PROGRAMACIÓN ORIENTADA A OBJETOS
Herencia e interfaces
ADEMAS Java desde consola
2020-1
Laboratorio 3/6

OBJETIVOS
Desarrollar competencias básicas para:

1. Aprovechar los mecanismos de la herencia y el uso de interfaces. 2. Organizar las fuentes en paquetes.
3. Extender una aplicación cumpliendo especificaciones de diseño, estándares y verificando su corrección.
4. Usar la utilidad jar de java para entregar una aplicación.
5. Vivenciar las prácticas XP : Use collective ownership.
   Only one pair integrates code at a time.

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. Ⓟ En el foro de entrega deben indicar el estado de avance de su laboratorio y los problemas pendientes por resolver.
Ⓟ 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

UEl 6 de abril del 1909 Robert E. Peary y un equipo formado por esquimales (Ooqueah, Ootah, Henson, Egingwah y Seeglo) afirmaron ser los primeros a llegar al polo Norte geográfico. En esta aplicación un equipo de nativos (todos esquimales) se embarcarán en una expedición para repetir la hazaña mientras son filmados desde un satelite. Para que todo salga bien ellos deben estar atentos a seguir las órdenes clásicas de rodaje – acción y corten – y también pueden improvisar.

Conociendo [En lab03.doc y artico.asta]

1. En el directorio descarguen los archivos contenidos en artico.zip. Revisen el código de la aplicación a) ¿Cuántos paquetes tiene? b) ¿Cuántas clases tiene? c) ¿Cuál es el propósito del paquete de presentación? d) ¿Cuál es el propósito del paquete de dominio?

a) Cuenta con dos paquetes y son: Dominio y presentación

b) Cuenta con una Interfaz(EnArtico), una clase abstracta(persona) y cuatro clases concretas(Artico, Esquimal, Iglu, ArticoGUI).

c) El propósito del paquete presentación es encargarse de la parte visual del aplicativo

d) El propósito del paquete dominio es encargarse del funcionamiento interno del aplicativo

2. **En este laboratorio vamos a ejecutar la aplicación, no a solicitar servicios a objetos.** ¿Qué método se usa para ejecutar una aplicación java? ¿Qué clase tiene ese método?

a) El metodo usado es: main(String[] args)
b) La clase ArticoGUI

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

a) Las funcionalidades son: Accion, improvisen, corten y camara rapida
b) No hace nada actualmente, porque no están codificados

Arquitectura general. [En lab03.doc y artico.asta]

1. Consulten el significado de las palabras package e import de java. ¿Qué es un paquete?

¿Para qué sirve? Explique su uso en este programa.

a) 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.
b) Los paquetes son el mecanismo que usa Java para facilitar la modularidad del código.
c) Su uso en este programa es con los paquetes, dominio y presentación, donde este último importa el paquete dominio que es el encargado del funcionamiento interno del aplicativo, así este podrá cumplir su función de hacerlo visual.

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

a) Contamos con el directorio Ártico, y este contiene los subdirectorios dominio y presentación, los cuales se componen ya sea de clases concretas, clases abstractas e interfaces
b) Que ambos son un conjunto de clases.

3. Inicien el diseño con un diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos.
En astah, crear un diagrama de clases (cambiar el nombre por Package Diagram0)
Realizado en Astah

4. Ahora que conocen los paquetes, ¿cuál es la visibilidad real que del modificador protected?
Este modificador permite el acceso desde subclases y miembros del mismo paquete.

Arquitectura detallada. [En lab03.doc y artico.asta]

1. Usando ingeniería reversa prepararen el proyecto para **MDD**. Organicen el diseño estructural actual de la aplicación: realice los diagramas de clases de los paquetes de presentación y dominio. Muevan las clases a los paquetes correspondientes.
Realizado en Astah.

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

a) En el paquete de dominio
b) Asociándolo con la clase Ártico ya que en esta es donde se encuentran las acciones que se pueden realizar.
c) Porque de allí probaremos el código.

3. Escriban dos pruebas y Ejecutenlas. No olvide el estandar de nombres deberia y noDeberia. Presente un pantallazo con el resultado de las pruebas.

4. Estudie la clase Artico. ¿Qué tipo de colección se usa para albergar los elementos? ¿Puede recibir esquimales? ¿Por qué?

a) El tipo de colección que se utiliza es un ArrayList
b) Si puede recibir esquimales ya que estos implementan la interfaz en la clase Ártico.

5. Estudie la clase Persona; ¿qué atributos pueden usar otras clases? ¿que atributos pueden usar sus subclases? ¿qué métodos no pueden cambiar las personas?

a) Los atributos que pueden usar las otras clases son los public final static : ARRIBA, FRENTE y ABAJO.
b) Los atributos que pueden cambiar sus subclases son los protected: nombre y color.
c) Los metodos que no pueden cambiar las personas son: mueveBrazo, muevePierna, getPosicionBrazo, getPosicionPierna, avance, getPosicionX y getPosicionY.

6. Estudie el código de la clase EnArtico; ¿qué atributos pueden usar otras clases? ¿qué métodos **deben** implementar las clases que están en Artico?

a) Los atributos que pueden usar otras clases son todos los de EnArtico
b) Los métodos que deben implementar las clases que están en Artico son: acción, corte e improvise

7. Revisen el contexto de los esquimales. ¿qué atributos son parte de su estado? ¿qué atributos puede acceder en su código? ¿qué métodos no pueden cambiar los esquimales? ¿Qué métodos pueden cambiar parcialmente los esquimales?

a) Los atributos que son parte de su estado son: palabras
b) Los atributos que puede acceder en su código son: palabras, nombre, color, ARRIBA, FRENTE y ABAJO.
c) Los métodos que no pueden cambiar los esquimales son: acción y mensaje.
d) Los métodos que pueden cambiar parcialmente los esquimales son: los públicos que se heredan de la clase Persona.

8. Revisen el comportamiento de los esquimales: ¿Qué métodos deben tener el mismo comportamiento para todos los esquimales? ¿Qué métodos pueden cambiar totalmente? ¿Qué métodos pueden cambiar en parte? Explique.

a) Los métodos que deben tener el mismo comportamiento son: mensaje y el constructor.
b) Los métodos que pueden cambiar totalmente son: actúe, acción y corte

9. Completen el diagrama de clases de la capa de dominio.
Realizado en Astah.

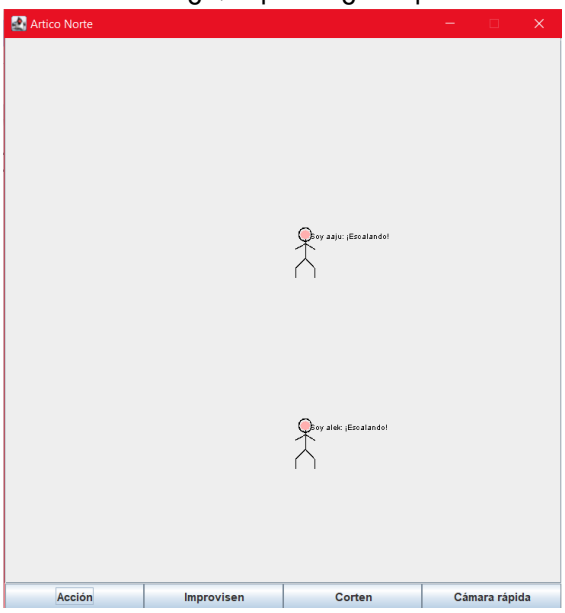
Ciclo 1. Esquimales normales [En lab03.doc y *.java]

(NO OLVIDE BDD – MDD) (Construir: diseñar, implementar y probar)

1. Estudie el código de la clase Esquimal : ¿de qué color es? ¿qué mensaje tiene? ¿qué hacen cuando dan orden de acción? ¿cuando cortan? ¿cuándo improvisan?. Explique claramente dónde está la información de las respuestas de cada una de las preguntas.

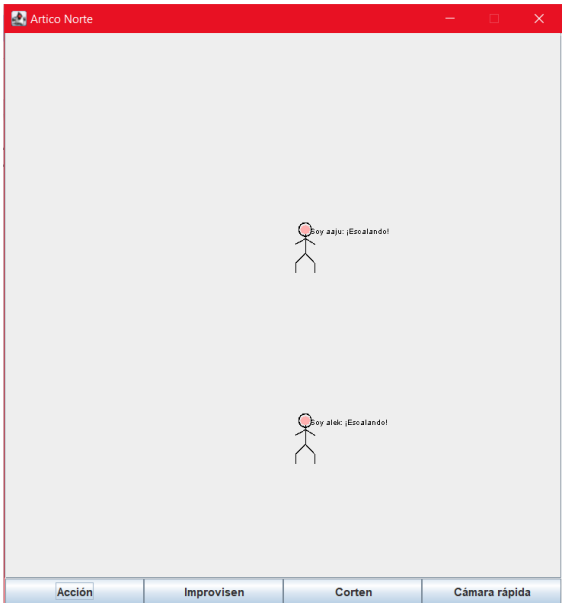
a) El color es: Black. Lo encontramos en el constructor de la clase abstracta Persona.
b) El mensaje es: Escalando. Lo encontramos en el constructor de la clase Esquimal.
c) Por ahora nunca.
d) Por ahora nunca.
e) Por ahora nunca.

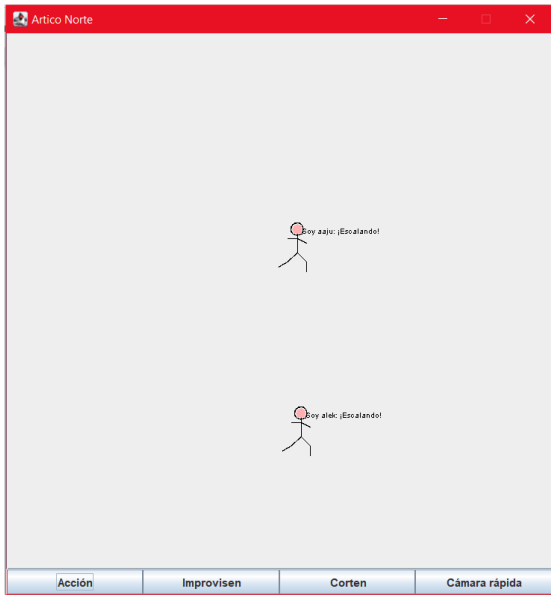
2. En el método algunosEnArtico de la clase Artico cree dos esquimales en diferentes posiciones y acondiciónelos al Artico llámelos aaju y alek. Ejecuten el programa y Capturen la pantalla. ¿Qué pasa ahora? ¿Pídales acción? ¿Qué pasa? ¿Por qué?



a) Cuando se les pide acción a los Esquimales no hacen nada. Porque el botón de acción en la clase Ártico no está construido.

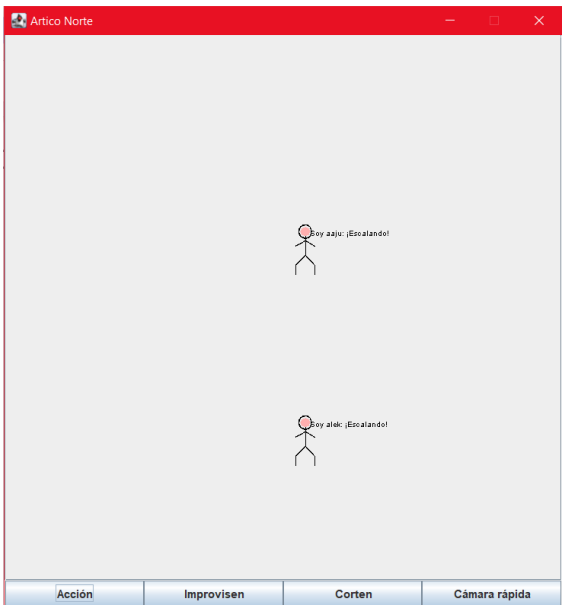
3. En este punto vamos a construir el método que atiende el click del botón ***Acción*** de la interfaz: el método llamado accion() de la clase Artico. Ejecuten el programa y haga tres click en el botón **Acción**. ¿Actúan aaju y alek como lo esperaban? Capturen la pantalla inicial y la final.

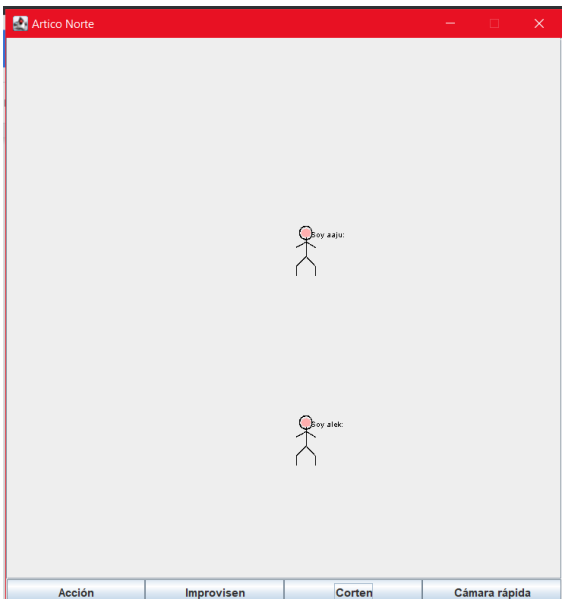




a) Cuando se les pide acción a los Esquimales, mueven sus brazos y piernas siguiendo un orden, como lo especifica el realizar la acción, entonces si está bien.

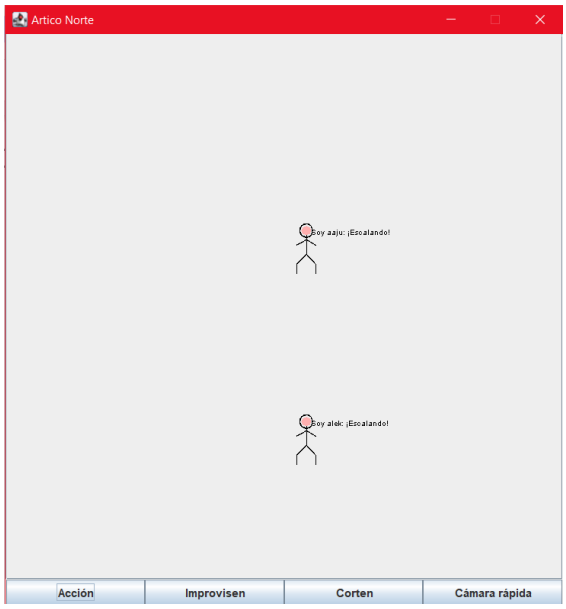
4. En este punto vamos a construir el método que atiende el click del botón **Corten** de la interfaz: el método llamado corten() de la clase Artico. Construya el método, ejecute el programa y haga click en el botón **Corten**. ¿Cómo quedan todos los esquimales después de esta orden? ¿Es adecuado? Capturen la pantalla inicial y la final.

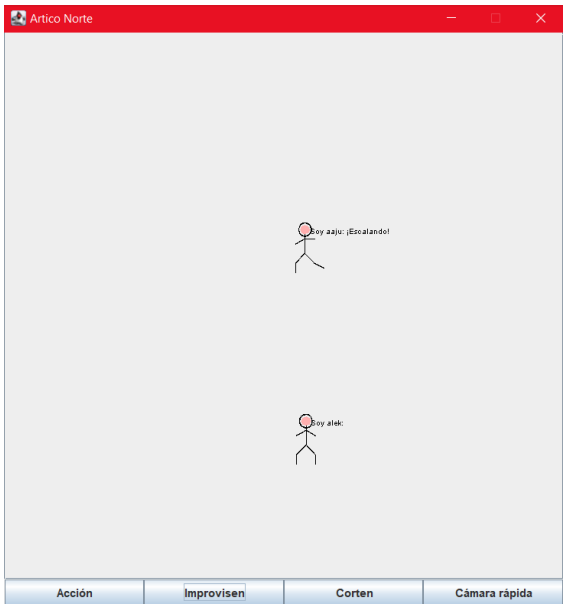




a) Los Esquimales se quedan quietos, tambien desaparece su letrero Escalando y los paraliza practicamente como lo ordena lo especifica el método corten.

5. En este punto vamos a construir el método que atiende el click del botón **Improvisen** de la interfaz: el método llamado improvisen() de la clase Artico. Construya el método, Ejecuten el programa y haga click en el botón **Improvisen**. ¿Como quedan todos los esquimales después de esta orden? ¿Es adecuado? Capturen la pantalla inicial y la final.





a) Los Esquimales realizan su improvisación por separado, uno continúa escalando es decir actúa y el otro corta sus acciones, si es adecuado, ya que así está definido.

6. Capturen el diseño de secuencia, el diagrama de clase y el resultado de ejecución de las pruebas de unidad. Expliquen.
Realizado en astah

Ciclo 2. Incluyendo a los esquimales sordos **[En lab03.doc, Artico.asta y *.java]**
(NO OLVIDE BDD – MDD) (Construir: diseñar, implementar y probar)
El objetivo de este punto es permitir recibir en la Artico esquimales sordos. Los esquimales sordos: (i) están vestidos de verde; (ii) cuando se les pide acción, solo suben los brazos; (iii) cuando se les pide que corten, avanzan cinco veces en dirección norte buscando el polo y (iv) cuando se les pide que improvisen cortan y se visten de amarillo (solo quedan amarillos en el corte). Adicionalmente, su mensaje es “Qué qué?” .

1. Implemente este nuevo esquimal. ¿cuáles métodos se sobre-escriben (overriding)?
Varios

2. Adicionen una pareja de esquimales sordos, llámelos aguu y ivanna, Ejecuten el programa y pídales a todos que actúen, corten e improvisen. Capturen una pantalla significativa. ¿Qué pasa?

3. Presenten los cambios en el diseño de secuencia, los cambios en el diagrama de clase y el resultado de ejecución de las pruebas de unidad. Expliquen.

Ciclo 3. Adicionando iglus **[En lab03.doc, Artico.asta y *.java]**
El objetivo de este punto es incluir en la Artico iglus (sólo vamos a permitir un tipo de iglú). Las iglus

cuando están en acción son redondos y negros, cuando están en corte son blancos con mensaje "CERRADO" y cuando improvisan repiten lo que hicieron la última vez.

(NO OLVIDE BDD – MDD) (Construir: diseñar, implementar y probar)

1. ¿Qué debemos hacer para tener iglús en la Artico?
   En la clase Artico agregar un iglú de manera similar como hicimos con los Esquimales

2. Construya la clase Iglu para poder adicionarla en el Artico. ¿qué cambios incluyó?
   En la clase iglú a diferencia de las otras teníamos que poner cuatro iglus cada uno en una esquina del artico
3. Adicionenn cuatro iglus en las esquinas del Artico, llámenlas superiorDerecha, superiorIzquierda, inferiorDerecha y inferiorIzquierda. Ejecuten el programa pídales a todos acción. Capturenn la pantalla. ¿Qué pasa? ¿es correcto?

4. Presenten los cambios en el diseño de secuencia, los cambios en el diagrama de clase y el resultado de ejecución de las pruebas de unidad. Expliquen.

Ciclo 4. Creando un nuevo esquimal: el explorador **[En lab03.doc, Artico.asta y *.java]**

El objetivo de este punto es permitir recibir en la Artico esquimales miedosos. El esquimal explorador cuando le piden que actue debe hacer un recorrido tal que garantice que recorre todo el artico, de modo que sin importar donde está el polo norte eventualmente pase por allí. Cuando le piden que corte busca acercarse al esquimal más cercano. Cuando le piden que improvise, indica la posición en la que se encuentra. Está vestido con color rojo.

(NO OLVIDE BDD – MDD) (Construir: diseñar, implementar y probar)

1. Implemente este nuevo esquimal. ¿cuáles métodos se sobre-escriben (overriding)?
   Varios
2. Adicionen una pareja de esquimales minuciosos, llámelos nanuk y sialuk, Ejecuten el programa y pídales a todos que actúen y que paren. Capturen la pantalla. ¿Qué pasa?
3. Presenten los cambios en el diseño de secuencia, los cambios en el diagrama de clase y el resultado de ejecución de las pruebas de unidad. Expliquen.

Ciclo 5. Nuevo esquimal: Proponiendo y diseñando **[En lab03.doc, Artico.asta y *.java]**

El objetivo de este punto es permitir recibir en un nuevo tipo de esquimal. **(NO OLVIDE BDD – MDD) (Construir: diseñar, implementar y probar)**

1. Propongan, describan e implementen un nuevo tipo de esquimal.
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.
3. Presenten los cambios en el diseño de secuencia, los cambios en el diagrama de clase y el resultado de ejecución de las pruebas de unidad. Expliquen.

Ciclo 6. Nuevo elemento: Proponiendo y diseñando **[En lab03.doc, Artico.asta y *.java]**

El objetivo de este punto es permitir recibir en un nuevo elemento en la Artico

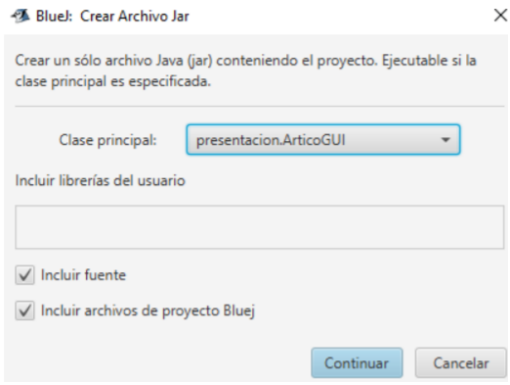
(NO OLVIDE BDD – MDD) (Construir: diseñar, implementar y probar)

1. Propongan, describan e implementen un nuevo tipo de elemento
2. Incluyan dos de ellos con el nombres semánticos. Ejecuten el programa con dos casos significativos. Explique la intención de cada caso y Capturen las pantallas correspondientes.

3. Presenten los cambios en el diseño de secuencia, los cambios en el diagrama de clase y el resultado de ejecución de las pruebas de unidad. Expliquen.

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

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



2. Consulten el comando java para ejecutar un archivo jar. Ejecutenlo ¿qué pasa?
El codigo es desde la carpeta y para ejecutar ponemos: java -jar “nombre del archivo.jar”
Se ejecuta nuestro ArticoGui

3. ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.
Podemos tener un acceso directo al método main es decir estamos haciendo lo mismo que en BlueJ pero desde la consola.

DE BLUEJ A CONSOLA

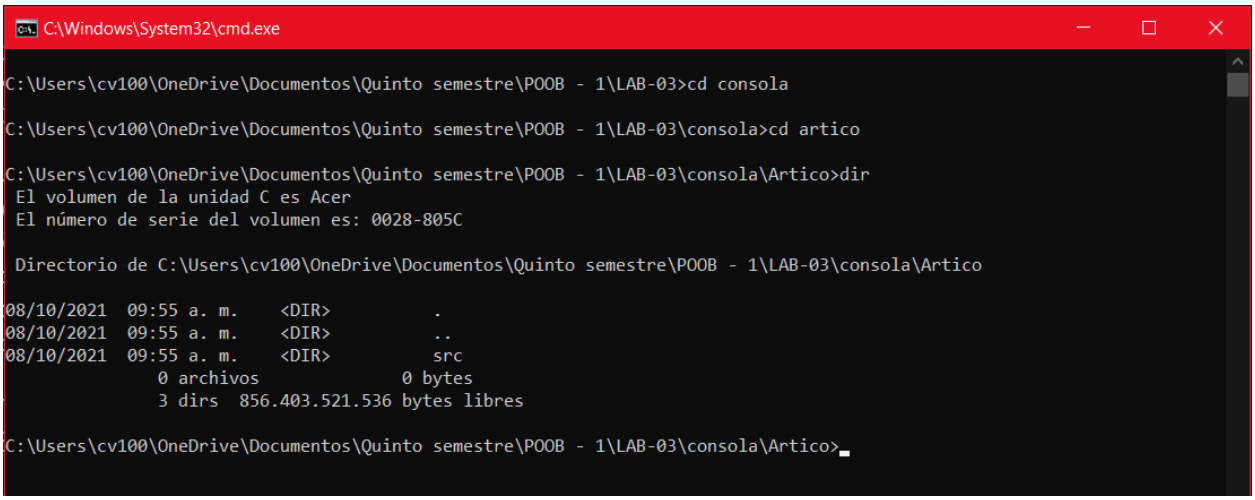
En esta sección del laboratorio vamos a aprender a usar java desde consola. Para esto se va a trabajar con el proyecto del punto anterior.

Comandos básicos del sistema operativo [En lab03.doc]

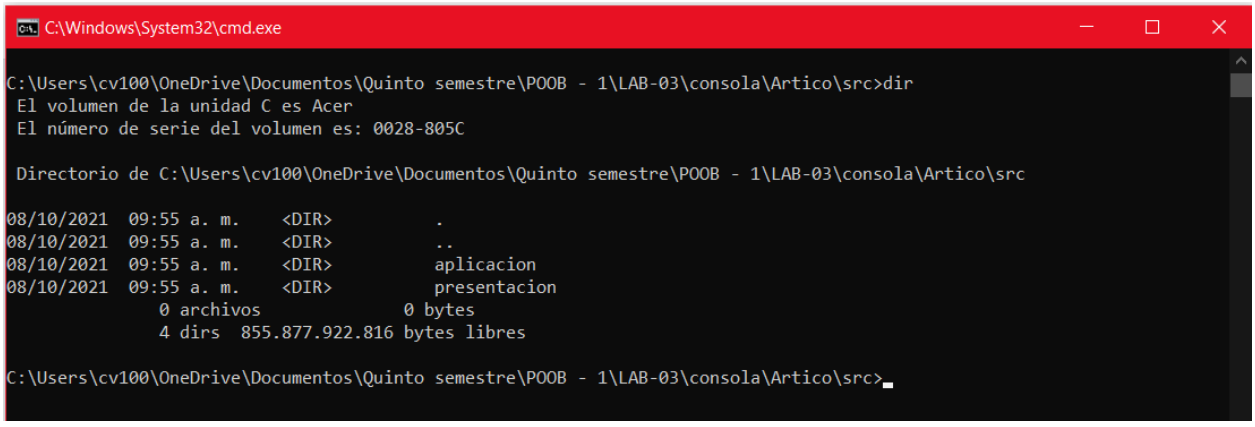
Antes de iniciar debemos repasar los comandos básicos del manejo de la consola.

1. Investiguen los comandos para moverse en la estructura de directorios: crear, borrar, listar su contenido y copiar o eliminar un archivo.
   Comando crear: md
   Comando borrar: del /s /q
   Comando listar: dir
   Comando copiar: copy / xcopy
   Comando eliminar: del | rd borrar directorio

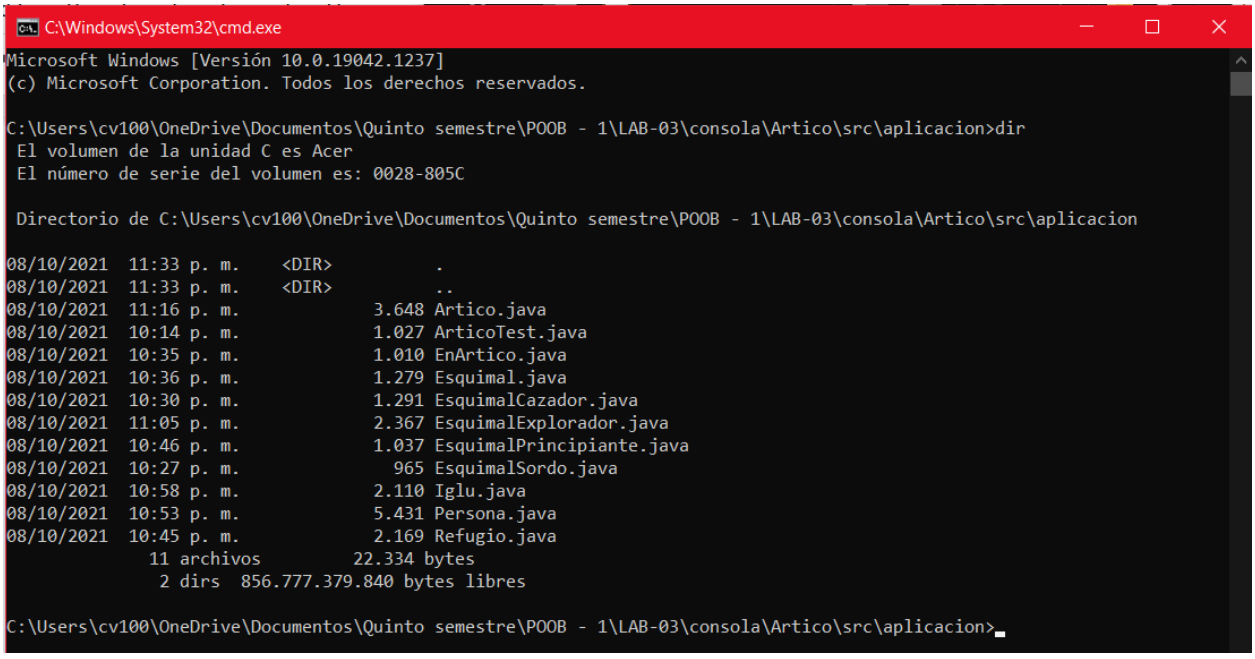
2. Organicen un nuevo directorio con la estructura propuesta para probar desde allí su habilidad con los comandos de consola. Consulten y Capturen el contenido de su directorio
   Artico



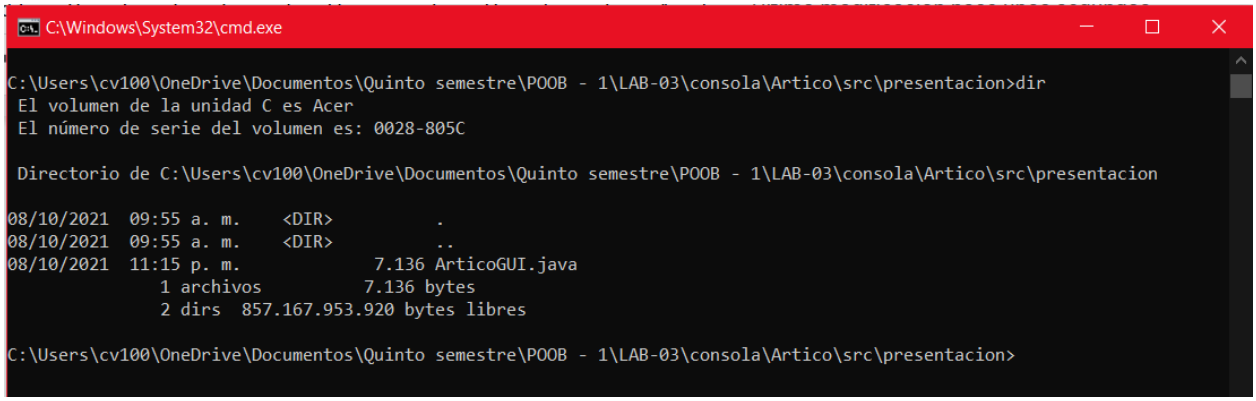
src



aplicacion

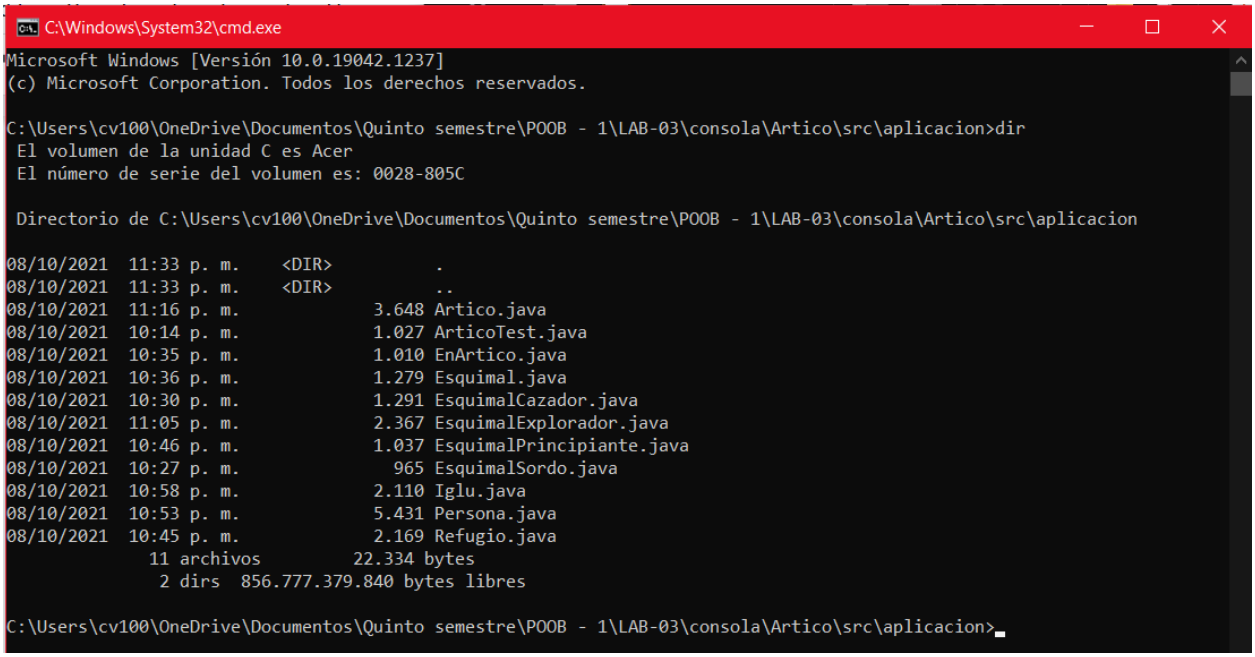


presentacion



pruebas

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



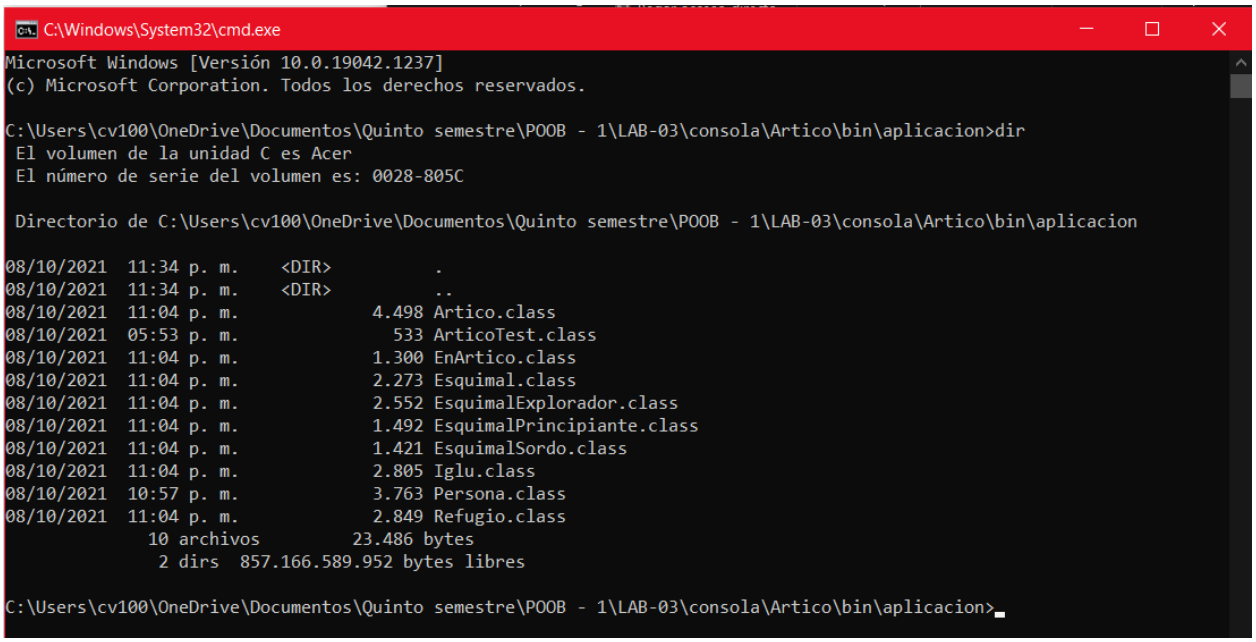
Estructura de proyectos java [En lab03.doc]

En java los proyectos se estructuran considerando tres directorios básicos.

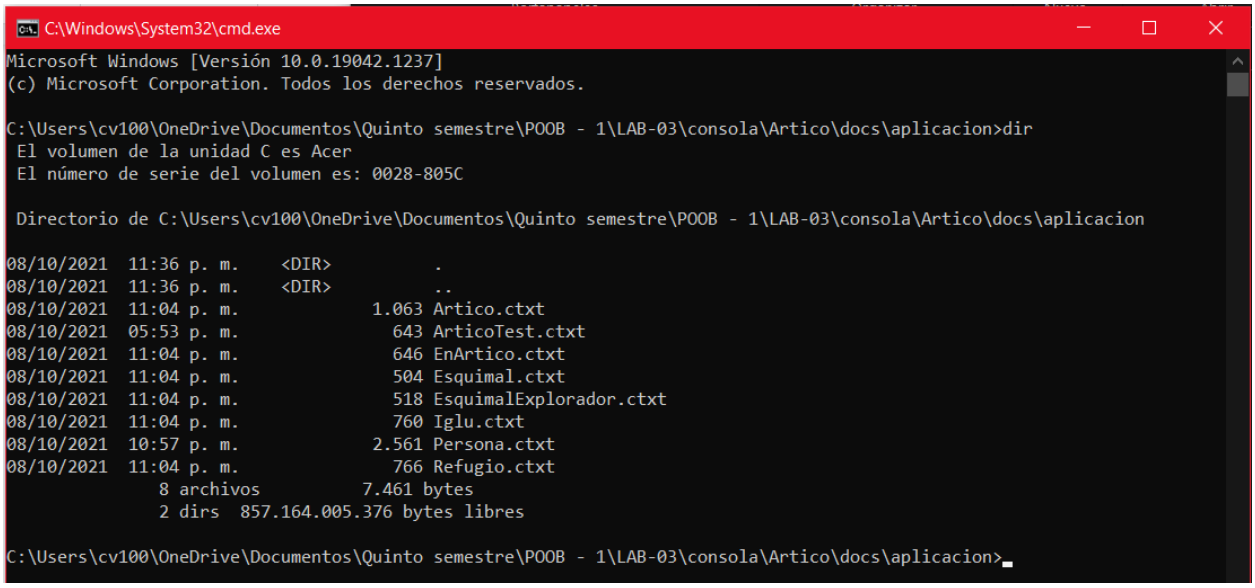
artico
src
bin
docs

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

Los archivos con extencion *.class



Los archivos con extencion *.ctxt



Comandos de java [En lab03.doc]

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

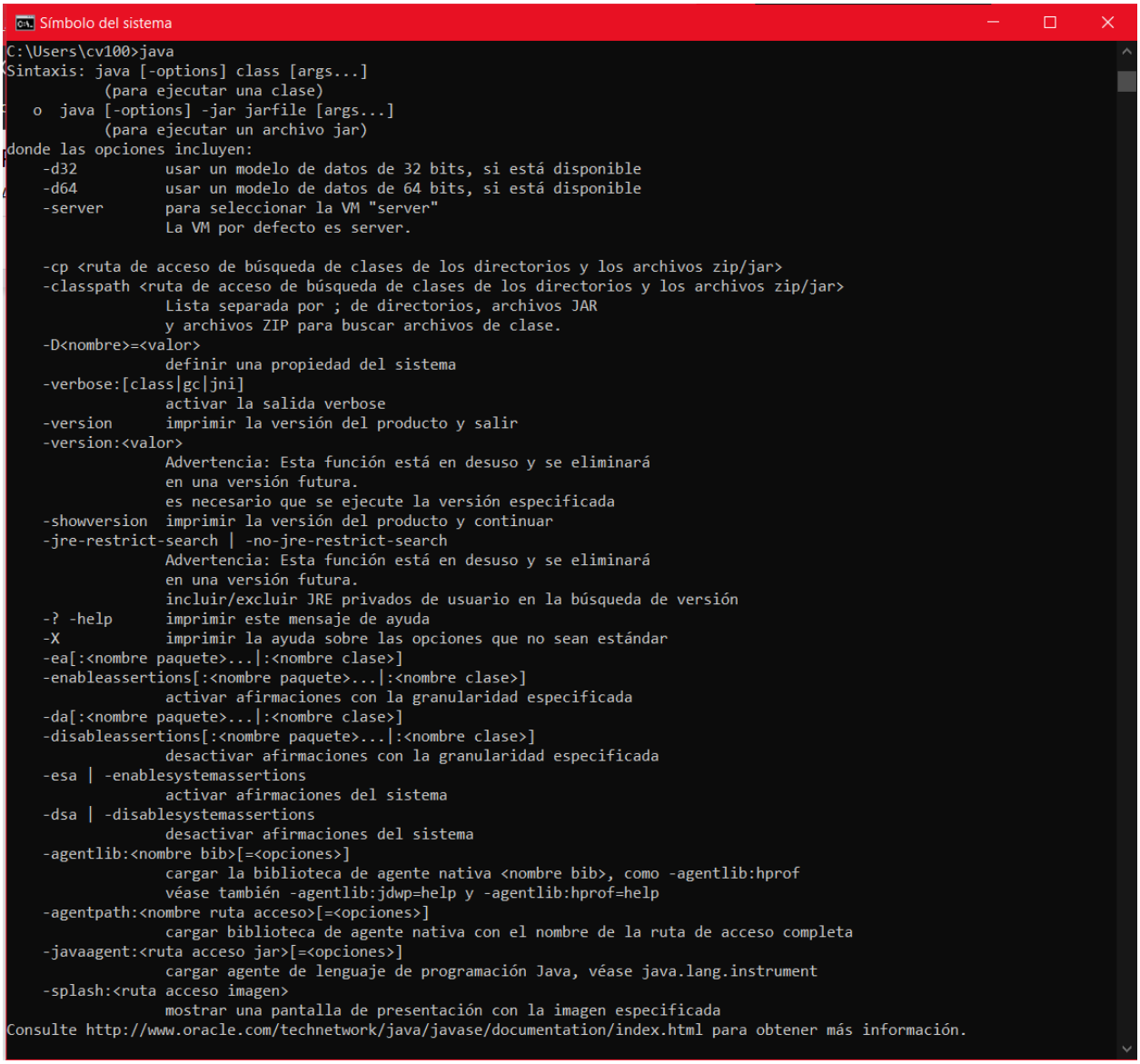
   javac: Es una utilidad incluida en el JDK, es el compilador de java, un programa ejecutable que nos permite compilar nuestro código fuente.

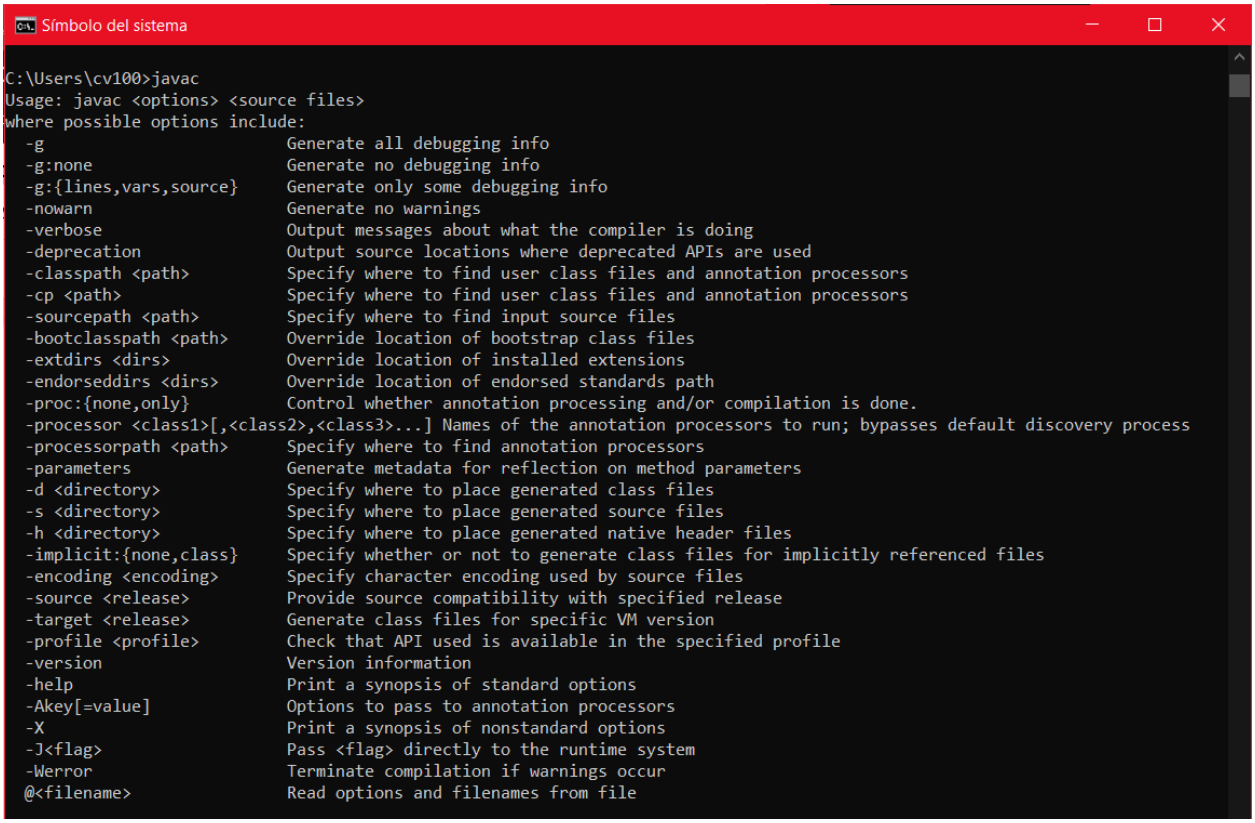
   java: Es una utilidad incluida en el JDK, que ejecuta el programa en nuestro código fuente

   javadoc: Es una utilidad incluida en el JDK, que genera documentación automática en formato HTML, a partir del código fuente.

   jar: Es una utilidad incluida en el JDK, que realiza la empaquetamiento incluyendo directorios con clases.

2. Creen una sesión de consola y consulten en línea las opciones de los comandos java y javac. Capturen las pantallas.


Símbolo del sistema
C:\Users\cv100>java
Sintaxis: java [-options] class [args...]
(para ejecutar una clase)
o java [-options] -jar jarfile [args...]
(para ejecutar un archivo jar)
donde las opciones incluyen:
-d32 usar un modelo de datos de 32 bits, si está disponible
-d64 usar un modelo de datos de 64 bits, si está disponible
-server para seleccionar la VM "server"
La VM por defecto es server.
-cp <ruta de acceso de búsqueda de clases de los directorios y los archivos zip/jar>
-classpath <ruta de acceso de búsqueda de clases de los directorios y los archivos zip/jar>
Lista separada por ; de directorios, archivos JAR
y archivos ZIP para buscar archivos de clase.
-D<nombre>=<valor>
definir una propiedad del sistema
-verbose:[class|gc|jni]
activar la salida verbose
-version imprimir la versión del producto y salir
-version:<valor>
Advertencia: Esta función está en desuso y se eliminará
en una versión futura.
es necesario que se ejecute la versión especificada
-showversion imprimir la versión del producto y continuar
-jre-restrict-search | -no-jre-restrict-search
Advertencia: Esta función está en desuso y se eliminará
en una versión futura.
incluir/excluir JRE privados de usuario en la búsqueda de versión
-? -help imprimir este mensaje de ayuda
-X imprimir la ayuda sobre las opciones que no sean estándar
-ea[:<nombre paquete>...|:<nombre clase>]
-enableassertions[:<nombre paquete>...|:<nombre clase>]
activar afirmaciones con la granularidad especificada
-da[:<nombre paquete>...|:<nombre clase>]
-disableassertions[:<nombre paquete>...|:<nombre clase>]
desactivar afirmaciones con la granularidad especificada
-esa | -enablesystemassertions
activar afirmaciones del sistema
-dsa | -disablesystemassertions
desactivar afirmaciones del sistema
-agentlib:<nombre bib>[=<opciones>]
cargar la biblioteca de agente nativa <nombre bib>, como -agentlib:hprof
véase también -agentlib:jdwp=help y -agentlib:hprof=help
-agentpath:<nombre ruta acceso>[=<opciones>]
cargar biblioteca de agente nativa con el nombre de la ruta de acceso completa
-javaagent:<ruta acceso jar>[=<opciones>]
cargar agente de lenguaje de programación Java, véase java.lang.instrument
-splash:<ruta acceso imagen>
mostrar una pantalla de presentación con la imagen especificada
Consulte http://www.oracle.com/technetwork/java/javase/documentation/index.html para obtener más información.



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

java -version => javac 1.8.0_202
javac -version => java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)

Compilando [En lab03.doc]

1. Utilizando el comando javac, **desde el directorio raíz (desde artico 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.
   El comando es: javac src\aplicacion*.java -d bin

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

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 Capturen la pantalla.

Ejecutando [En lab03.doc]

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

Probando [En lab03.doc]

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?
2. 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
3. Pegue en su documento el resultado de las pruebas

Empaquetando [En lab03.doc]

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 ?
2. ¿Cómo se ejecuta el proyecto empaquetado?

RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/ Hombre)
   (15 horas / Edgar Henao)
   (15 horas / Cesar Vásquez)

2. ¿Cuál es el estado actual del laboratorio? ¿Por qué? (Para cada método incluya su estado)
   El estado actual del laboratorio es casi finalizado

3. Considerando las prácticas XP del laboratorio de hoy ¿por qué consideran que son importantes?
   Las prácticas XP para este laboratorio son importantes ya que debe ser colectivo, obviamente ambos integrantes en este caso deben aportar sus ideas, y así poder realizar las mejores elecciones sobre lo que se está trabajando, de otra manera podemos decir que es útil programar por separado y que luego el compañero encuentre en que puede estar fallando el otro y así realizar la corrección luego en conjunto, también funciona para cuando ambas personas no tengan las mismas disponibilidades de tiempo pero se requiere adelantar el trabajo.

4. ¿Cuál considera fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?
   Nuestro mayor logro fue
   No tuvimos como tal un mayor problema sino que se nos dificulto lo largo que es este laboratorio, pues toca dedicarle bastante tiempo.

5. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?
   Lo que hicimos bien como equipo fue llevar el proceso, parando y observando detalladamente los puntos del laboratorio, también preguntar a profesor y monitor cuando no se tenía claridad en alguna cosa. Nos comprometemos a dedicar más tiempo para el próximo laboratorio y también estudiar bien las lecturas y demás materiales de clase.