**Conociendo**

1. En el directorio descarguen los archivos contenidos en bodyTic.zip y bodyTic.asta. Revisen el código del programa a) ¿Cuántos paquetes tiene? b) ¿Cuántas clases tiene en total? c) ¿Cuál es la clase ejecutiva? ¿Por qué?
2. El programa tiene dos paquetes(aplicación y presentación).
3. Son 5 clases.
4. En el paquete presentación la clase BodyTicGUI es la clase ejecutiva, ya que es la que ejecuta el programa.
5. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?

* Entre, inicien, paren, decidan y salgan son las funcionalidades que ofrece. Actualmente no hace nada porque los metodos estan vacios.

**Arquitectura general.**

1. Consulten el significado de las palabras package e import de java. Explique su uso con un ejemplo de este programa.

package: es un directorio, el nombre del directorio es el mismo del paquete que contiene esas clases.  
  
import: se llama al paquete que se quiere usar desde otras librerias en una declaración del import.

package presentación;

import aplicación;

1. ¿Qué es un paquete? ¿Para qué sirve? Inicien el diseño con un diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos.

* Un Paquete en Java es un contenedor de clases que permite agrupar las distintas partes de un programa y que por lo general tiene una funcionalidad y elementos comunes, definiendo la ubicación de dichas clases en un directorio de estructura jerárquica.
* Agrupamiento de clases con características comunes.
* Reutilización de código al promover principios de programación orientada a objetos como la encapsulación y modularidad.
* Mayor seguridad al existir niveles de acceso.
* Evita la colisión de clases que tengan el mismo nombre. Pueden existir clases con el mismo nombre siempre y cuando su fully qualified class name sean únicos.
* Mantenibilidad de código. Si un paquete se enfoca en la agrupación de clases con características comunes, el cambio en la funcionalidad se limita a las clases contenidas en dicho paquete, además, si es un paquete grande soporta la reusabilidad, si por el contrario es pequeño soporta su mantenibilidad.

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

* El directorio contiene los archivos del paquete.

**Arquitectura detallada.**

1. ¿Qué observaciones haría al respecto?

* Podemos observar que en ciertas clases existen atributos de tipo protected.

**Ciclo 1. Iniciando con los deportistas normales.**

1. Estudie la clase Salon ¿Qué tipo de colección usa para albergar los elementos? ¿Puede recibir deportistas? ¿Por qué?

* Utiliza un ArrayList de tipo privado para albergar los elementos.
* No, no puede recibir deportistas. Porque no tiene ningún atributo de tipo deportista.

1. Estudie el código de la clase Deportista, ¿qué otros componentes software la definen? ¿cómo?

* La define *Persona* y *EnSalon*, extiende de la clase *Persona* e implementa *EnSalon*. Entonces estas clases definen algunos comportamientos y atributos de esta clase.

1. Por ser una Persona ¿qué saben hacer? ¿qué no puede hacer distinto? ¿qué deben aprender a hacer? Justifique su respuesta.

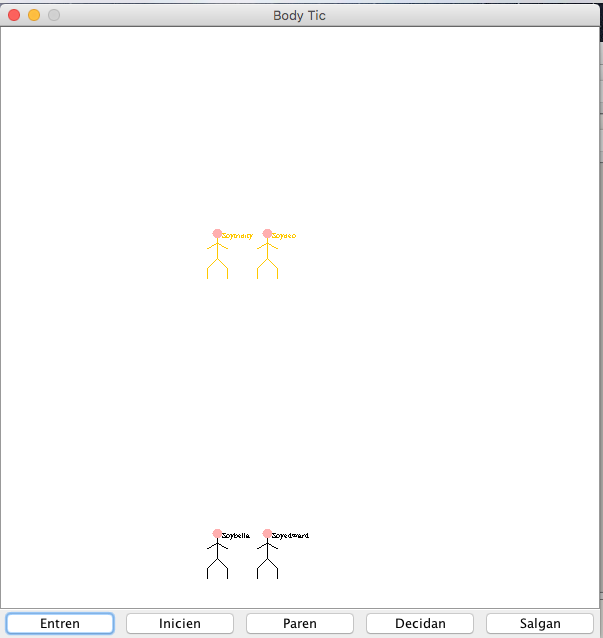
* Sabe su color, muevaBrazo, muevaPierna, getPosicionBrazo, getPosicionPierna, muévase, getPosicionX, getPosicionY.
* No debe aprender a hacer nada, porque los metodos ya están implementados.

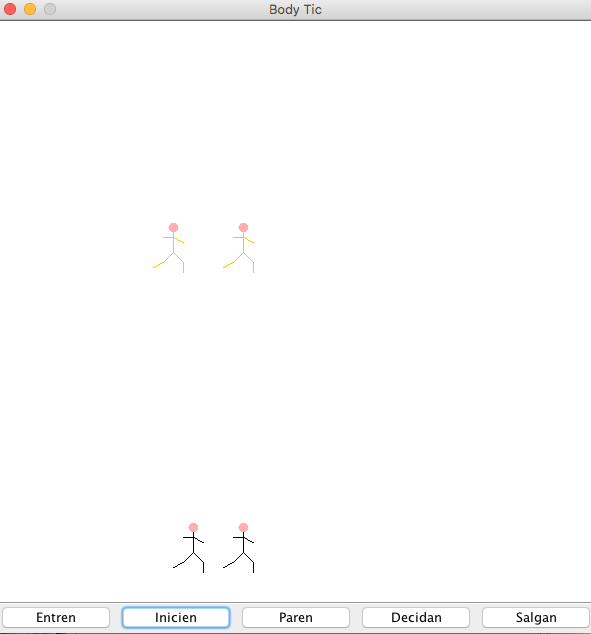
**Ciclo 2. Incluyendo a los deportistas avanzados**

1. ¿Cuáles son las adiciones necesarias en el diseño? ¿y los cambios? ¡Hágalos! ¿cuáles métodos se sobre-escriben (overriding)? Ahora escriba el código correspondiente al deportista avanzado.

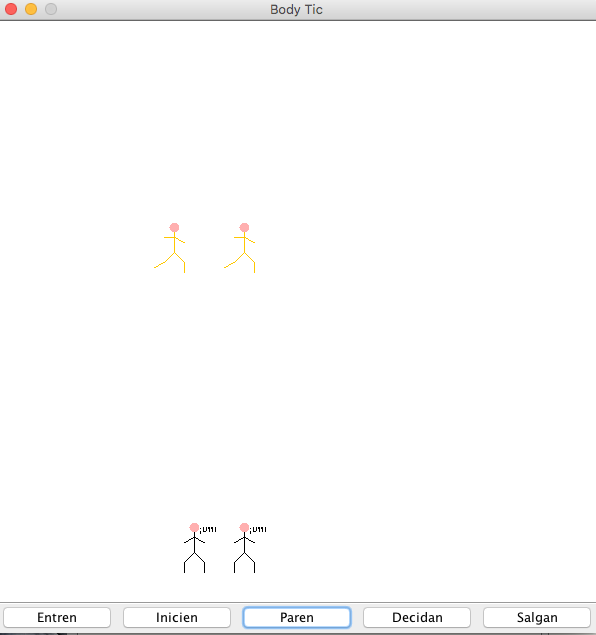
* Se creará una nueva clase, la cual se llamará DeportistaAvanzado esta es herencia de Deportista. Los métodos que se van a sobreescribir son inicie(), pare() y decida().

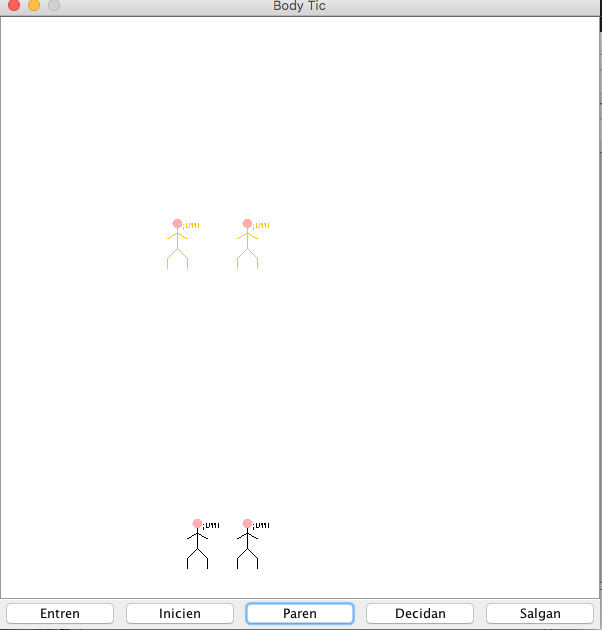
1. Adicionen una pareja de deportistas avanzados en la fila 3, llámense neo y trinity, Ejecute el programa y pídales que entren, inicien, decidan y paren. Capturan pantallas significativas. ¿Qué pasa?

* Entren: Tanto como los Deportistas y los DeportistasAvanzado entran al salon.
* Inicien: Comienzan a hacer ejercicio todos los deportistas. Con la diferencia de que los DeportistasAvanzados se mueven más.

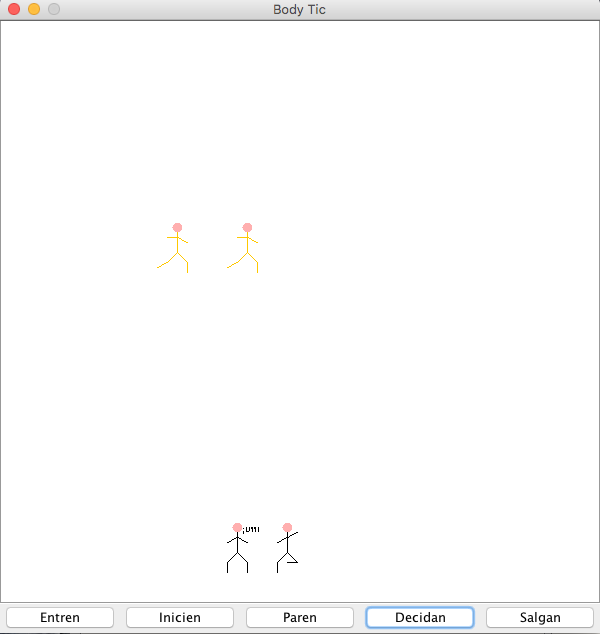


* Paren: Todos los deportistas paran de hacer ejercicio y hablan.





* Decidan: Los DeportistasAvanzados siempre bailan mientras que los Deportistas hacen algo aleatorio.

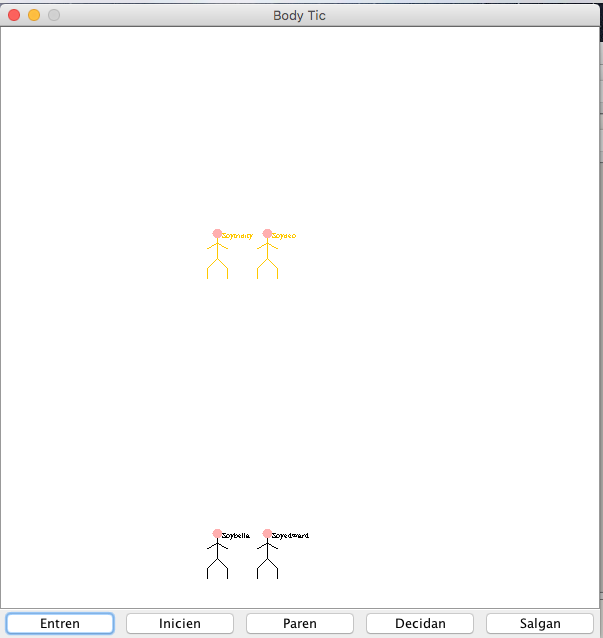


1. Ahora, los avanzados quieren sorprender con su resistencia; es decir, sólo paran muy pocas veces. ¿Qué modificaría para lograr este comportamiento? ¡Hágalo!

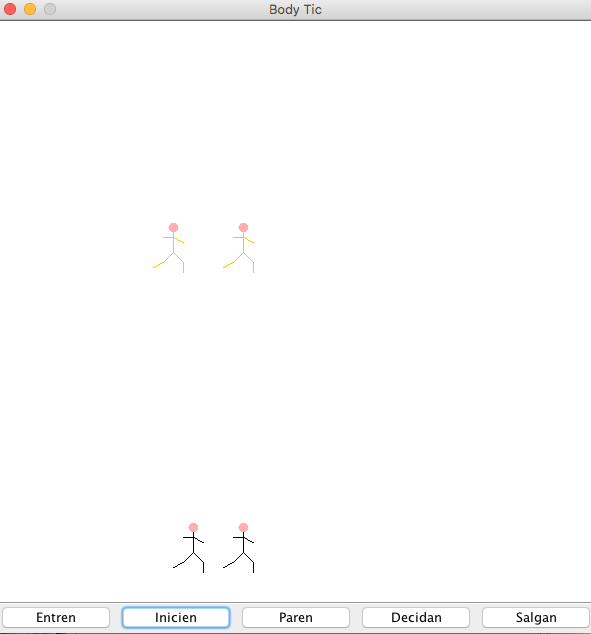
* Se modifica pare(), creado un parámetro entero. Ellos pueden parar un cierto número de veces, luego de pasar este límite de descansos siguen bailando sin parar.

1. Nuevamente ejecute el programa y pídales a todos que entren, inicien, decidan y paren. Capturen una pantalla significativa. ¿Qué pasa?

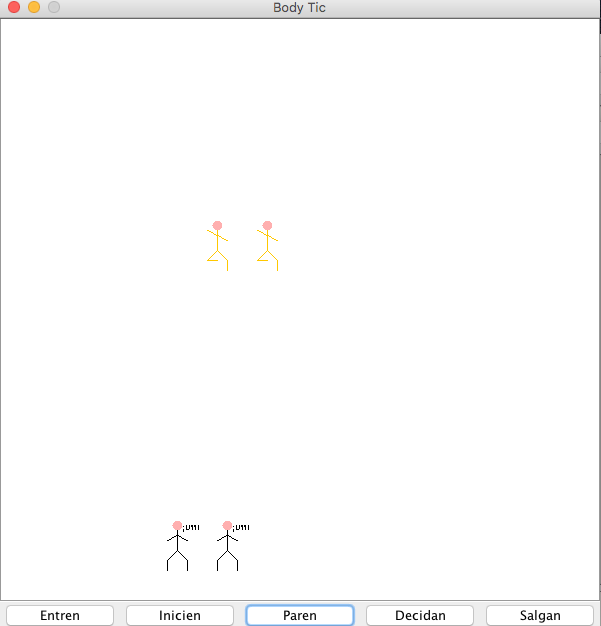
* Entren: Tanto como los Deportistas y los DeportistasAvanzado entran al salon.



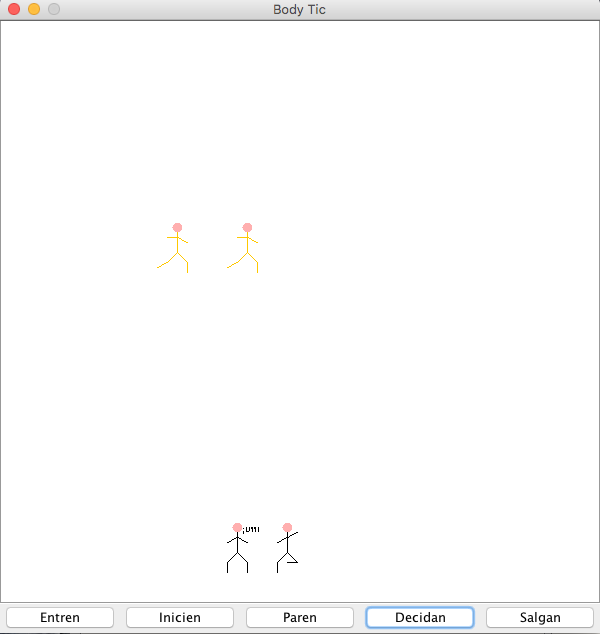
* Inicien: Comienzan a hacer ejercicio todos los deportistas. Con la diferencia de que los DeportistasAvanzados se mueven más.



* Paren: Después de 5 veces que paren, los DeportistasAvanzados siguen haciendo ejercicio y no paran.



* Decidan: Los DeportistasAvanzados siempre bailan mientras que los Deportistas hacen algo aleatorio.

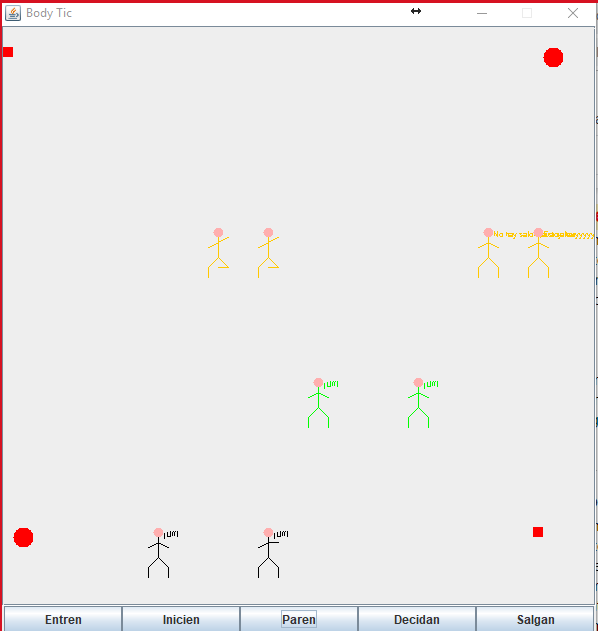


**Ciclo 2. Incluyendo a los deportistas habladores.**

1. ¿Cuáles son las extensiones necesarias en el diseño? ¿y los cambios? ¡Hágalos! ¿cuáles métodos se sobre-escriben (overriding)?

Se realiza una herencia sobre la clase Deportista, se cambia el comportamiento de inicien, se sobre escribe paren cambiando la variable palabras, deciden se deja por defecto.

1. Implemente al deportista hablador.
2. Adicionen una pareja de deportistas avanzados en la fila 3, llámenlas han y leila, Ejecute el programa y pídales que entren, inicien, decidan y paren. Capturen pantallas significativas. ¿Qué pasa?



* Los diferentes deportistas realizan una acción diferente al momento de decidir, pero luego en parar unos realizan una acción como hablar y otros como parar completamente. Claro está que cada deportista realiza una cosa diferente.

**Ciclo 3. Adicionando bolas de pilates.**

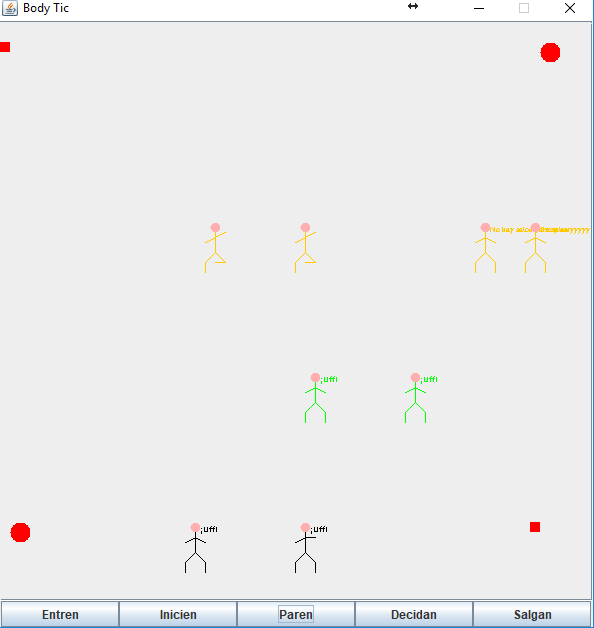
1. ¿Cuáles son las adiciones necesarias en el diseño? ¿y los cambios? ¡Hágalos! ¿cuáles métodos se sobre-escriben (overriding)? Ahora escriba el código correspondiente a las bolas de pilate.

* Se crea un constructor en bola para asignarle sus respectivos atributos. En cuanto a los métodos sobreescritos fueron pare(),inicie() ya aue asignamos distintos colores a la bola cuando para y cuando inicia.

1. Para aceptar este elemento , ¿debe cambiar en el código del Salon en algo? ¿por qué?

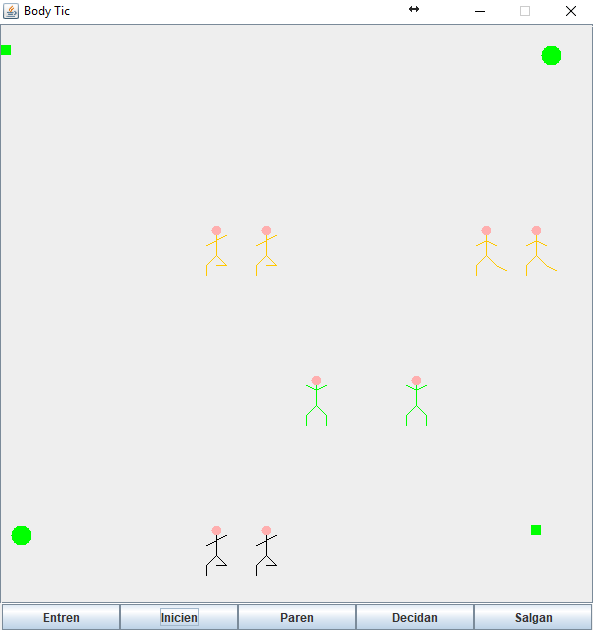
* No se debe cambiar nada, solamente tenemos que crear las dos bolas nuevas en el respectivo salón.

1. Adicionen dos bolas cerca en las esquinas del Salon, llámenlas suroeste y noreste, Escriba la prueba correspondiente.
2. Nuevamente ejecute el programa y pídales a todos que entren, inicien, decidan y paren. Capturen una pantalla significativa. ¿Qué pasa?

* Deciden: Deciden aleatoriamente   
    
  

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

1. Propongan, describen e Implementen un nuevo tipo de deportistas.   
   - El nuevo deportista es el DeportistaPrincipiante, el cual se cansa muy rapido.
2. Incluyan una pareja de ellos con el nombre de ustedes. ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes.

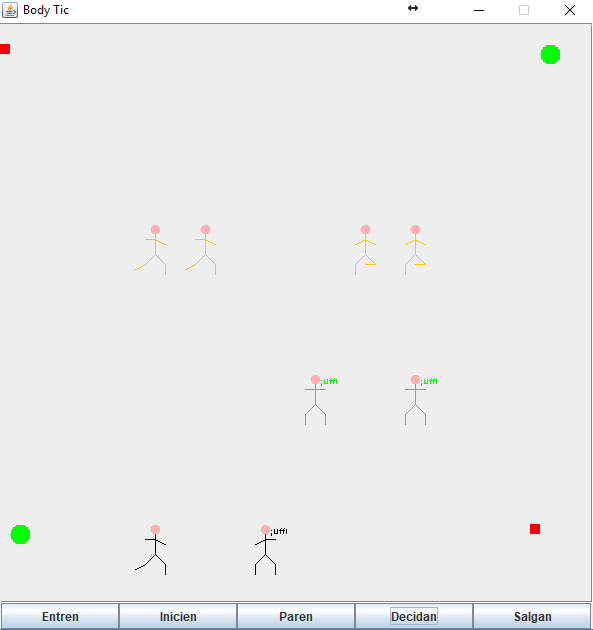


**Ciclo 5. Nuevo elemento: Proponiendo y diseñando**

1. Propongan, describan e Implementen un nuevo tipo de elemento

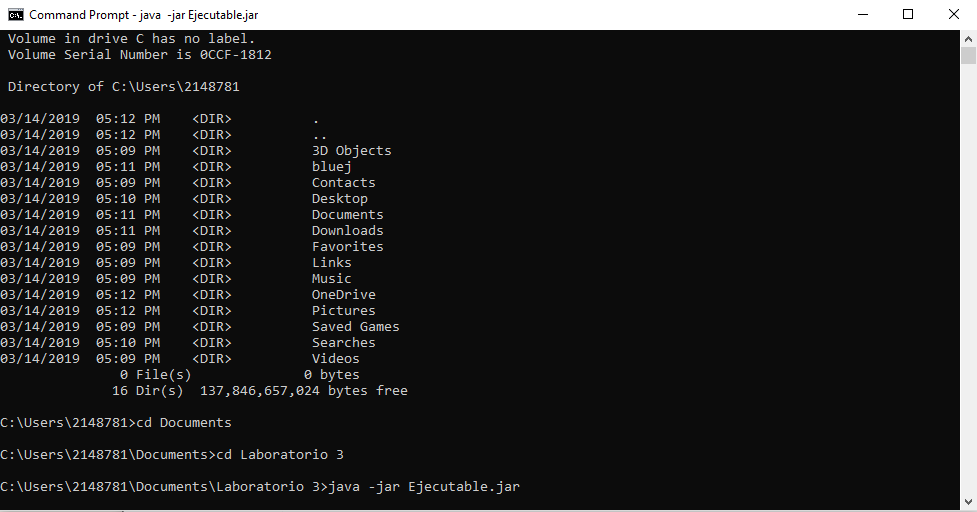
* EL nuevo elemento es una caja de salto, el cual estara ubicado en las esquinas opuestas de las bolas.

1. Incluyan un par de ellos con el nombres semánticos. ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes



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

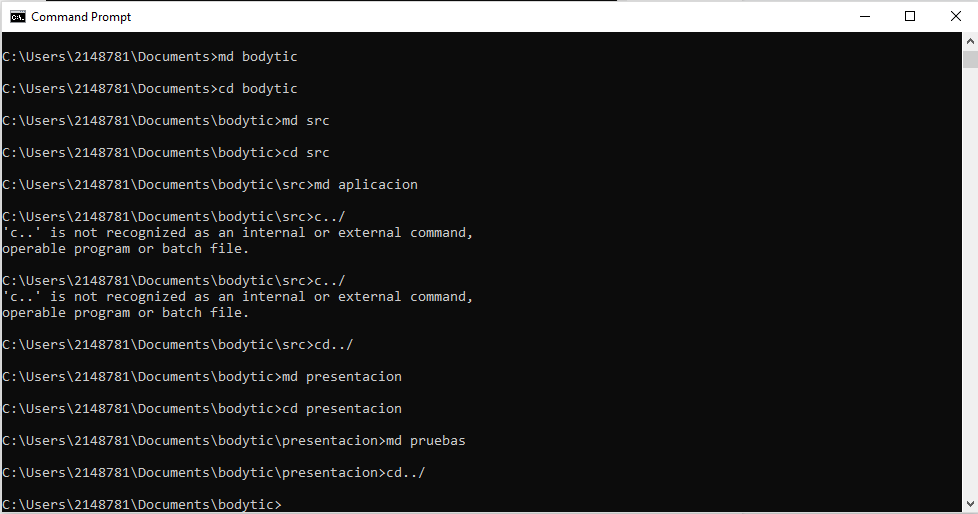
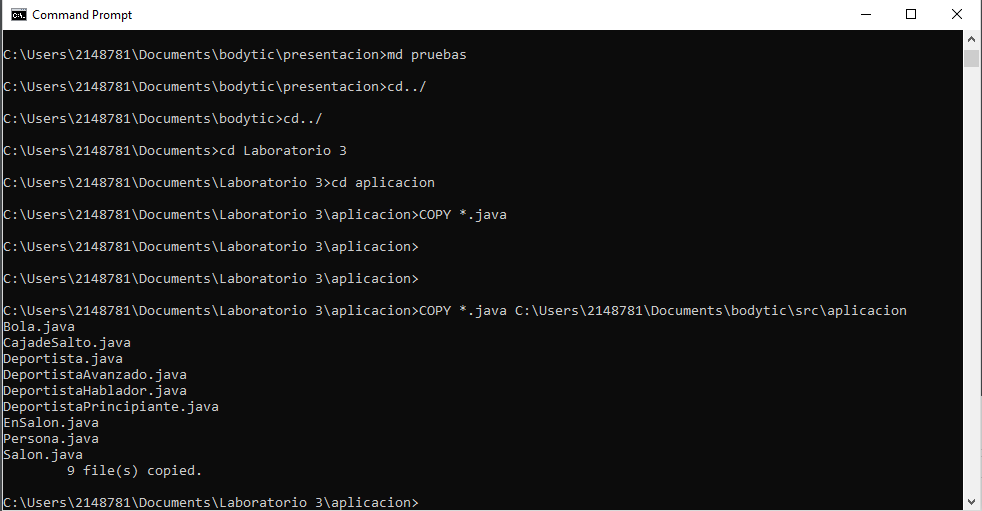
2. Consulten el comando java para ejecutar un archivo jar. ejecútenlo ¿qué pasa?

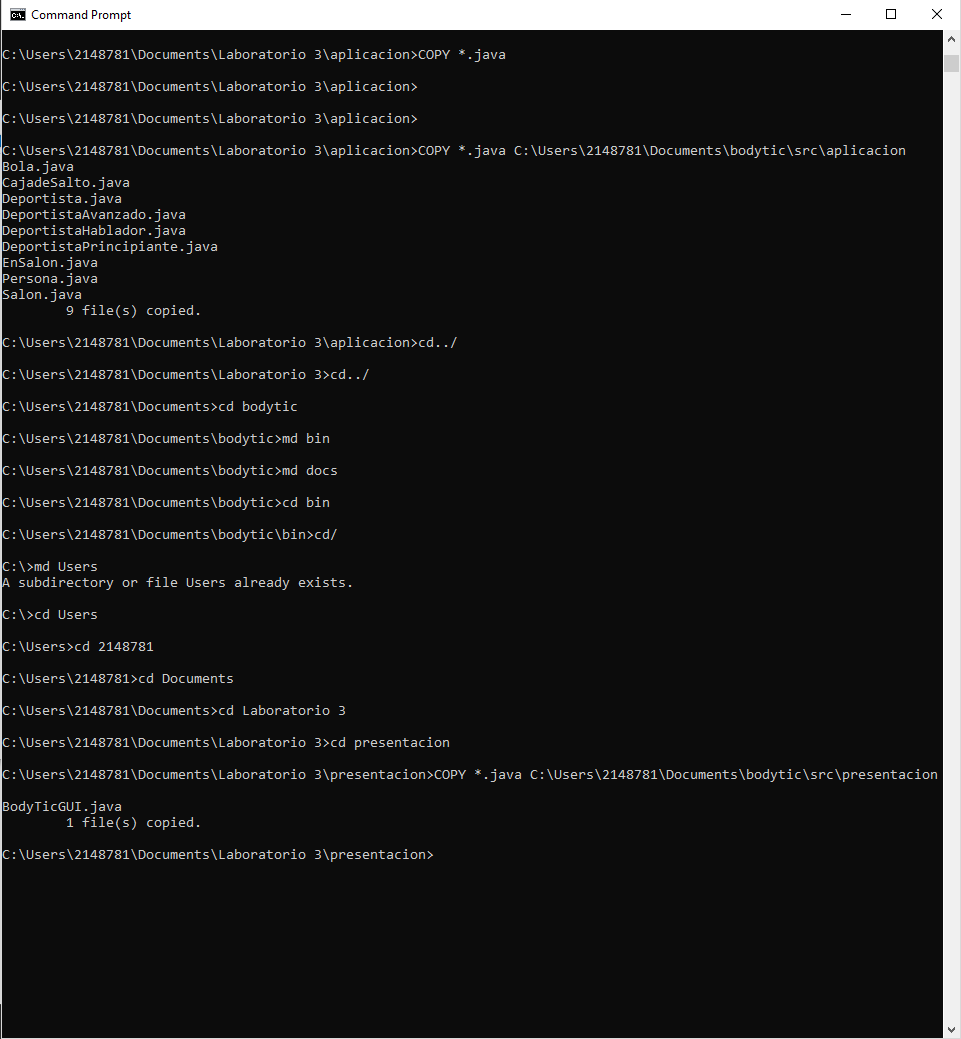
* Solo se ejecuta el paquete presentación ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente
* El usuario puede ejecutar el programa sin necesidad de tener las fuentes.  
  

**Comandos básicos del sistema operativo**   
Antes 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.

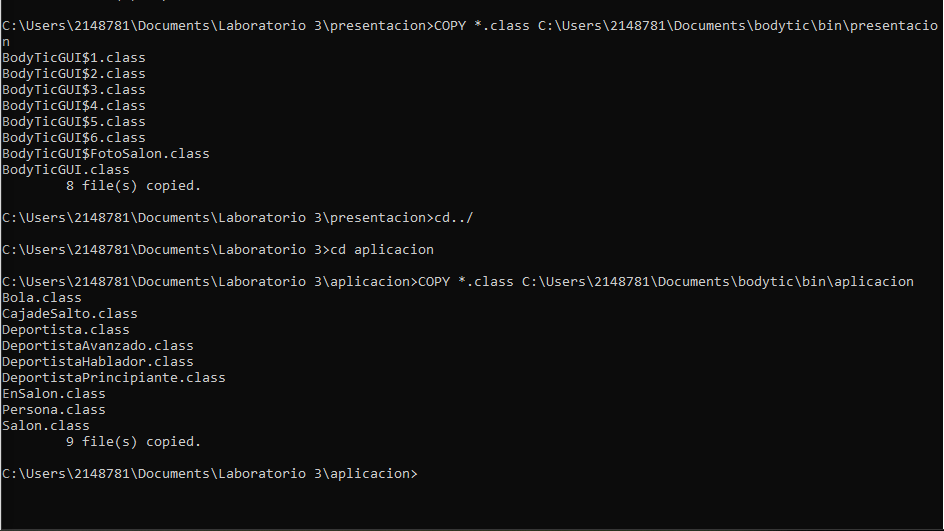
* crear : md nombre del directorio.
* borrar: rd nombre de la carpeta.
* listar su contenido: dir
* copiar= xcopy directorio1 directorio2
* Eliminar: del nombre del archivo

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



**Estructura de proyectos java**

1. Investiguen los archivos que deben quedar en cada una de esas carpetas y la organización interna de cada una de ellas.
2. ¿Qué archivos debería copiar del proyecto original al directorio bin? ¿Por qué? Cópialos y consulte y Capturen el contenido del directorio que modificó.

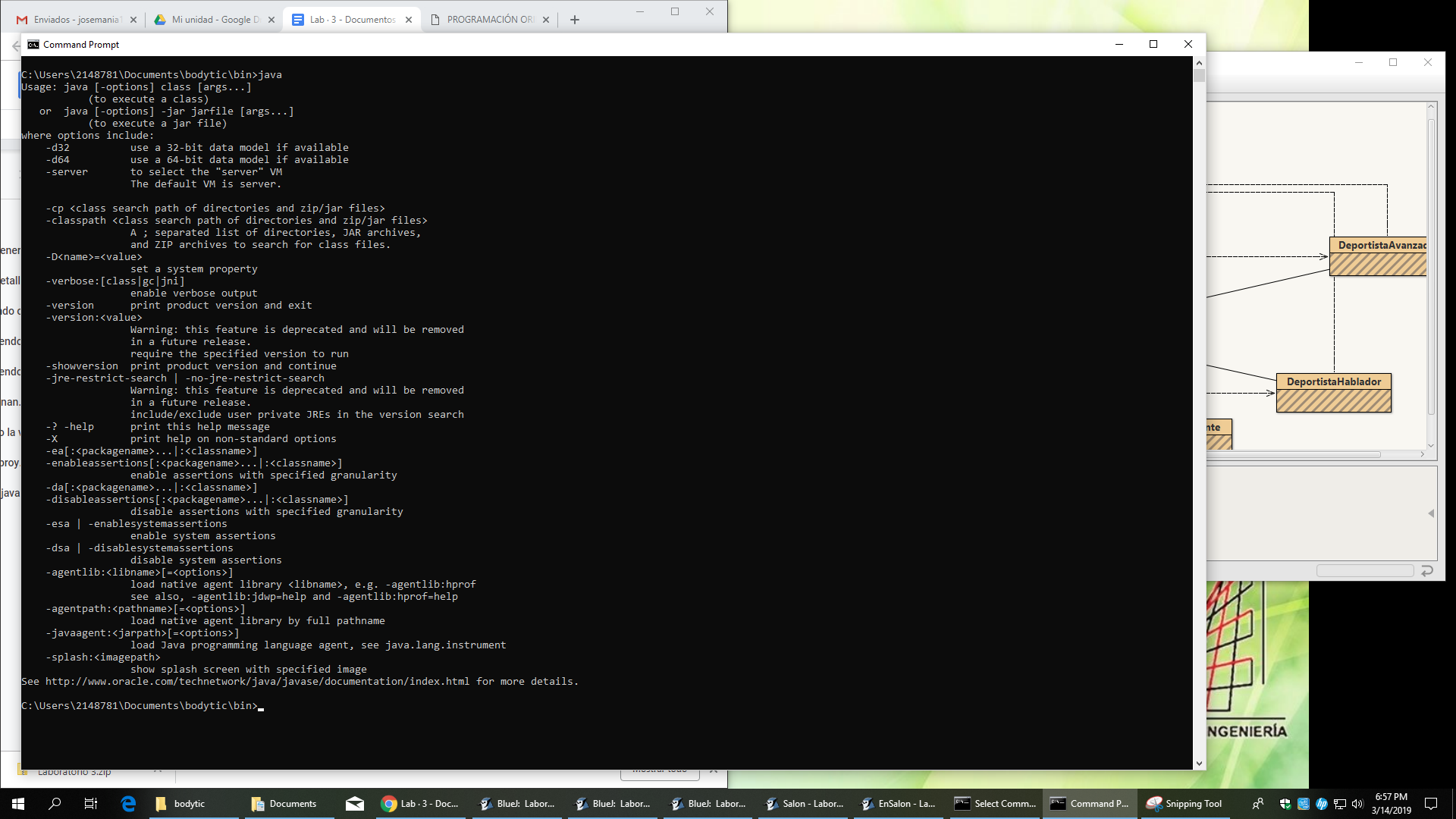


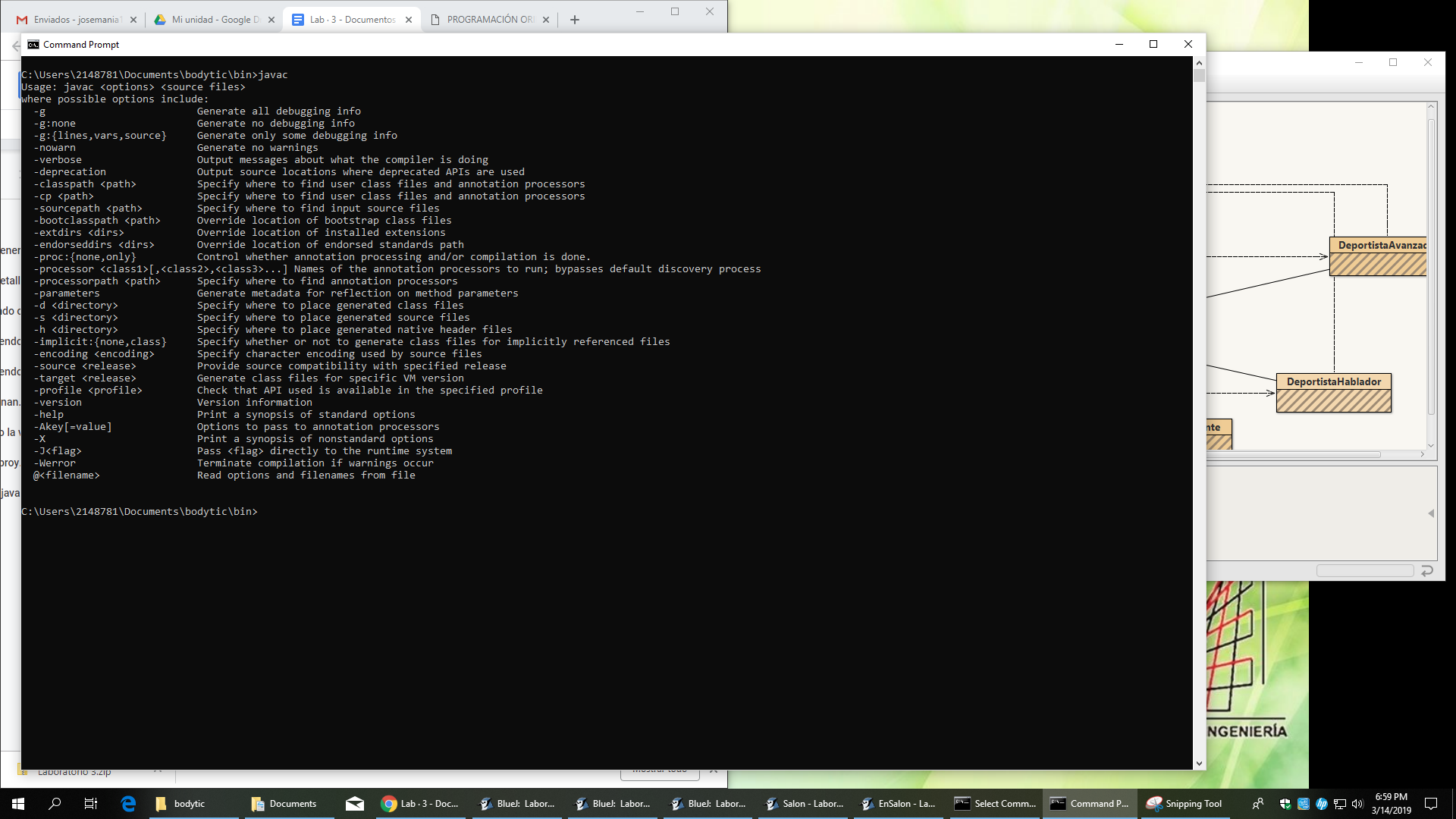
* Deben ir los ejecutables en bin, en src deben ir las fuentes y en docs van la documentación.

**Comandos de java**

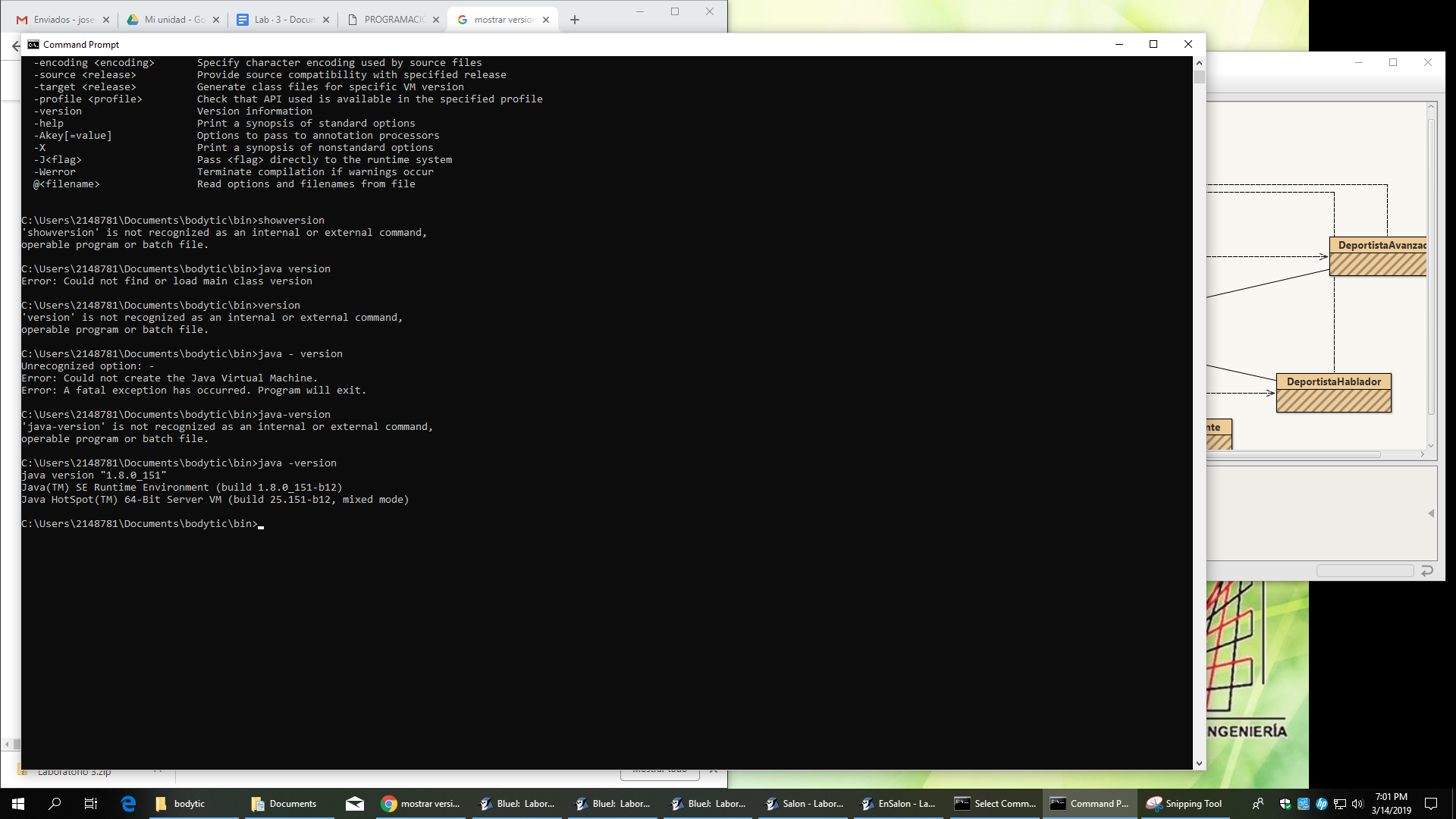
1. Consulten para qué sirven cada uno de los siguientes comandos:  
   <https://misapuntesdeprogramacion.wordpress.com/2013/01/23/linea-de-comandos/>

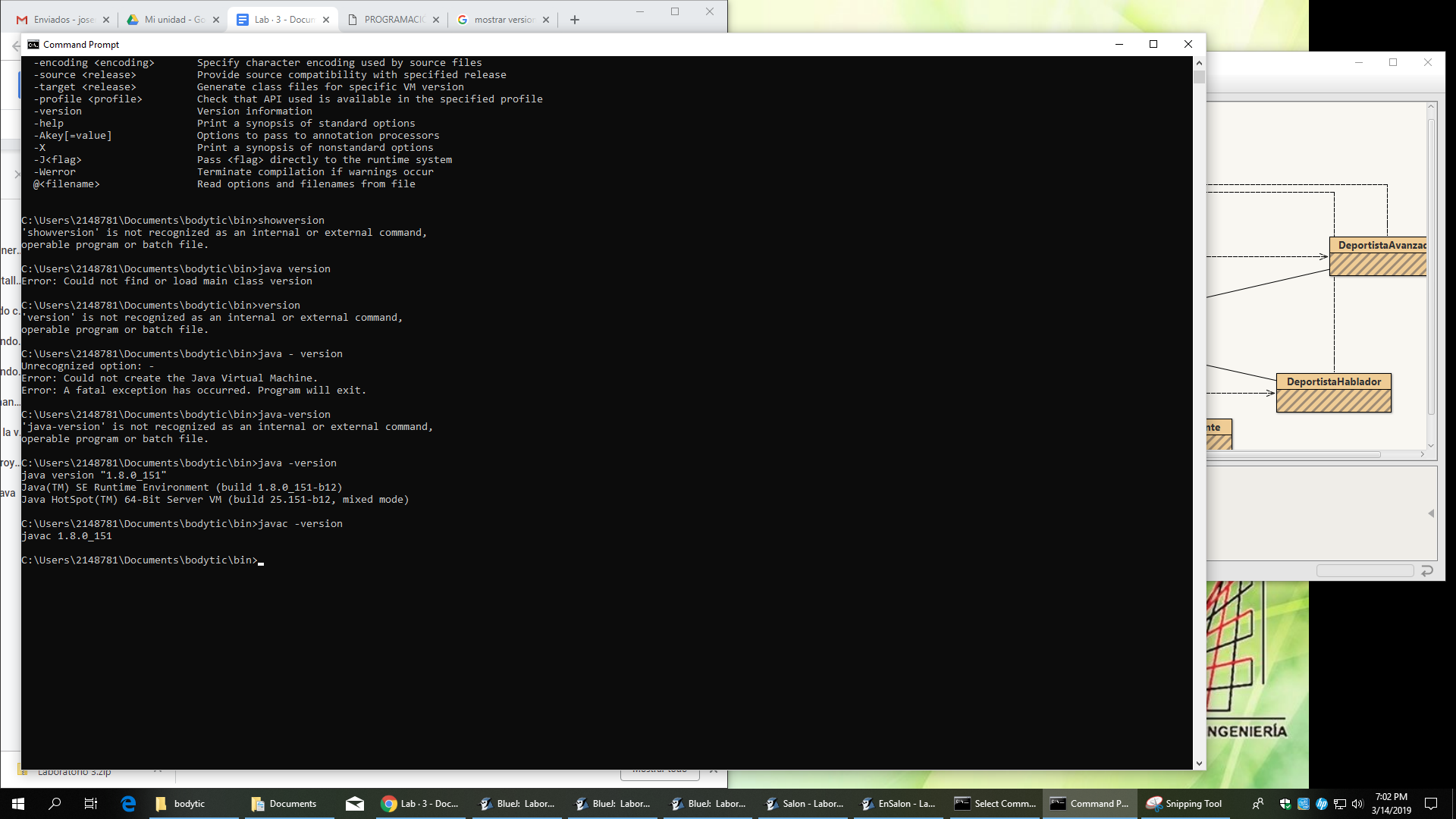
* javac: es el compilador **de** java, un programa ejecutable que nos permitirá compilar nuestro código fuente.
* java: compilar y ejecutar o correr fácilmente programas en **Java** desde ventana de comandos (consola o **cmd**) sin configurar las variables de entorno.
* javadoc:genera documentación API en formato HTML para el paquete especificado o para los archivos fuentes individuales en Java en la línea de comandos. Javadoc genera un fichero *.html* por cada fichero *.java* y paque que encuentra.
* jar: Comprimir y empaquetar ficheros en un archivo JAR ejecutable.

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



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





**Compilando**

1. Utilizando el comando javac, desde el directorio raiz (desde bodyTic con una sola 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

javac -d bin src\aplicacion\\*.java src\presentacion\\*.java

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

* Aparece en bin una nueva carpeta aplicacion, con los documentos tipo .Class

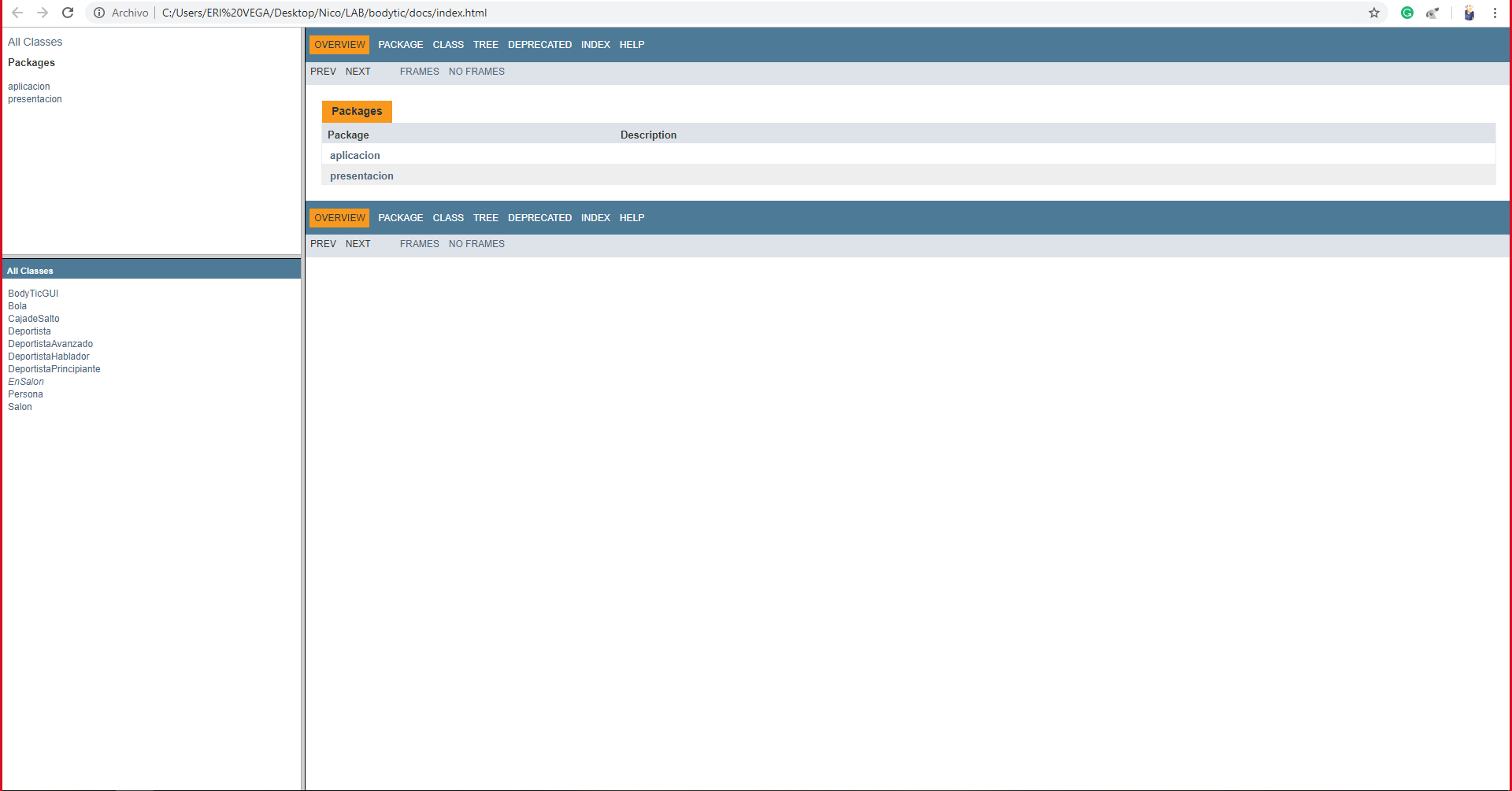
**Documentando**

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 docs src\aplicacion\\*.java src\presentacion\\*.java

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

* Con index.html podemos ver la documentacion



**Ejecutando**

1. Empleando el comando java, desde el directorio raiz, ejecuten el programa. ¿Cómo utilizó este comando?

* java -cp bin presentacion.BodyTicGUI
* Se utilizó para ejecutar la clase BodyTicGUI

**Probando**

1. Adicionen 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?

javac -d bin -cp junit-4.8.jar src\aplicacion\\*.java src\pruebas\\*.java

1. Ejecuten desde consola las pruebas . ¿Cómo utilizó este comando?. Puede ver ejemplos de cómo ejecutar el“test runner”en: http://junit.sourceforge.net/doc/cookbook/cookbook.htm

java -cp junit-4.8.jar;bin org.junit.runner.JUnitCore pruebas.Test1

1. Pegue en su documento el resultado de las pruebas

**Empaquetando**

1. Consulten 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 ?

jar -cfe bodytic.jar presentacion.BodyTicGUI -C bin

1. ¿Cómo se ejecuta el proyecto empaquetado?  
     
   java -jar bodytic.jar

**Retrospectiva**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?  
   26 horas
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?  
   90%, ya que falta compilar las pruebas por consola y generar el archivo .jar por consola
3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?  
   Las practicas XP fue la programacion a par, ya que es una manera efectiva y buena de realizar trabajos en grupo.
4. ¿Cuál considera fue el mayor logro? ¿Por qué?  
   Aprender a programar ciertas cosas desde consola.
5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?  
   No saber nada sobre la consola, pero investigando y consultando se resolvieron dudas.