# PROGRAMACIÓN ORIENTADA A OBJETOS

**Herencia e interfaces ADEMAS Java desde consola 2023-2**

# Laboratorio 3/6

# CRISTIAN JAVIER ALVAREZ – JULIANA BRICEÑO

**OBJETIVOS**

Desarrollar competencias básicas para:

1. Aprovechar los mecanismos de la herencia y el uso de interfaces.
2. Organizar las fuentes en paquetes.
3. Usar la utilidad jar de java para entregar una aplicación.
4. Extender una aplicación cumpliendo especificaciones de diseño, estándares y verificando su corrección.
5. Vivenciar las prácticas XP : Code must be written to agreed [standards](http://www.extremeprogramming.org/rules/standards.html). Code the [unit test first](http://www.extremeprogramming.org/rules/testfirst.html).



1. Utilizar los programas básicos de java (javac, java, javadoc, jar), desde la consola.

# ENTREGA

* Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
* Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios preparados para tal fin.

DESARROLLO

# Contexto

Un sistema complejo es un sistema compuesto de partes interrelacionadas que como un conjunto exhiben propiedades y comportamientos no evidentes a partir de la suma de las partes individuales.

Las hormigas vistas de manera individual tienen un comportamiento muy simple, para buscar comida realizan una caminata aleatoria. Pero trabajando en conjunto, como colonia, pueden encontrar un camino óptimo a la comida, es decir pueden ir encontrando rutas más cortas del hormiguero a la fuente de comida. Para hacer esto no hay ningún líder que les vaya diciendo por dónde irse. Van encontrando el mejor camino medie la interacción que tienen entre ellas y entre su ambiente.

**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? b) ¿Cuál es el propósito del paquete presentación? c) ¿Cuál es el propósito del paquete dominio?
2. Tiene dos paquetes
3. Mostrar la parte gráfica del problema
4. Agrupa toda la lógica y la interacción entre las clases
5. Revisen el paquete de dominio, a)¿Cuáles son los diferentes tipos de componentes de este paquete? b) ¿Qué implica cada uno de estos tipos de componentes?
6. Tenemos 1 interfaz y 4 clases, de las cuales 1 es abstracta, Interfaz: Entity, Clases: Food, Ant, Colony, Clase abstracta: Agent

Interfaces:

Implicaciones:

* + Las interfaces definen un conjunto de métodos que deben ser implementados por las clases que las implementan. No pueden contener campos (variables de instancia) ni implementaciones de métodos.
  + Las clases que implementan una interfaz deben proporcionar implementaciones concretas para todos los métodos de la interfaz, lo que garantiza que cierto comportamiento esté disponible.
  + Una clase puede implementar múltiples interfaces, lo que permite la herencia múltiple de tipos.
  + Las interfaces son útiles para definir contratos y establecer una estructura común para clases relacionadas.

Abstractas:

Implicaciones:

* Las clases abstractas son clases que no pueden ser instanciadas directamente; en su lugar, se utilizan como plantillas para clases concretas.
* Pueden contener campos y métodos concretos, así como métodos abstractos (sin implementación) que deben ser implementados por las clases derivadas.
* Las clases derivadas (subclases) deben proporcionar implementaciones concretas para todos los métodos abstractos de la clase abstracta.
* Las clases abstractas son útiles cuando se desea proporcionar una base común para varias clases relacionadas y garantizar que cierto comportamiento esté presente en todas ellas.

Concretas:

Implicaciones:

* Las clases normales o concretas son clases que pueden ser instanciadas directamente para crear objetos.
* Pueden contener campos, métodos concretos y métodos abstractos (si se declaran como tal).
* No tienen restricciones especiales en términos de implementación y se utilizan para crear objetos reales y funcionales en tu aplicación.

1. Revisen el paquete de presentación, a) ¿Cuántos componentes tiene? b) ¿Cuántos métodos ofrece?
2. Tiene 1 clase normal
3. Ofrece actualmente solo un método, que es el método main, para ejectuar el programa.
4. Para ejecutar un programa en java, ¿Qué método se debe ejecutar? ¿En qué clase de colony se encuentra?

R//: Para ejecutar el programa se debe ejecutar el main que esta en el paquete de Presentation en la clase ColonyGUI

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

R//: Ofrece actualmente solo un método, que es el botón ti tac. Y nos muestra un tablero cuadriculado con una hormiga en una casilla y unas casillas pintadas de amarillo.

# (Deben ejecutar la aplicación java, no crear un objeto como lo veníamos haciendo)

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

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.

R//:

En Java, "package" y "import" son dos conceptos relacionados con la organización y el acceso a clases y recursos en un programa.

* Package (Paquete):

Un paquete en Java es un mecanismo de organización que agrupa un conjunto de clases y otros recursos relacionados en una estructura jerárquica. Los paquetes se utilizan para organizar y gestionar el código de una aplicación de manera más ordenada y evitar conflictos de nombres entre clases.

Los paquetes también pueden controlar el acceso a clases y miembros, permitiendo que algunas clases sean públicas y otras sean "package-private," lo que significa que solo son accesibles dentro del mismo paquete.

* Import:

La declaración "import" en Java se utiliza para dar acceso a clases o recursos de otros paquetes en tu programa. Permite usar clases de otros paquetes en tu código sin necesidad de escribir el nombre completo de la clase cada vez que la utilizas.

Importar un paquete te permite acceder a las clases que están dentro de ese paquete y usarlas en tu programa sin tener que usar su nombre completo (incluyendo el nombre del paquete) cada vez.

En el programa que nos dan, podemos ver que en el paquete Presentation se importa el paquete de domain.

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

R//: Tenemos una carpeta doc que contiene la documentación en archivos html, una carpeta llamada domain donde están descritos los .class .java y .cxtx de domain (paquete donde esta la lógica del programa), y tenemos la carpeta de presentation donde estan descritos los .class .java y .cxtx de presentation (paquete que nos da la representación visual del programa)

1. 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 colonyasta]

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?
2. Hacia falta las clases Ant, Food, Colony
3. 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?
4. Porque la clase esta definida dentro del mismo archivo de colonyGUI, y el .java es el código fuente
5. 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?

R//: Las pruebas deben estar en Domain porque en este paquete se realiza toda la lógica, por lo tanto es necesario las pruebas en este paquete. Se asocian a las clases de Ant, Colony y Food que son las clases concretas, no se hacen pruebas a la interfaz ni a la clase abstracta.

**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?

R//: Utiliza un vector 2D de entidades, si puede recibir hormigas porque las hormigas implementan entity.

1. 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.

R//: Se crean en estado Alive, Usa para pintarse la forma INSECT, La edad aumenta cuando se utilice el método turn, a Ant la definen que es un Agente(herencia) e implementa Entity(interfaz).

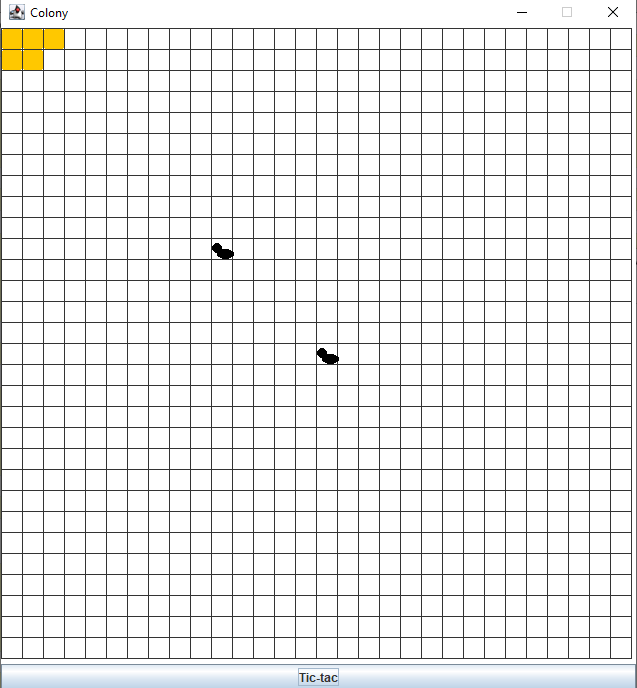
1. Ant por ser un Agent, ¿atributos tiene? ¿qué puede hacer (métodos)? ¿qué decide hacer distinto? ¿qué no puede hacer distinto a todos los agentes? ¿qué debe aprendar a hacer? Justifique sus respuestas.

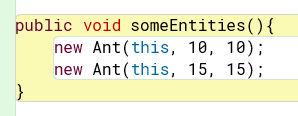
R//: Por ser un agent tiene de atributos UKNOWN, ALIVE, DEAD, state, age. Los métodos que puede hacer son: turn, move, getAge, isAlive. No puede hacer distinto getAge e isAlive porque son métodos finales. Debe aprender a hacer move porque es un método abstracto.

1. 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.

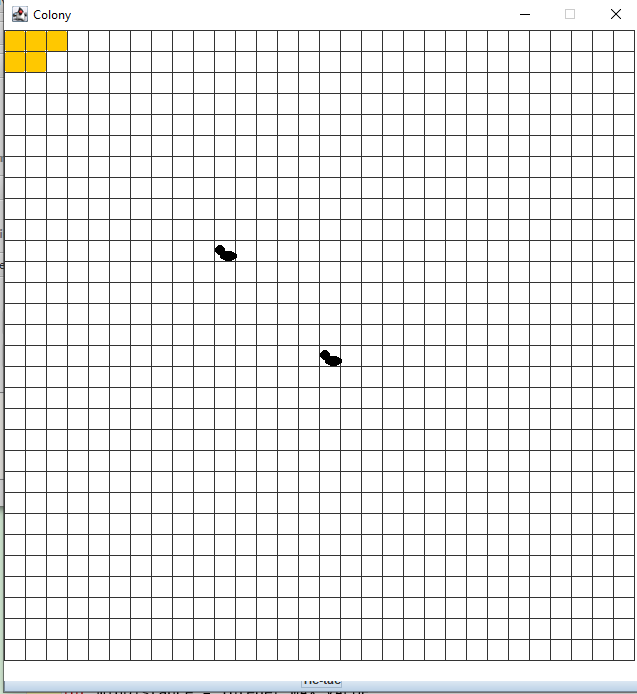
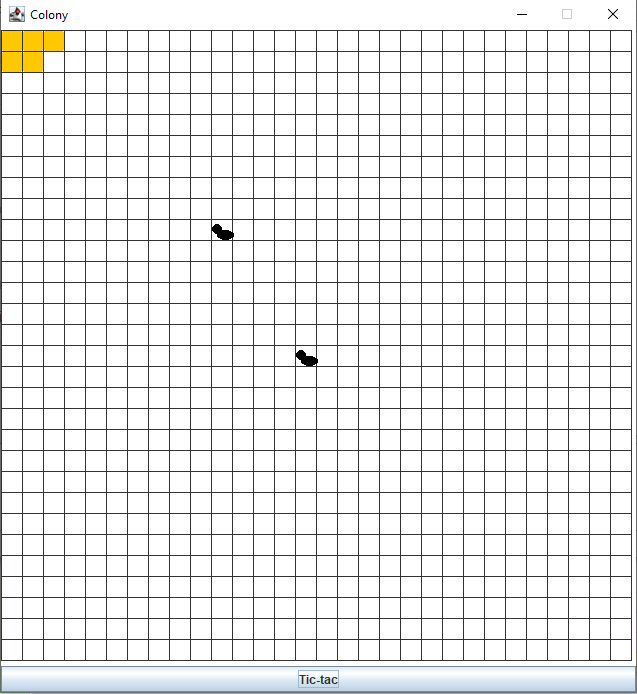
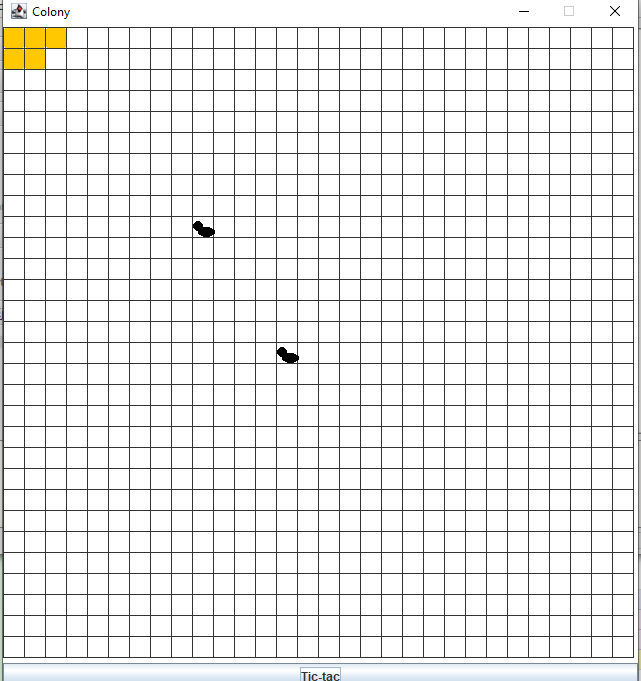
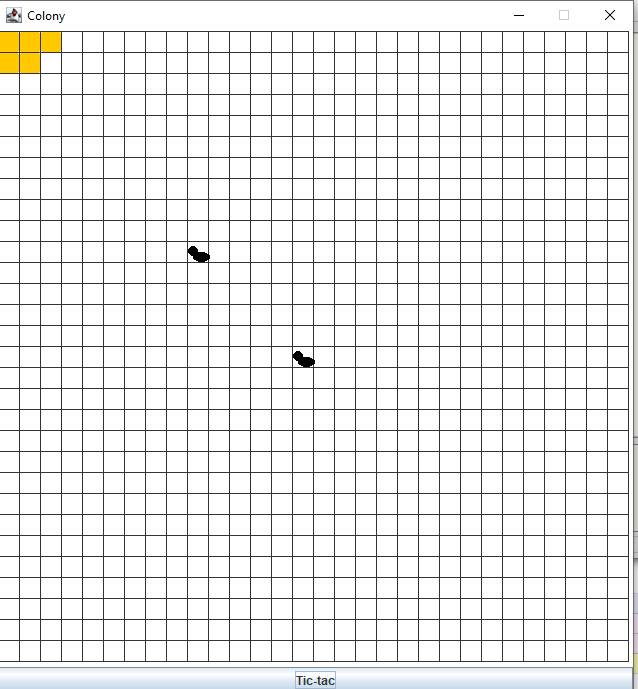
R//: Por ser una entity puede act, shape, getColor, is. Decide hacer distinto act porque es abstracto(no esta definido el cuerpo). No puede hacer distinto shape, getColor, is porque son métodos default. Debe aprender a hacer act

1. 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.

R// 



1. Diseñen, construyan y prueben el método llamado act() de la clase Ant. La hormiga se mueve aleatoriamente y muere después de 50 tic-tac.
2. Diseñen, construyan y prueben el método llamado ticTac() de la clase Colony.
3. ¿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?

R//:Después de 3 tictac flik y molt estan en posiciones diferentes a como arrancaron, esta correcto asi.

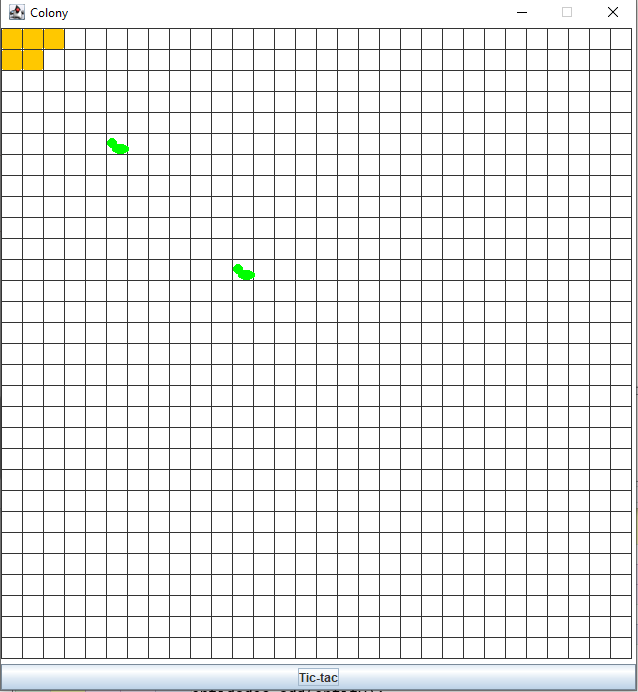
**Ciclo 2. Incluyendo a las hormigas hambrientas** [En lab03.doc y colonyasta]

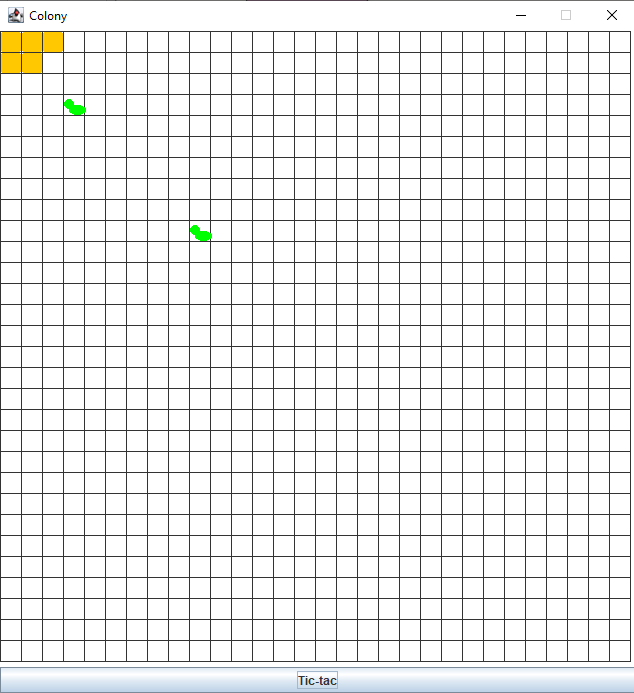
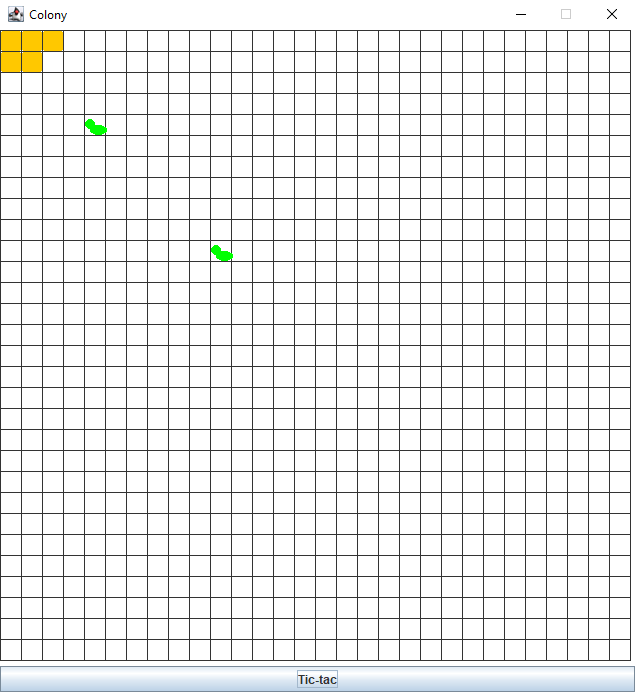
# (NO OLVIDE BDD – MDD)

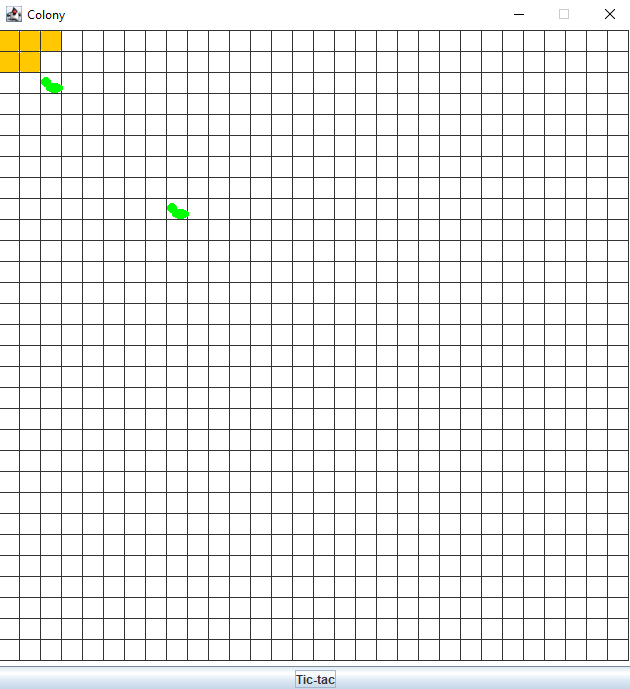
El objetivo de este punto es permitir recibir en la colonia hormigas hambrientas. Las sociables (i) son de color verde; (ii) se mueven hacia la fuente de comida, (iii) una vez comen se mueven aleatoriamente hasta que vuelven a sentir hambre (10 tic-tac).

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

R//: es necesario sobreescribir el move() ya que estas tienen un objetivo para moverse, a diferencia de su padre

1. Diseñen, construyan y prueben esta nueva clase.
2. 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 tacs podemos observar como están mas cerca de la comida, están correctamente yendo a buscarla.

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

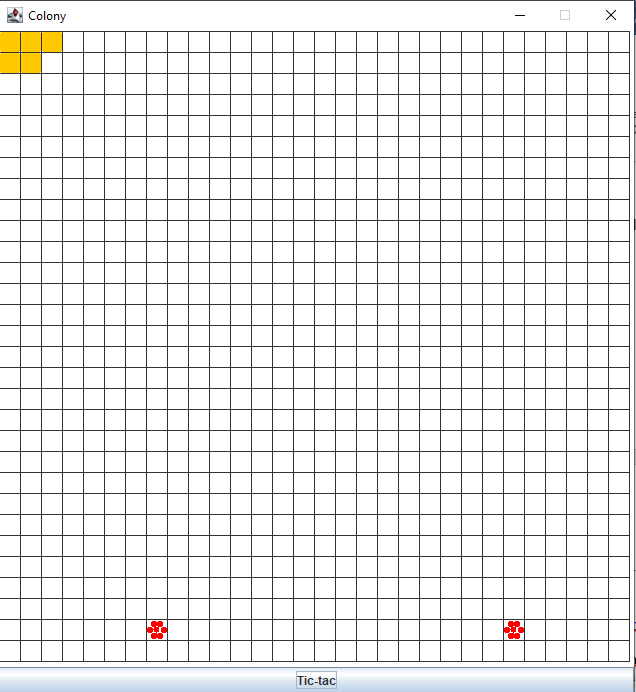
El objetivo de este punto es incluir en el Colony flores (sólo vamos a permitir el tipo básico de flores) las flores se abren (rojas) y se cierran (verdes).

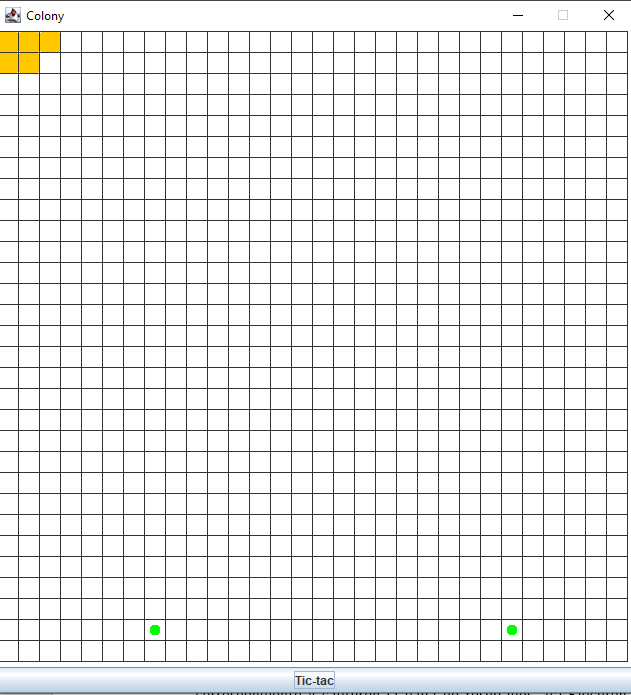
# (NO OLVIDE BDD – MDD)

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

R//: No deberíamos cambiar nada porque los métodos que diseñamos en colony están pensados para ser usados en entidades, como flower es una entidad no debería dañar lo que ya estaba hecho, asi seguimos los principios SOLID.

1. Diseñen , construyan el método y prueben esta nueva clase.
2. Adicionen dos cerca en las esquinas inferiores del Colony, llámenlos rose y violet,
   1. ¿Cómo quedarían después de cuatro **Tic-tac**? Ejecuten el programa y hagan cuatro clics en el botón. Capturen una palla significativa. (b) ¿Es correcto?





Las flores se abren y se cierran cuando ocurren los tic tacs.

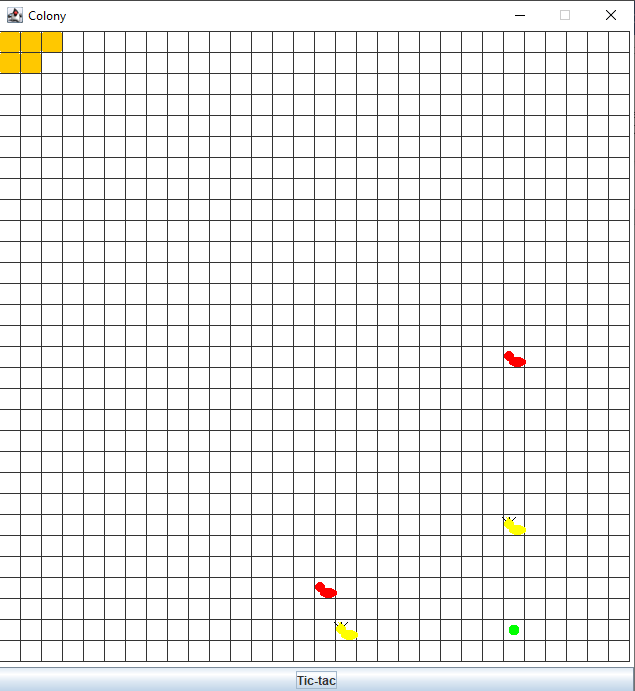
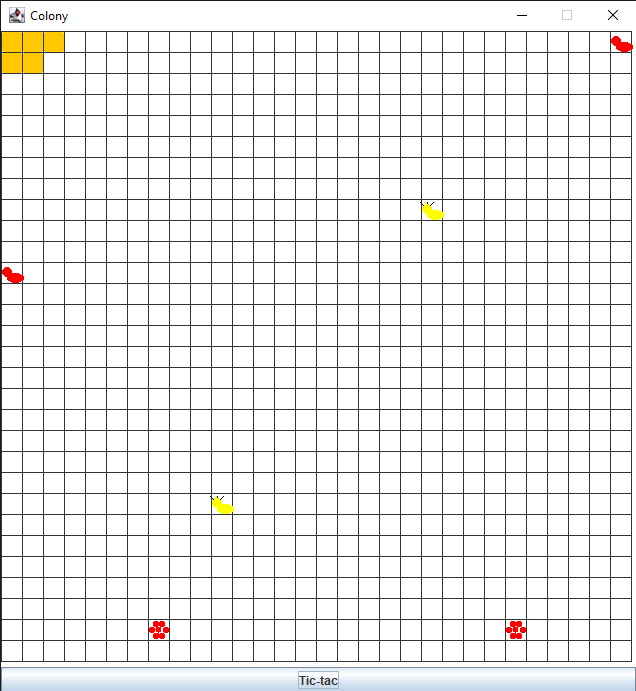
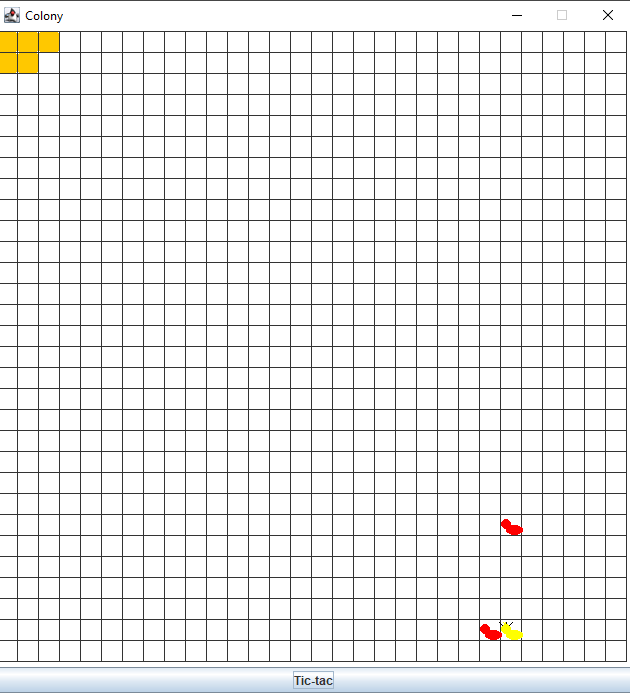
# Ciclo 4. Nueva hormiga: Proponiendo y diseñando

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)
2. Considerando una pareja de ellas con el nombre 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.

R//: a) vamos a generar las hormigas Guardianes y que persigan a las entidades que no son hormigas, en este caso a las abejas.

# Ciclo 5. Nueva entidad: Proponiendo y diseñando

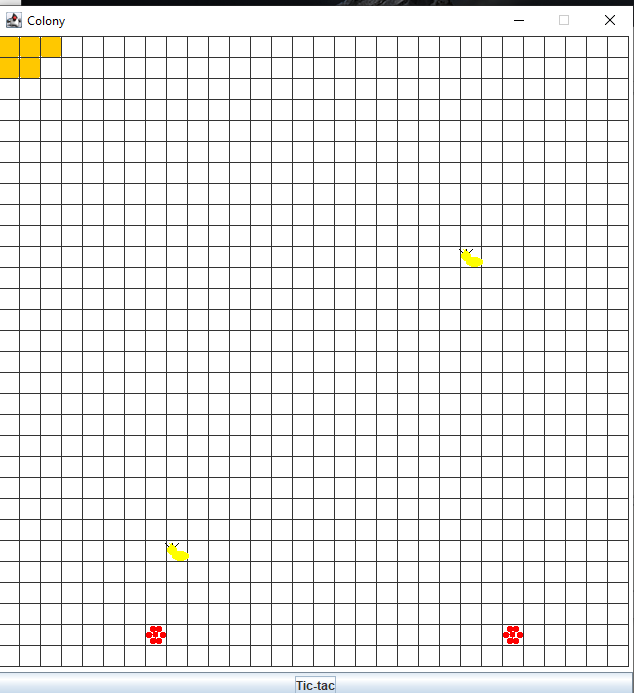
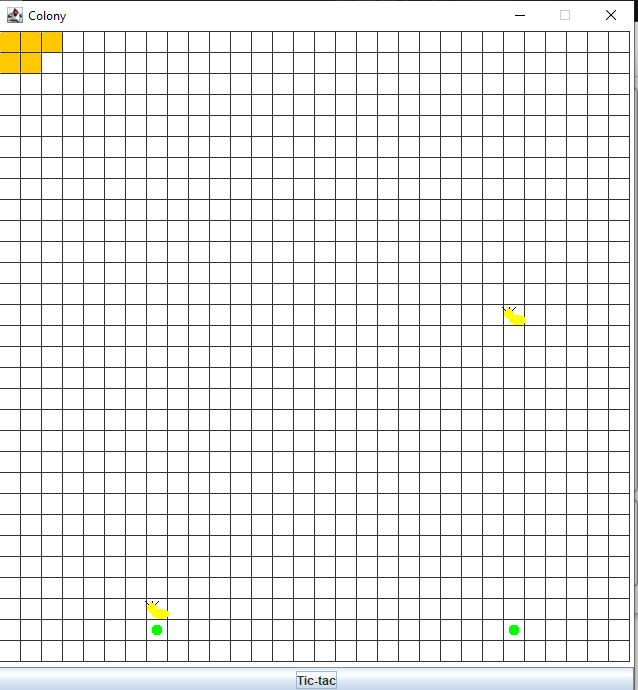
El objetivo de este punto es permitir recibir en una nueva entidad (no hormiga) en el

Colony.

# (NO OLVIDE BDD – MDD)

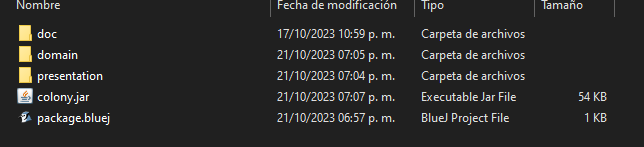
1. Propongan, describan e Implementen un nuevo tipo de Entity. (Mínimo dos pruebas de unidad)
2. Considerando un par de ellos con los nombres de ustedes . (a) Piensen una prueba significativa y expliquen la intención. (b) Codifiquen la pruebas de unidad correspondiente y capturen la palla de resultados. (c) Ejecuten el programa con esa prueba para capturar las pallas correspondientes.

R//: Propusimos una entidad que es abeja, la cual va a buscar las flores para polinizarlas

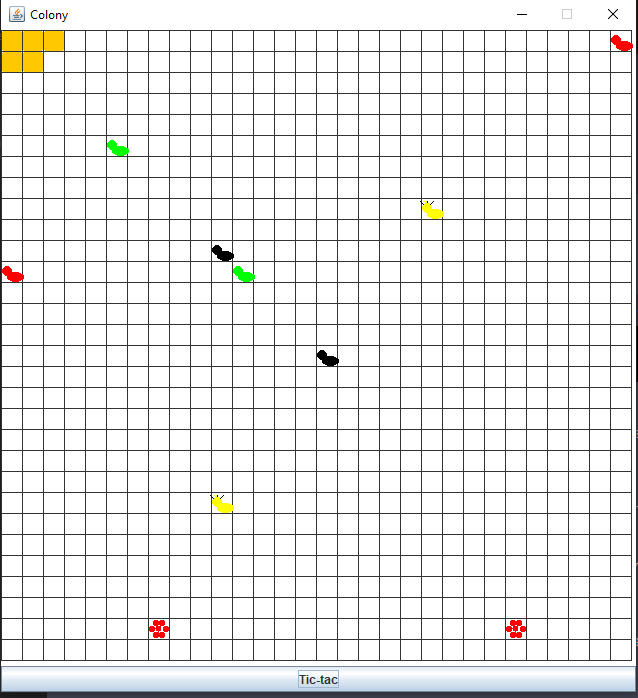
**Empaquetando la versión final para el usuario.** [En lab03.doc, colony.asta , \*.java, colony.jar]

1. Revise las opciones de BlueJ para empaquetar su programa entregable en un archivo .jar. Genere el archivo correspondiente.



1. Consulte el comando java para ejecutar un archivo jar. ejecutennlo ¿qué pasa





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

* Portabilidad: Uno de los principales beneficios de los archivos JAR es la portabilidad. Dado que Java es un lenguaje multiplataforma, los archivos JAR son compatibles con diferentes sistemas operativos, siempre que tengan una máquina virtual de Java (JVM) instalada. Esto significa que puede distribuir su programa de manera efectiva en diferentes sistemas sin preocuparse por las diferencias en el sistema operativo.
* Aislamiento de dependencias: Los archivos JAR pueden contener todas las clases y dependencias necesarias para su programa, lo que facilita la gestión de las bibliotecas o paquetes externos que su proyecto requiere. Esto evita la necesidad de que los usuarios configuren manualmente sus dependencias, ya que todas están empaquetadas en el archivo JAR.
* Facilidad de distribución: La entrega de proyectos en forma de archivo JAR simplifica la distribución de software. Puede compartir su programa simplemente enviando un archivo JAR a los usuarios, y estos pueden ejecutarlo sin necesidad de instalar nada adicional. Esto es especialmente útil en entornos en los que la instalación de software adicional puede ser complicada o restringida.
* Seguridad: Los archivos JAR proporcionan un nivel de seguridad al aislar el código y los recursos del proyecto. Esto significa que el código contenido en un archivo JAR no tiene acceso directo al sistema del usuario, lo que reduce el riesgo de vulnerabilidades y amenazas de seguridad.
* Actualizaciones sencillas: Si necesita actualizar su programa, puede simplemente distribuir una nueva versión del archivo JAR. Los usuarios pueden reemplazar la versión anterior sin necesidad de desinstalar nada ni realizar complicados procedimientos de actualización.
* Consistencia: Los archivos JAR aseguran que los usuarios ejecuten la misma versión de su programa, lo que evita problemas de compatibilidad y facilita el soporte técnico.

# 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]

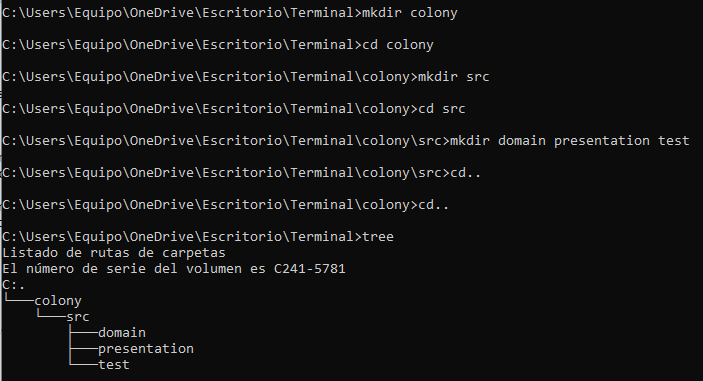
es de iniciar debemos repasar los comandos básicos del manejo de la consola.

1. Investiguen los comandos para moverse en la estructura de directorios: crear, borrar, listar su contenido y copiar o eliminar un archivo.
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

colony

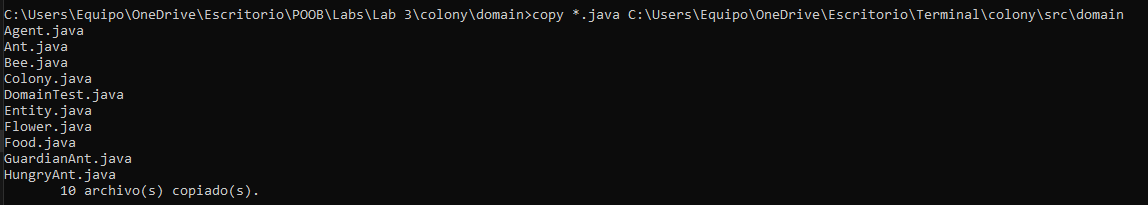
src

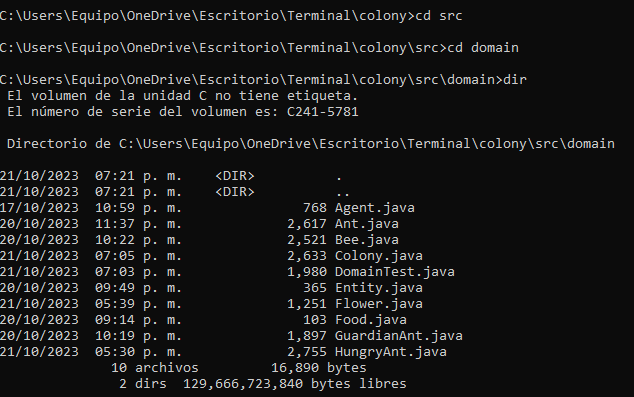
domain presentation test



1. 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

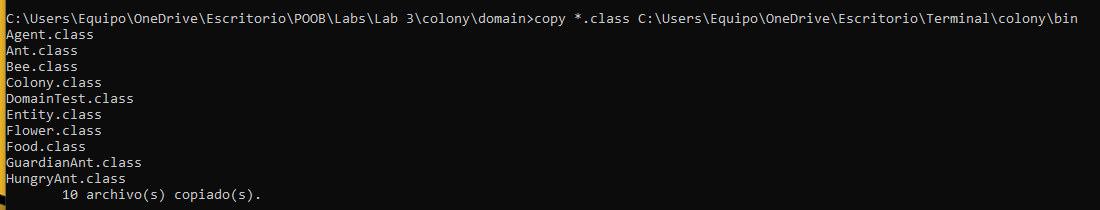
1. Investiguen los archivos que deben quedar en cada una de esas carpetas y la organización interna de cada una de ellas.

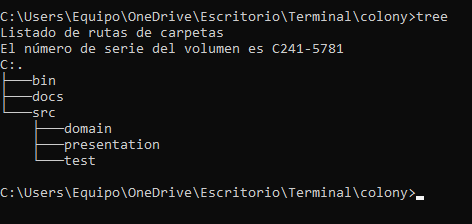
* Src(Fuente): En el directorio src se encuentran todos los archivos fuente (código fuente) de su proyecto Java. Este es el lugar donde se escriben las clases Java y otros recursos necesarios para el proyecto. Tipos de archivos: En el directorio src, encontrará principalmente archivos Java (.java). Además, puede contener otros recursos, como archivos de propiedades, archivos de configuración, plantillas, etc., que se utilizan en su proyecto.
* Bin: El directorio bin se utiliza para almacenar los archivos binarios generados a partir de la compilación de los archivos fuente en el directorio src. Estos archivos binarios suelen ser archivos de clase (.class) que contienen bytecode Java ejecutable. Tipos de archivos: Principalmente, encontrará archivos de clase (.class) generados por el compilador Java.
* Docs: El directorio docs se utiliza para almacenar la documentación relacionada con el proyecto. Esto puede incluir documentación de código, documentación de usuario, informes, diagramas, manuales, etc. Tipos de archivos: En docs, encontrará principalmente archivos de documentación en diversos formatos, como archivos de texto, HTML, PDF, imágenes, etc.

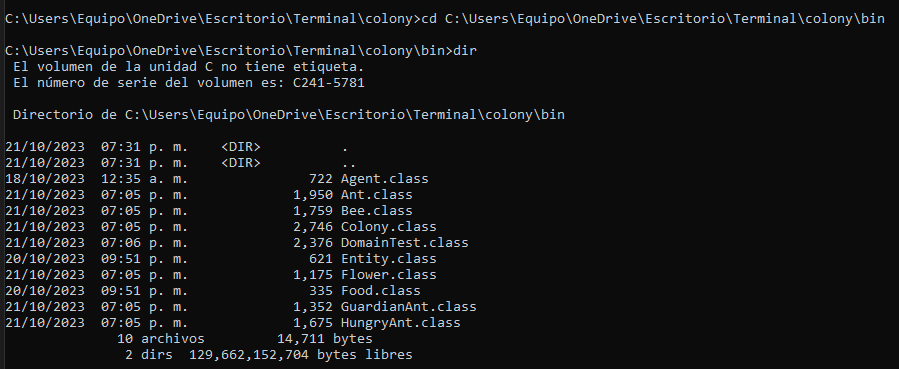
1. ¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué? Cópielos y consulte y capture el contenido del directorio que modificó.

R//: En un proyecto Java, el directorio bin se utiliza típicamente para almacenar los archivos binarios generados a partir de la compilación de los archivos fuente (ubicados en el directorio src). Los archivos que deben copiarse del proyecto original al directorio bin son principalmente los archivos de clase (.class) generados durante el proceso de compilación. La razón principal para esto es la separación entre el código fuente y los archivos binarios, lo cual es una buena práctica en desarrollo de software por varias razones:

* Compilación y ejecución separadas: Separar los archivos binarios del código fuente permite que el código fuente sea independiente de la plataforma y se compile en archivos de clase que contienen bytecode Java, que son los que se ejecutan en la máquina virtual de Java (JVM). Esto facilita la distribución del software, ya que los usuarios solo necesitan los archivos de clase (y no el código fuente) para ejecutar la aplicación.
* Seguridad y propiedad intelectual: El código fuente puede contener información sensible o propiedad intelectual que los desarrolladores desean proteger. Al separar los archivos de clase, es más difícil que terceros accedan al código fuente y lo modifiquen sin permiso.
* Mantenimiento y control de versiones: Mantener una separación entre el código fuente y los archivos binarios facilita el control de versiones y el mantenimiento del proyecto. Puede realizar cambios en el código fuente sin afectar los archivos binarios existentes y viceversa.
* Reducción del tamaño del proyecto: Los archivos de clase generados a partir del código fuente suelen ser más pequeños que los archivos fuente. Esto reduce el tamaño total del proyecto y facilita su distribución.







**Comandos de java** [En lab03.doc]

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

javac java javadoc jar

R//:

* javac:

Descripción: javac es el compilador de Java. Se utiliza para compilar archivos de código fuente Java (archivos .java) en archivos de clase Java (archivos .class) que contienen bytecode Java ejecutable.

Uso típico: Ejemplo de uso: javac MiPrograma.java. Esto compilará el archivo MiPrograma.java en un archivo de clase MiPrograma.class.

* java:

Descripción: java es el intérprete de Java. Se utiliza para ejecutar aplicaciones Java compiladas. Toma un archivo de clase como entrada y lo carga en la máquina virtual de Java (JVM) para su ejecución.

Uso típico: Ejemplo de uso: java MiPrograma. Esto ejecutará la clase MiPrograma contenida en el archivo de clase MiPrograma.class.

* javadoc:

Descripción: javadoc es una herramienta de documentación en Java que genera documentación a partir de comentarios en el código fuente. Los comentarios deben seguir el formato de documentación de Java (JavaDoc) y pueden incluir descripciones de clases, métodos, campos, parámetros, etc.

Uso típico: Ejemplo de uso: javadoc MiClase.java. Esto generará documentación en formato HTML basada en los comentarios del archivo MiClase.java.

* jar:

Descripción: jar es una herramienta que se utiliza para crear, manipular y gestionar archivos JAR (Java Archive). Los archivos JAR son archivos comprimidos que contienen archivos de clase, recursos y metadatos de un proyecto Java.

Uso típico: Ejemplo de uso:

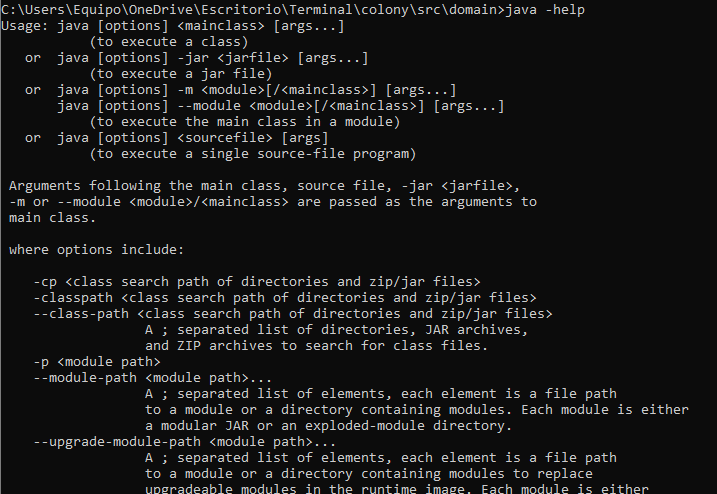
Para crear un archivo JAR: jar cvf MiArchivo.jar MiClase.class OtroArchivo.class.

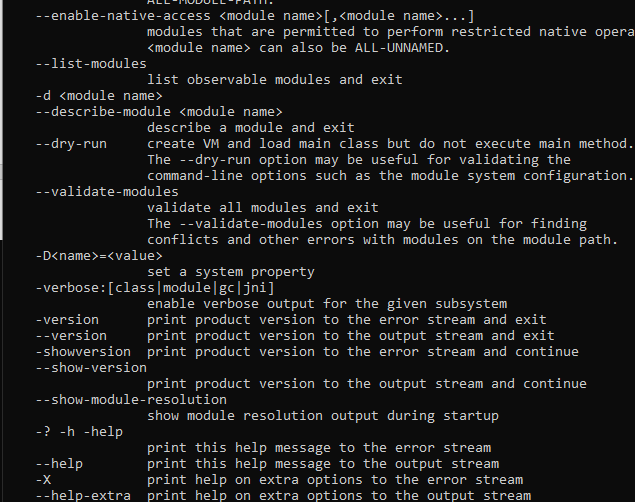
Para extraer archivos de un archivo JAR: jar xvf MiArchivo.jar.

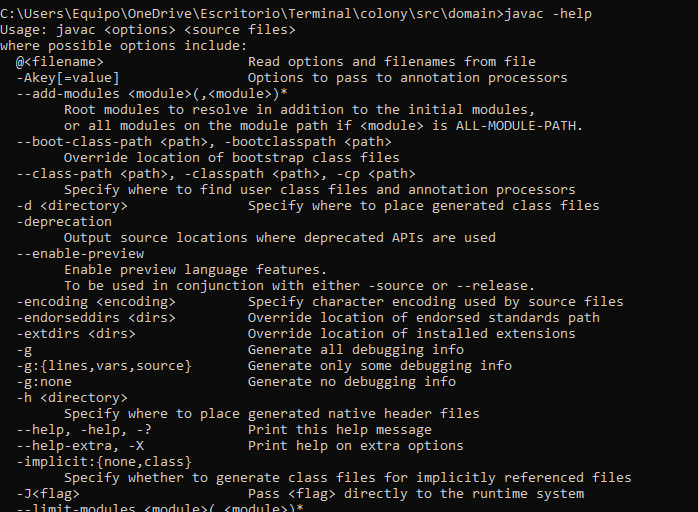
Para visualizar el contenido de un archivo JAR: jar tf MiArchivo.jar.

1. Cree una sesión de consola y consulte en línea las opciones de los comandos java y

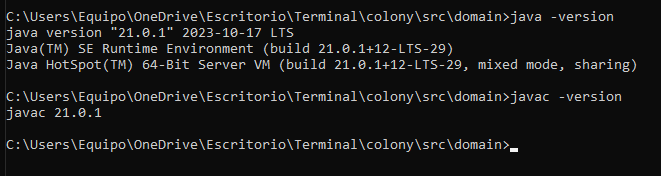
javac. Capture las pallas.







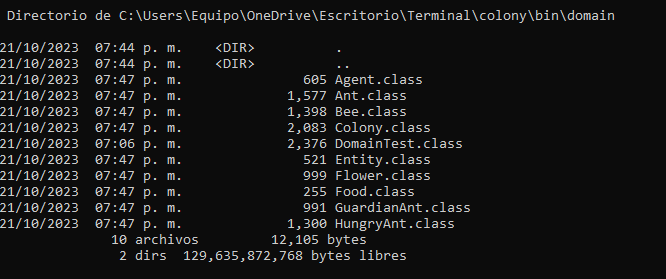
1. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.



**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.





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

Ahora aparecen Subdirectorios dentro de bin que son domain y presentation, dentro de cada uno salen los .class correspondientes.

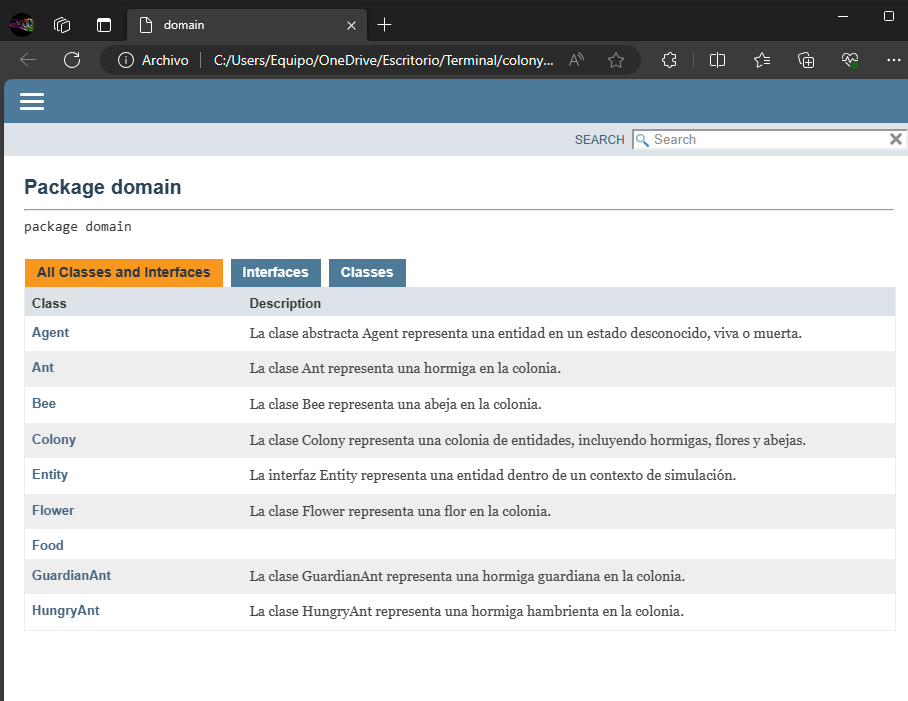
**Documentando** [En lab03.doc]

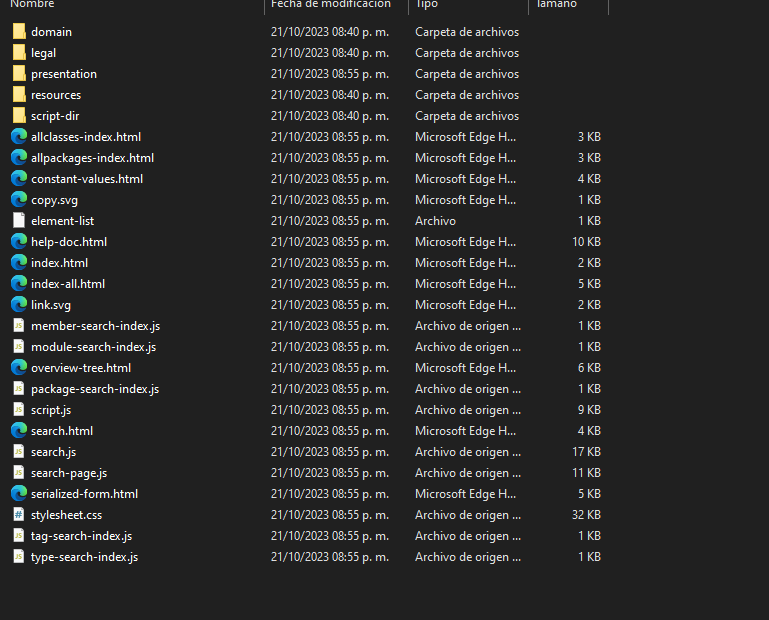
1. Utilizando el comando javadoc, desde el directorio raiz, genere la documentación (API) en formato html, en este directorio. ¿cuál es el comando completo para generar esta documentación?



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

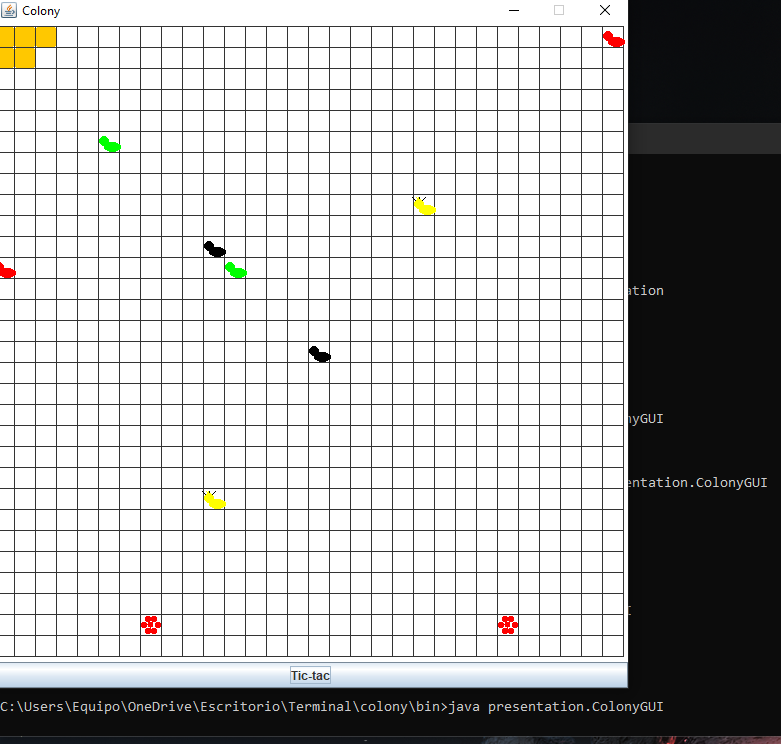
Hay que abrir el index.html





**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](https://stackoverflow.com/questions/2235276/how-to-run-junit-test-cases-from-the-command-line)

3.

4. Pegue en su documento el resultado de las pruebas

**Empaquetando** [En lab03.doc]

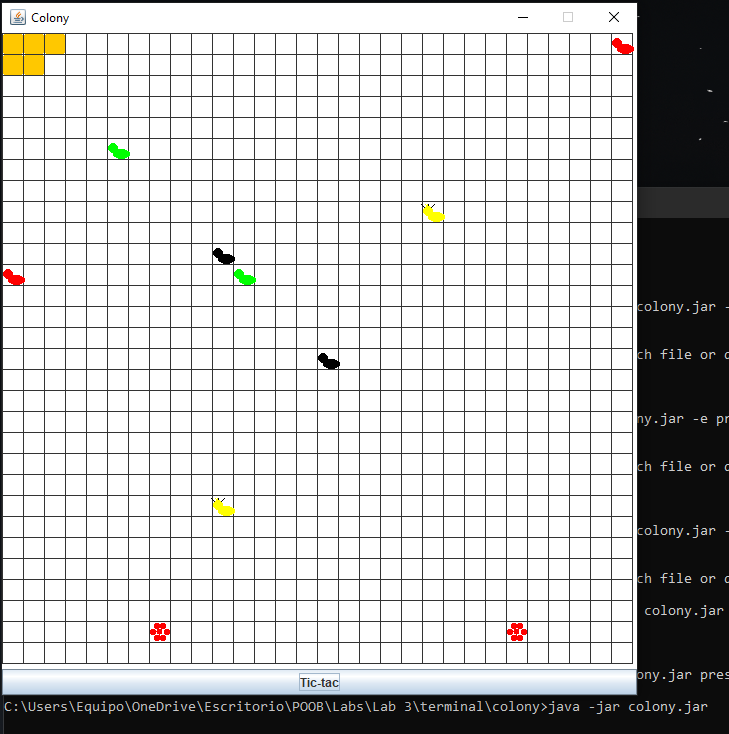
1. Consulte como utilizar desde consola el comando jar para empaquetar su programa entregable en un archivo .jar, que contenga los archivos bytecode necesarios (no las fuentes ni las clases de prueba), y que se pueda ejecutar al instalarlo en cualquier directorio, con solo tener la máquina virtual de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

¿Cómo empaquetó jar ?

* Compila tus fuentes: Asegúrate de que todas tus clases estén compiladas y listas para ser empaquetadas en el archivo JAR. Esto se hace normalmente con el comando javac.
* Organiza tus clases: Asegúrate de que tus clases se encuentren en la estructura de paquetes correcta en tu sistema de archivos. Por ejemplo, si tienes un paquete llamado presentation, las clases de ese paquete deberían estar en una carpeta llamada presentation.
* Abre la línea de comandos: Abre una ventana de línea de comandos en la ubicación de tu proyecto. Esto es importante para que puedas ejecutar los comandos necesarios.
* Utiliza el comando jar para empaquetar:



1. ¿Cómo se ejecuta el proyecto empaquetado?



# RETROSPECTIVA

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

R//:

1. ¿Cuál es el estado actual de laboratorio? ¿Por qué? (Para cada método incluya su estado)
2. Considerando las prácticas XP del laboratorio de hoy ¿por qué consideran que son importe?
3. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?
4. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?