
**Javadoc:** El comando javadoc es una herramienta que genera documentación a partir de comentarios de estilo especial en el código fuente de Java. Estos comentarios se conocen como comentarios de documentación y suelen comenzar con /\*\* y contienen información sobre clases, métodos, variables y otros elementos del código.

**jar:** El comando jar se utiliza para crear, manipular y extraer archivos JAR (Java Archive). Los archivos JAR son archivos comprimidos que pueden contener clases Java, recursos, bibliotecas y metadatos. Son utilizados para distribuir y empaquetar aplicaciones Java.

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

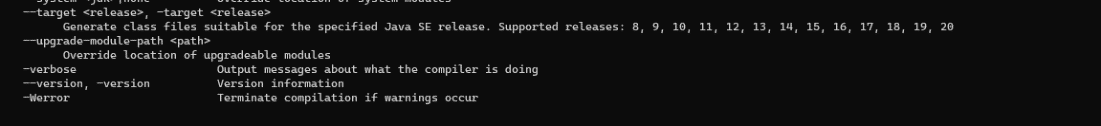
**JAVA:**



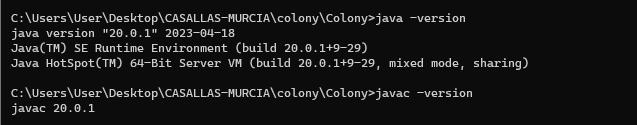


**JAVAC:**





**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 colony 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.**

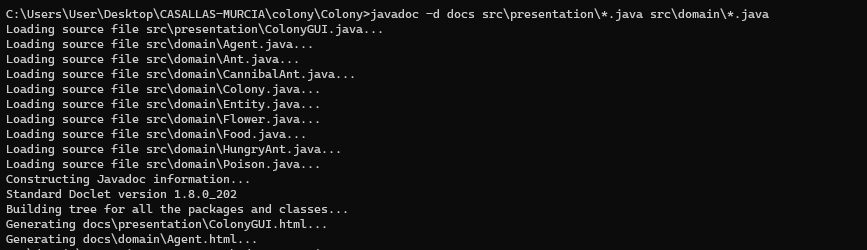


**2. Revise de nuevo el contenido del directorio de trabajo y sus subdirectorios. ¿Cuáles nuevos archivos aparecen ahora y dónde se ubican?**

* Ahora en la carpeta de bin, tenemos dos directorios llamados domain y presentation, en donde se encuentran sus respectivos archivos .class organizados respectivamente.

**Documentando [En lab03.doc]**

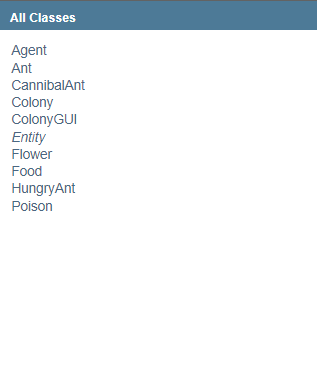
**1. Utilizando el comando javadoc, desde el directorio raiz, genere la documentación (API) en formato html, en este directorio. ¿cuál es el comando completo para generar esta documentación?**



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



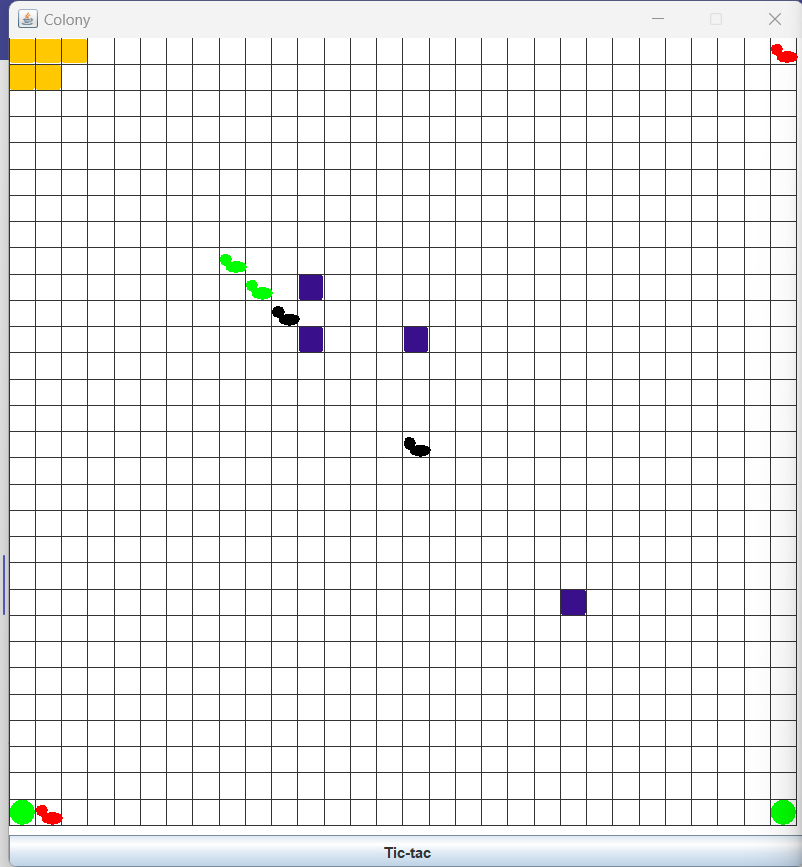
* Este archivo al ejecutarlo nos va a abrir un menú en donde vamos a poder ver la documentación de todas las clases que tenemos.



**Ejecutando [En lab03.doc]**

**4. Empleando el comando java, desde el directorio raiz, ejecute el programa. ¿Cómo utilizó este comando?**





**Probando [En lab03.doc]**

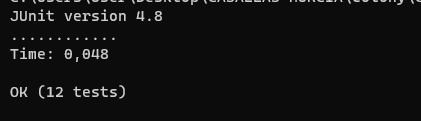
**1. Adicione ahora los archivos del directorio pruebas y trate de compilar nuevamente el programa. Tenga en cuenta que estas clases requieren la librería junit 4.8. ¿Cómo se incluye un paquete para compilar? ¿Qué instrucción completa tuvo que dar a la consola para compilar?**



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



**3. Pegue en su documento el resultado de las pruebas**

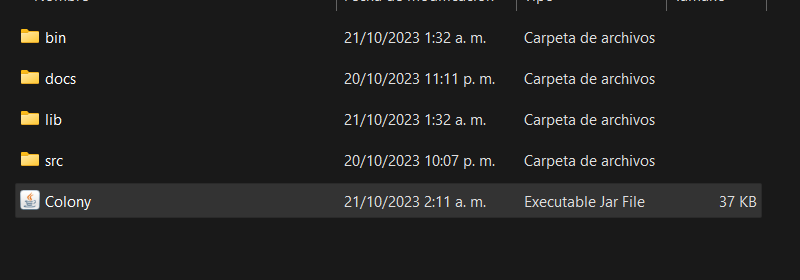


**Empaquetando [En lab03.doc]**

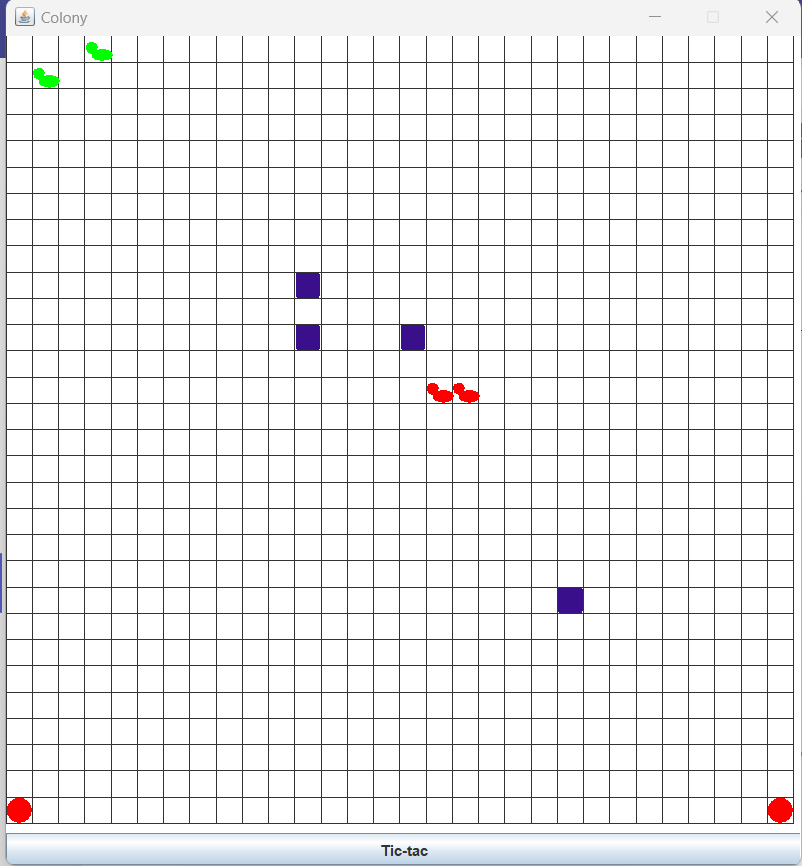
1. Consulte como utilizar desde consola el comando jar para empaquetar su programa entregable en un archivo .jar, que contenga los archivos bytecode necesarios (no las fuentes ni las clases de prueba), y que se pueda ejecutar al instalarlo en cualquier directorio, con solo tener la máquina virtual de java y su entorno de ejecución (JRE). ¿Cómo empaquetó jar ?



2. ¿Cómo se ejecuta el proyecto empaquetado?



Se ejecuta por medio de un archivo. Jar, que al abrirlo podemos visualizar el simulador.



**RETROSPECTIVA**

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

24 horas por persona.

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

Se encuentra realizado en su totalidad y todo se encuentra funcional.

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

La refactorización de código nos sirvió, pues gracias a esta pudimos corregir problemas que teníamos con el simulador, pensándolos de otra manera.

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

El mayor logro fue por investigación propia poder aprender a ejecutar los comandos en consola, sin necesidad de ayuda externa, en donde el mayor problema fue encontrar información al respecto, sin embargo, después de dedicarle tiempo se logró.

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

Trabajar a la par, con el fin de que los dos entendiéramos lo que se estaba realizando. Nos comprometemos a seguir así.