Angie Medina – Jose Pérez

POOB - 1

2020 - 2

Laboratorio 3/6 Herencia e interfaces

**Conociendo**

1. En el directorio descarguen los archivos contenidos en islaTortuga.zip. Revisen el código de la aplicación

* ¿Cuántos paquetes tiene?

El archivo contiene 2 paquetes: presentación y dominio

* ¿Cuántas clases tiene en total?

Tiene 6 clases: Isla, EnIsla, Persona, Pirata, Palmera y IslaGUI

* ¿Cuál es la clase con la que se ejecuta la aplicación? ¿Por qué?

Isla, esta es la principal ya que contiene todos los elementos de la aplicación y ocurren eventos en esta.

1. Ejecuten el programa.
   * ¿Qué funcionalidades ofrece?

Actúen, Aprisa, Paren y Decidan

* + ¿Qué hace actualmente?

Actualmente no realiza ninguna de las funcionalidades ofrecidas

* + ¿Por qué?

Porque no se han implementado en el paquete dominio

**Arquitectura general**

1. Consulte
   * El significado de las palabras package e import de java.
     + ***package***: contenedor que permite agrupar clases, generalmente con funcionalidad y elementos comunes
     + ***Import:*** palabra reservada, permite a un módulo de código hacer referencia a los tipos definidos en el paquete especificado
   * ¿Qué es un paquete? ¿Para qué sirve?

Los paquetes son el mecanismo que usa Java para facilitar la modularidad del código. Un paquete puede contener una o más definiciones de interfaces y clases, distribuyéndose habitualmente como un archivo. Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia ***import***.

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

El número de subdirectorios al igual que sus nombres coinciden con los paquetes encontrados en BlueJ, el contenido dentro de estos corresponde a las clases e interfaces que encapsulan cada uno de los paquetes.

1. Revisando las clases que contiene, ¿cuál es el propósito de cada uno de los paquetes encontrados?

El propósito de ***presentación*** es ofrecer una interfaz gráfica al usuario. Por otro lado, el paquete ***dominio*** brinda los comportamientos de los elementos que pueden pertenecer a una isla.

1. Adicione el diseño de diagrama de paquetes en el que se presente los componentes y las relaciones entre ellos.

**Arquitectura detallada**

1. Usando ingeniería reversa prepararen el proyecto para MDD. Completen el diagrama de clases del paquete de dominio. No incluyan el paquete de presentación
2. Adicione en las fuentes la clase de pruebas necesaria para BDD. (No lo adicione al diagrama de clases) ¿En qué paquete debe estar? ¿Por qué? ¿Asociado a qué clase? ¿Por qué?

Debería estar en el paquete dominio ya que en este es donde se programa lo que después se vera reflejado en la interfaz (paquete presentación), esta asociada a la clase *Isla* ya que esta es la que contiene todos los elementos de la aplicación y es donde ocurren los eventos.

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

Se utiliza un ***arrayList<enIsla>***, puede recibir piratas pues es una implementación de la interfaz ***enIsla.***

1. Estudie la clase ***Persona***;
   * ¿Qué atributos pueden usar otras clases?
     + ARRIBA
     + FRENTE
     + ABAJO
   * ¿Qué atributos pueden usar otras clases?
     + Color
   * ¿Qué métodos no pueden cambiar las personas?
     + MuevaBrazo
     + MuevaPierna
     + getPosicionBrazo
     + getProsicionPierna
     + getPosicionX
     + getPosicionY

Ya que son métodos *final*

1. Estudie el código de la clase ***Pirata***; ¿qué atributos pueden usar otras clases?
   * ¿Qué atributos pueden usar sus subclases?
     + palabras
   * ¿qué métodos no pueden cambiar los piratas?
     + muevaBrazo
     + muevaPierna
     + getPosicionBrazo
     + getPosicionPierna
     + getPosicionX
     + getPosicionY
   * ¿qué métodos pueden cambiar parcialmente los piratas?
     + getColor
     + muevase
     + mensaje
2. Estudie el código de la clase E***nIsla***;
   * ¿qué atributos pueden usar otras clases?
     + FORMAS
     + r
   * ¿qué métodos deben implementar las clases que están en isla?
     + getPosicionX
     + getPosicionY
     + getColor
     + actue

**Ciclo 1. Actúan y descansan los Piratas**

1. Estudie el código de la clase ***Pirata***:
   * ¿de qué color es?

Negro

Esto se encuentra en el constructor de pirata, específicamente la línea:

*color = Color.BLACK;*

* + ¿qué mensaje tiene?

El mensaje varía dependiendo de:

* + - Si encontró el tesoro:

Soy *nombre del pirata: "*¡ENCONTRE EL TESORO!

* + - Si no encontró el tesoro:

Soy *nombre del pirata:* ¡Buscando!

Esta información se encuentra en el método *mensaje* de la clase Pirata

* + ¿cómo actúan?

Mueven sus extremidades, para cambiar de posición en una dirección aleatoria en una unidad, si el movimiento aleatorio no se puede realizar se vuelve a intentar hasta que el pirata se pueda mover.

Esta información se encuentra en el método *actue* de la clase Pirata

* + ¿qué hacen cuando paran?

Sus piernas vuelven a la posición inicial, y no dice nada.

Esta información se encuentra en el método *pare* de la clase Pirata

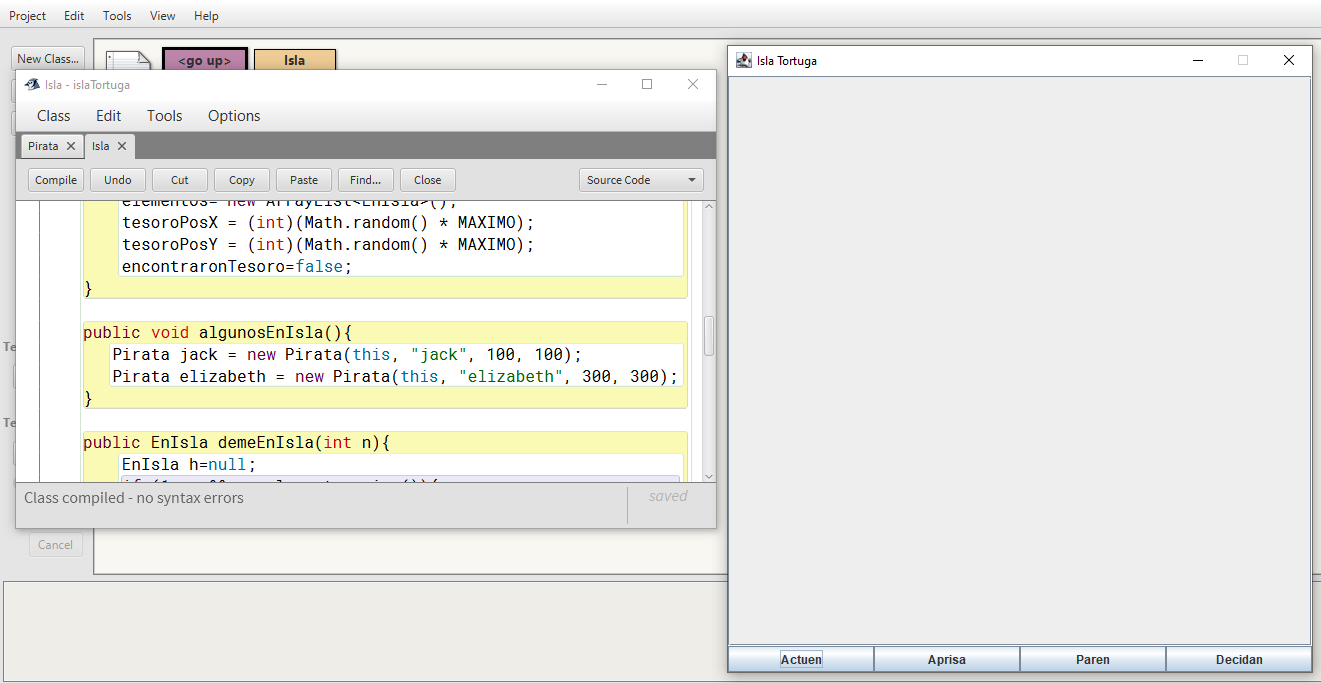
* + ¿cómo deciden?

El pirata decide si para o actúa aleatoriamente.

Esta información se encuentra en el método *decida* de la interfaz *enIsla.*

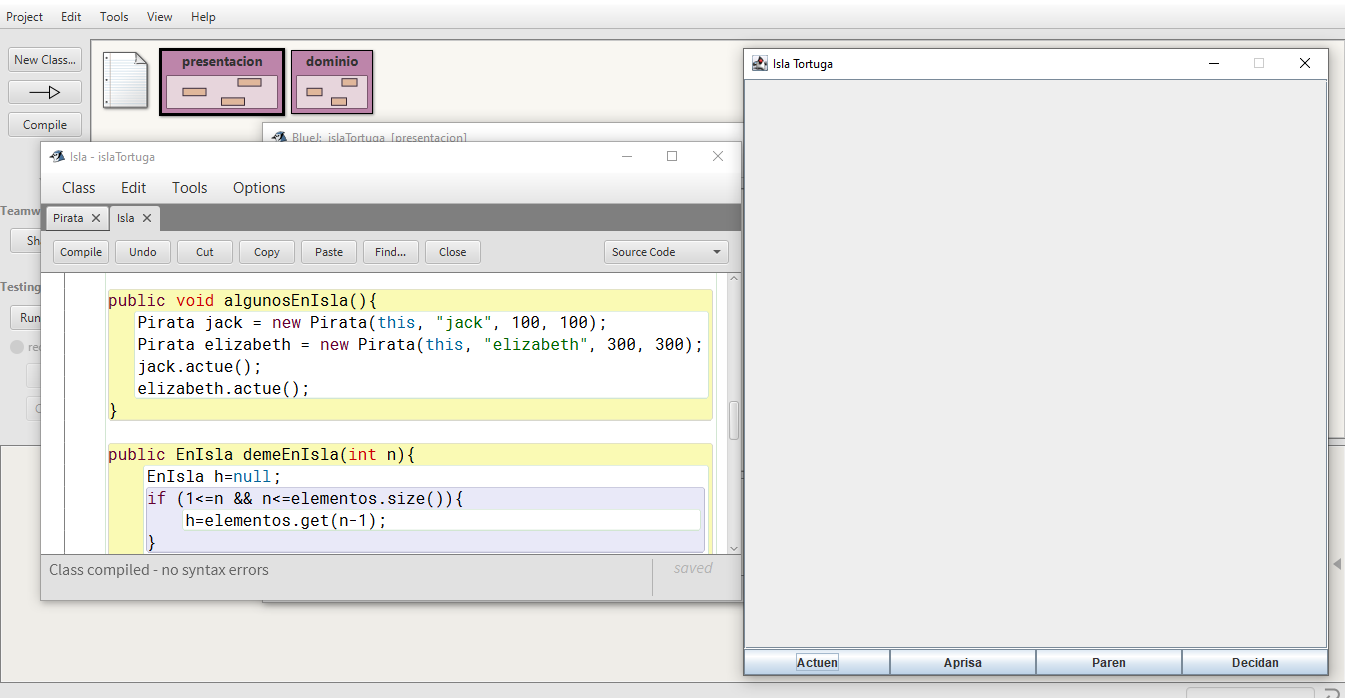
1. En el método ***algunosEnIsla*** de la clase Isla
   * Cree dos piratas en diferentes posiciones y acondiciónelos a la isla llámelos jack y elizabeth. Ejecute el programa y capture la pantalla. ¿Qué pasa ahora?

Nada



* + ¿pídales que actúen? ¿Qué pasa?

Nada

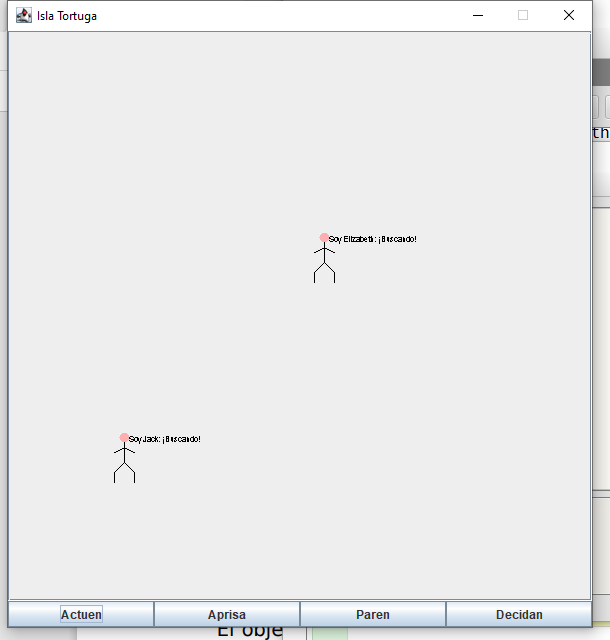
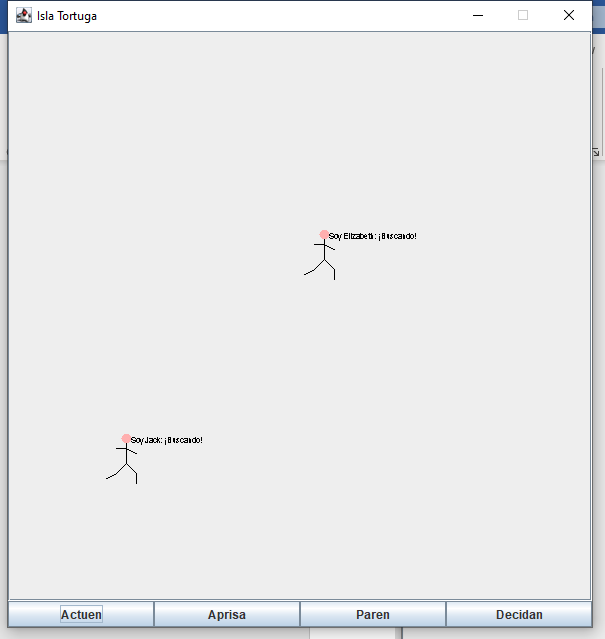


* + ¿Por qué?

Porque este método no está relacionado directamente con la interfaz, es decir, con ningún botón de la interfaz.

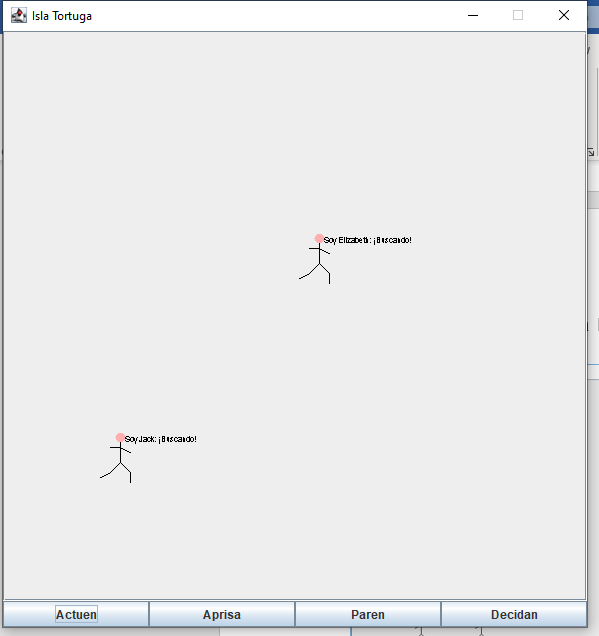
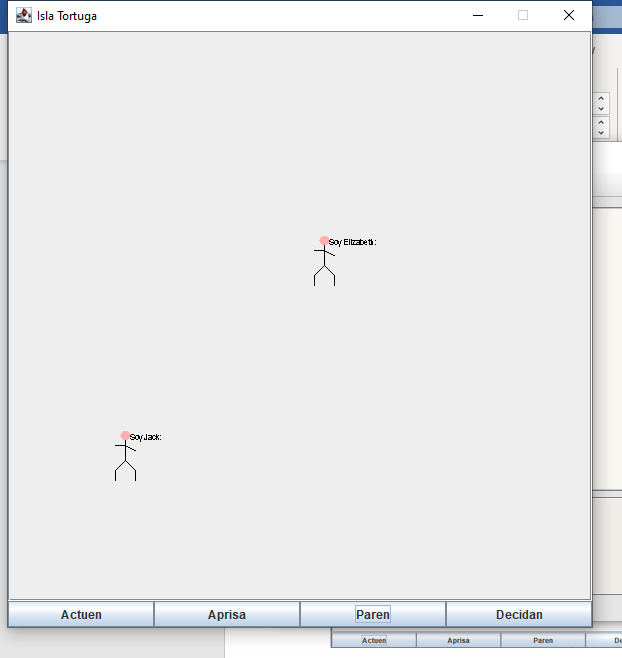
1. En este punto vamos a construir el método que atiende el click del botón Actuen de la interfaz: el método llamado actuen()de la clase Isla. Ejecute el programa y haga tres click en el botón Actuen. ¿Cómo actúan jack y elizabeth? Capture la pantalla inicial y la final.

Mueven sus brazos y piernas

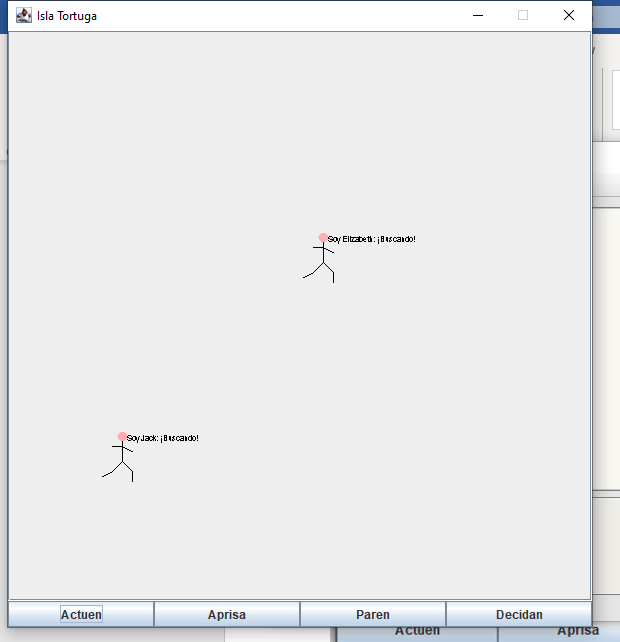
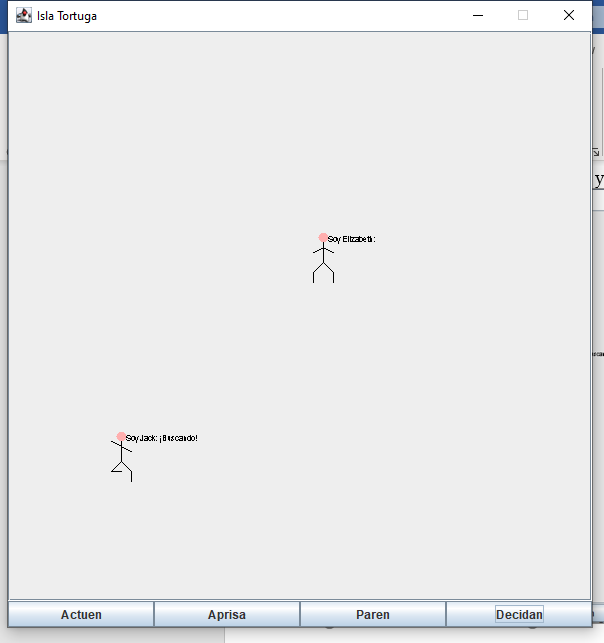
1. En este punto vamos a construir el método que atiende el click del botón Paren de la interfaz: el método llamado pare () de la clase Isla. Construya el método, ejecute el programa y haga click en el botón Paren. ¿Como quedan todos los piratas después de esta orden? Capture la pantalla inicial y la final.

Las piernas de todos los piratas vuelven a su posición inicial, y estos dejan de decir sus palabras

1. En este punto vamos a construir el método que atiende el click del botón Decidan de la interfaz: el método llamado decidan () de la clase Isla. Construya el método, ejecute el programa y haga click en el botón Decidan. ¿Como quedan todos los piratas después de esta orden? Capture la pantalla inicial y la final.

Después de esta orden cada pirata elegirá aleatoriamente si quiere actuar o parar.

**Ciclo 2. Incluyendo a los Piratas rebeldes**

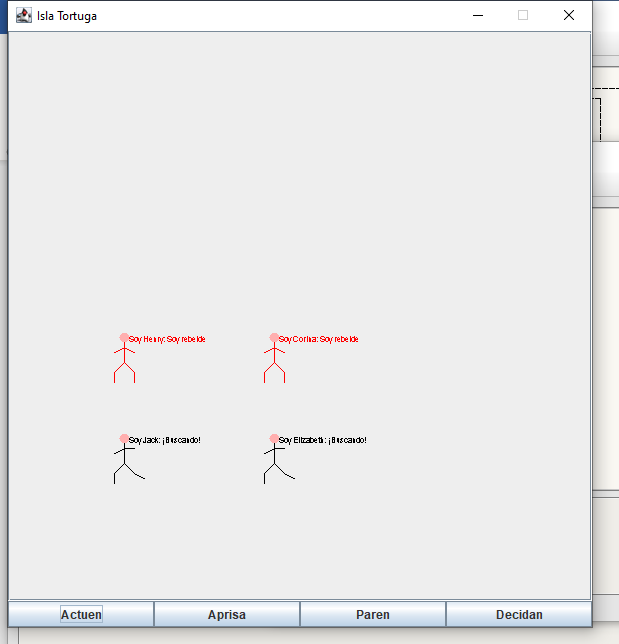
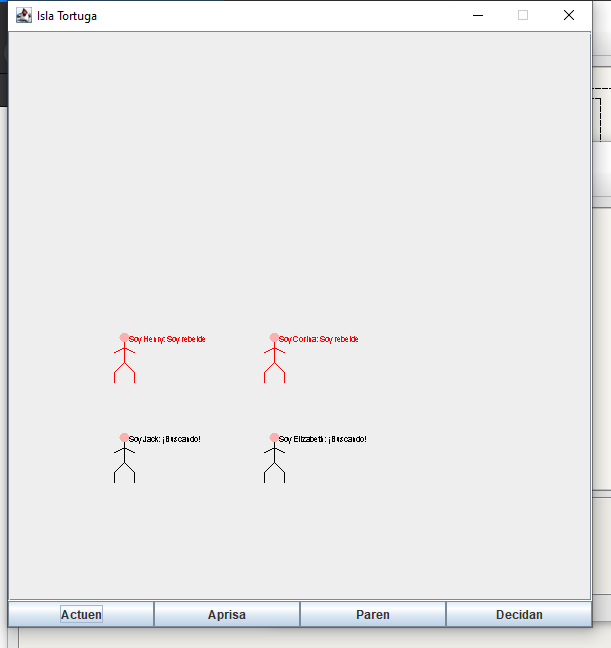
El objetivo de este punto es permitir recibir en la isla Piratas rebeldes. Los piratas rebeldes, vestidos de rojo, sólo actúan cada tres órdenes seguidas, cuando se les pide que paren, actúan y cuando se les pide que decidan, siempre paran si la última vez actuaron. Adicionalmente, su mensaje es “Soy rebelde”.

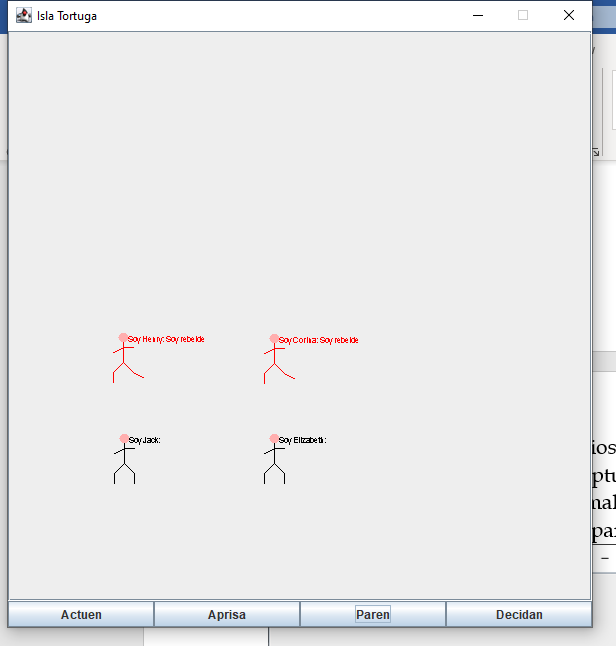
1. Implemente este nuevo pirata. ¿cuáles métodos se sobre escriben (overriding)?

Se sobre escriben los métodos de: actúe, pare y decida.

1. Adicione una pareja de piratas necios, llámelos, Henry y Corina, ejecute el programa y pídales a todos que actúen y que paren. Capture la pantalla. ¿Qué pasa?

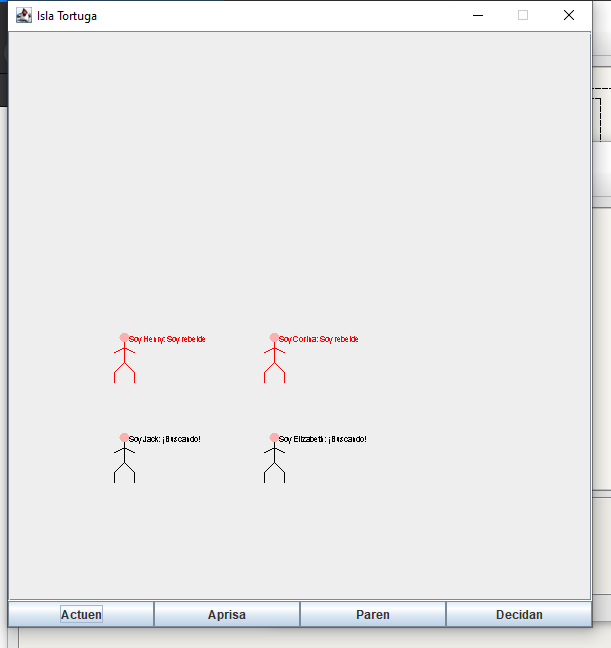
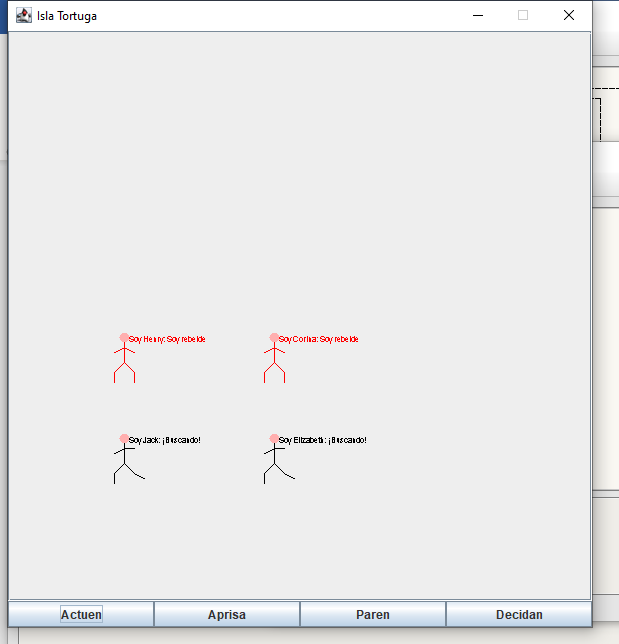
Jack y Elizabeth se comportan normal como se espera; los rebeldes no hicieron nada a la orden de actuen y actuaron a la orden de paren.

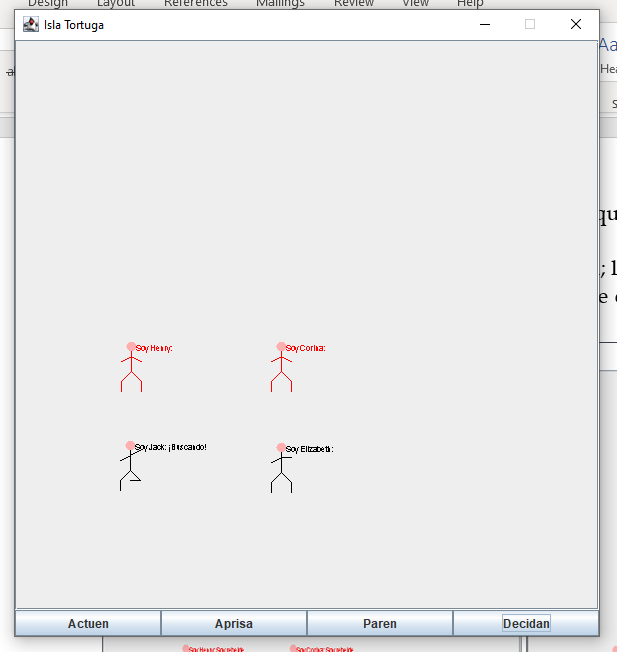
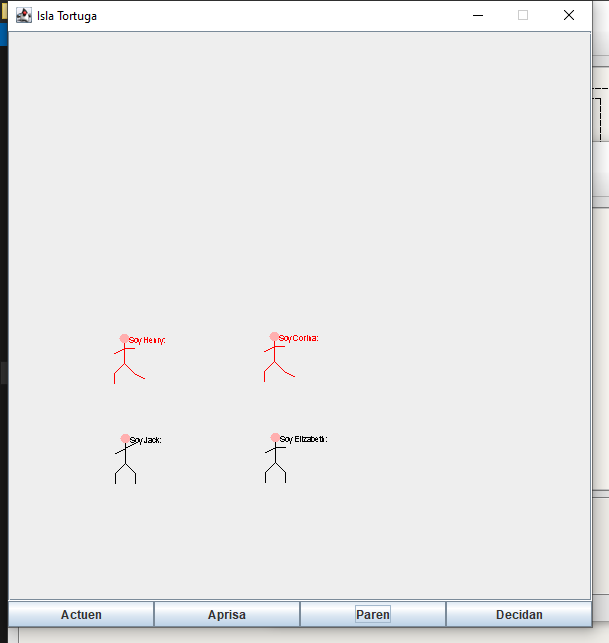




1. Nuevamente ejecute el programa y pídales a todos que actúen, decidan y paren. Capture la pantalla. ¿Qué pasa?

Jack y Elizabeth se comportan normal, como se espera; los rebeldes no hicieron nada cuando se dio la orden de actuar, decidieron parar a la orden de decidir y actuaron cuando se les dio la orden de parar.

**Ciclo 3. Adicionando palmeras**

El objetivo de este punto es incluir en la isla palmeras (sólo vamos a permitir un tipo de

palmeras). Las palmeras son verdes oscuras cuando actúan, verdes claras cuando cortan y deciden siguiendo dos veces las reglas de la forma estándar. Las palmeras son silenciosas.

1. ¿Qué debemos hacer para tener palmeras en la isla?

Hacer que la clase *Palmera* implemente la interfaz *enIsla.*

1. Construya la clase Palmera para poder adicionarla en el Isla. ¿qué cambios incluyó para aceptar este elemento, ¿debe cambiar en el código del Isla. en algo? ¿porqué?

La clase *Palmera* ahora implementa la interfaz *enIsla.*

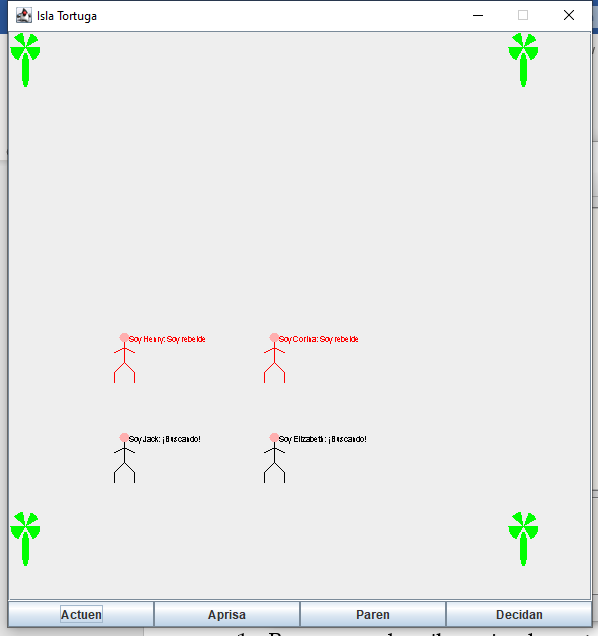
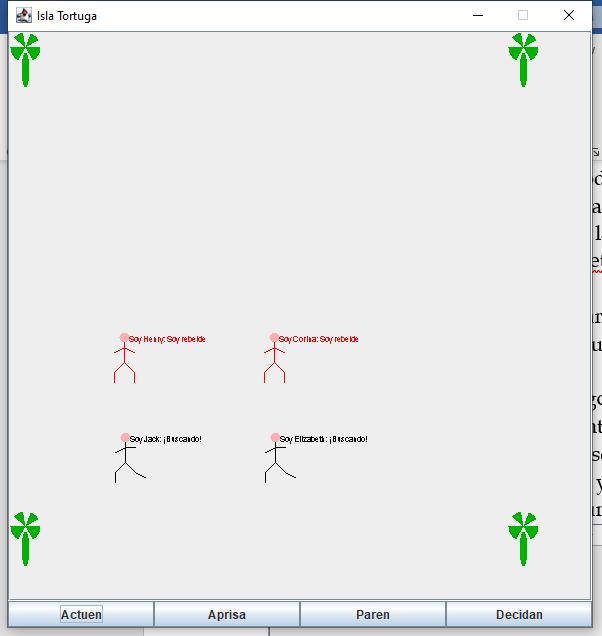
Se sobre escribieron los métodos: getPosicionX, getPosicionY, getColor, actue, decida, pare y forma.

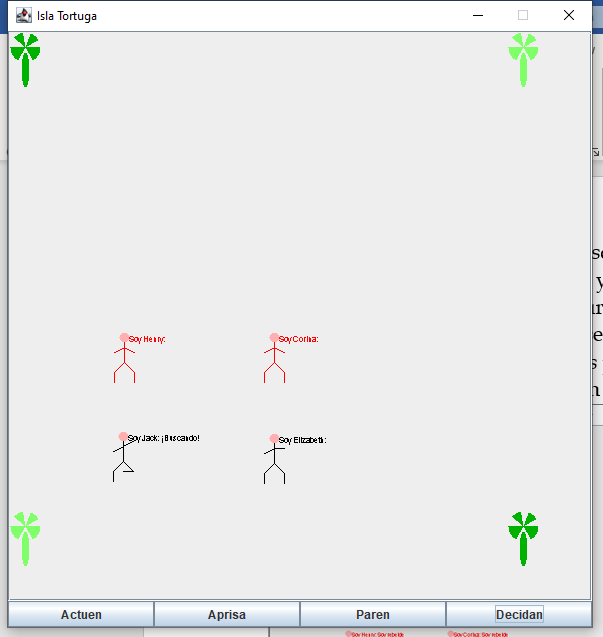
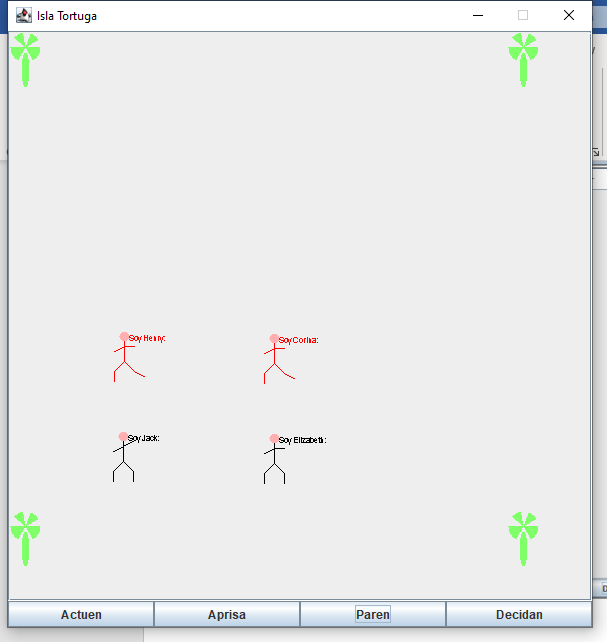
Adicionalmente se tuvo que agregar el String “Palmera” al contenedor de formas junto con adicionar código en IslaGUI para que esta reconociera y tuviera la forma que se quiere para una palmera.

No hay que cambiar nada del código de *isla* ya los elementos son un contenedor de objectos *enIsla* y palmera implementa esta interfaz.

1. Adicionen cuatro palmeras en las esquinas del Isla, llámenlas superiorDerecha, superiorIzquierda, inferiorDerecha y inferiorIzquierda. Ejecuten el programa pídales a todos que actúen, decidan y paren. Capturen la pantalla. ¿Qué pasa? ¿es correcto?

Los elementos que se adicionaron se comportan como es esperado, las palmeras se vuelven de un color verde oscuro cuando se les pide actuar, deciden dos veces según la regla estándar y cuando se les pide que paren toman un color verde claro.

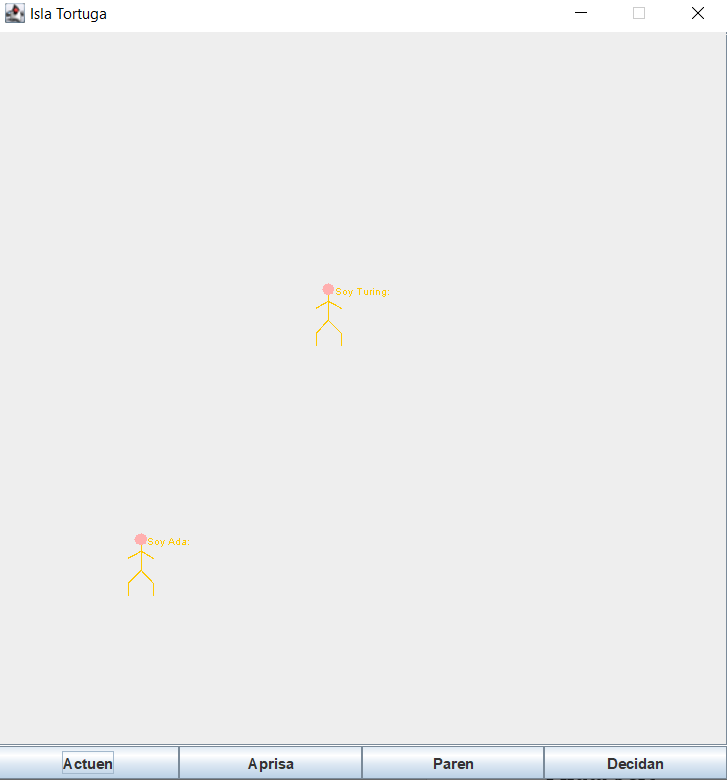
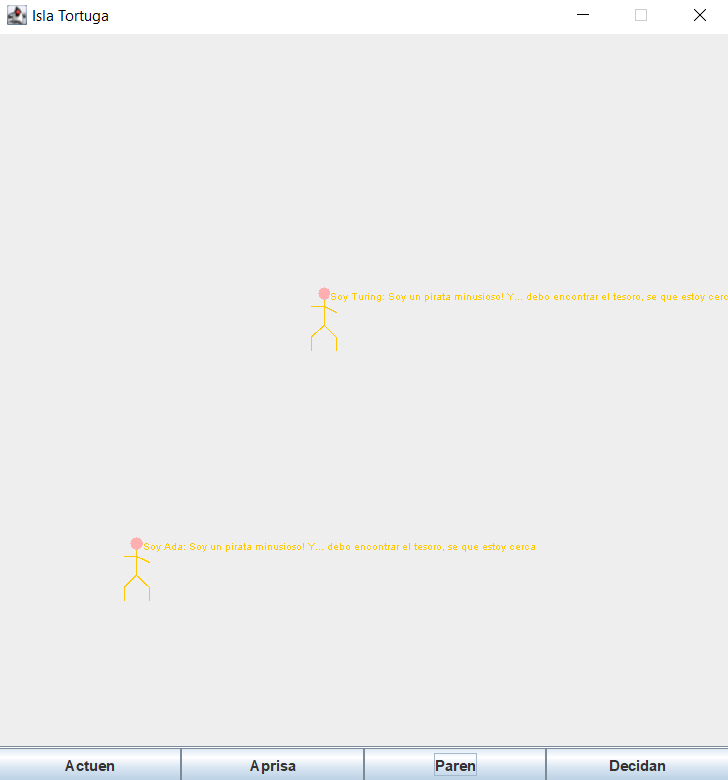
**Ciclo 4. Creando un nuevo pirata: el minucioso**

El objetivo de este punto es permitir recibir en la isla Piratas rebeldes. El pirata minucioso debe hacer un recorrido tal que garantice que recorre todo el desierto, de modo que sin importar donde está el tesoro eventualmente pase por allí. Está vestido con color naranja. Cuando para, habla mucho.

1. Implemente este nuevo pirata. ¿cuáles métodos se sobre escriben (overriding)?

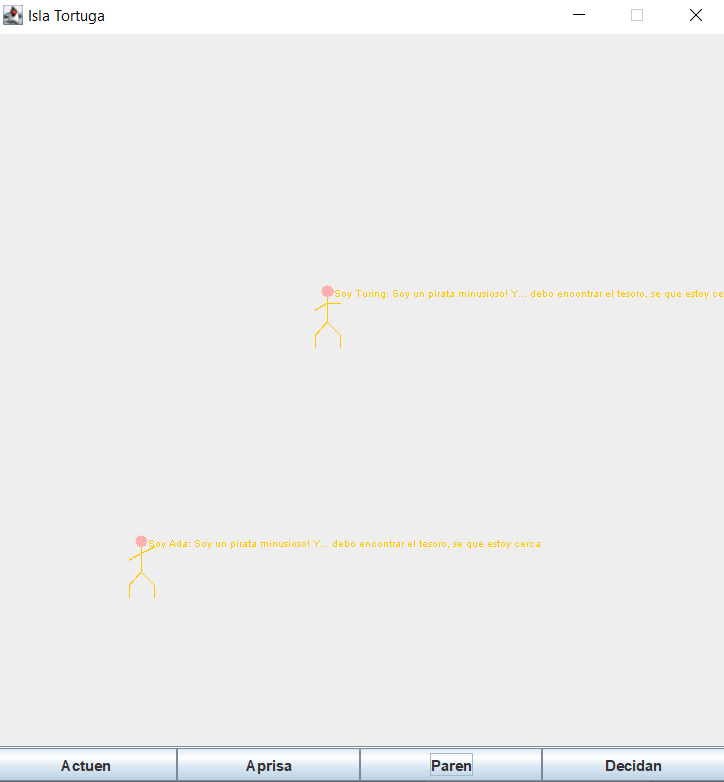
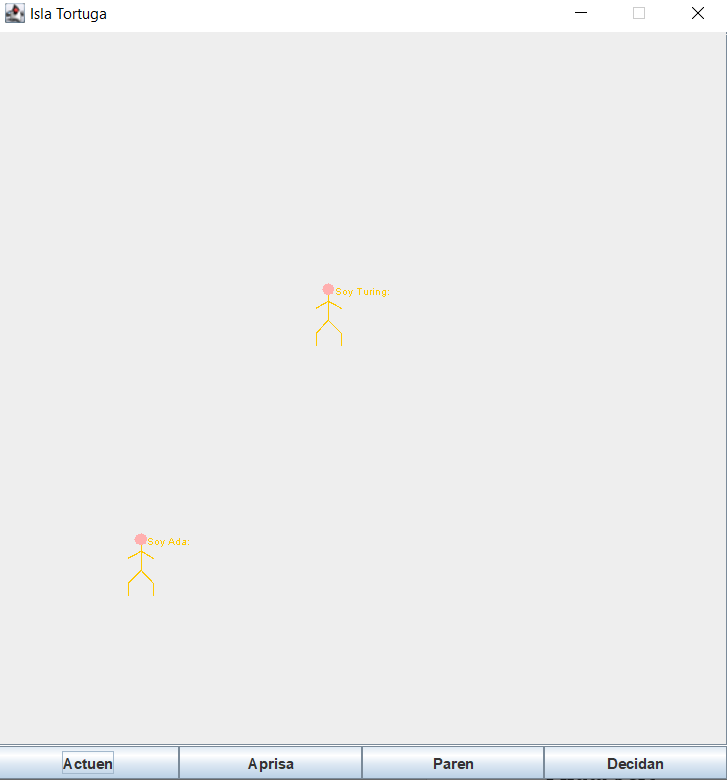
Se sobre escriben los métodos de actúe y pare.

1. Adicione una pareja de piratas minuciosos, llámelos, Ada y Turing, ejecute el programa y pídales a todos que actúen y que paren. Capture la pantalla. ¿Qué pasa?

Actúan como se esperaba; sus piernas vuelven a la posición inicial y se muestra una cantidad considerable de texto en su mensaje

1. Nuevamente ejecute el programa y pídales a todos que actúen, decidan y paren. Capture la pantalla. ¿Qué pasa?



Actúan como se esperaba; sus piernas vuelven a la posición inicial y se muestra una cantidad considerable de texto en su mensaje. Al presionar Decidan Turing decidió parar mientras que Ana decidió actuar.

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

El objetivo de este punto es permitir recibir en un nuevo tipo de pirata.

1. Propongan, describan e implementen un nuevo tipo de pirata.

Pirata perezoso

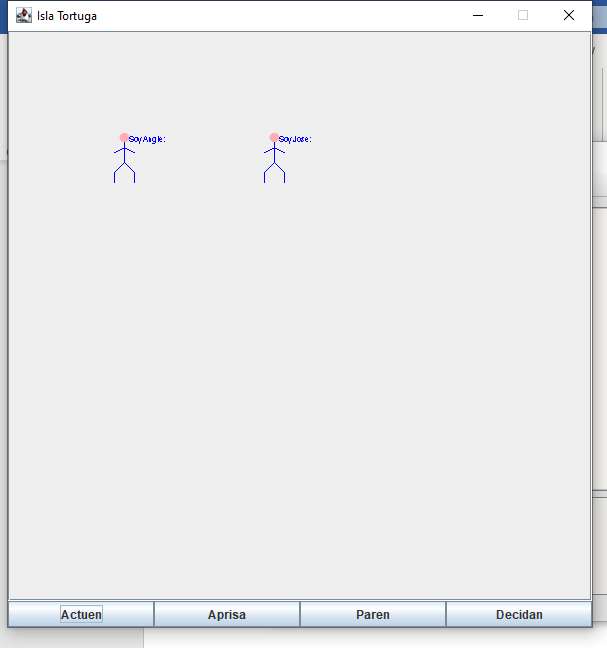
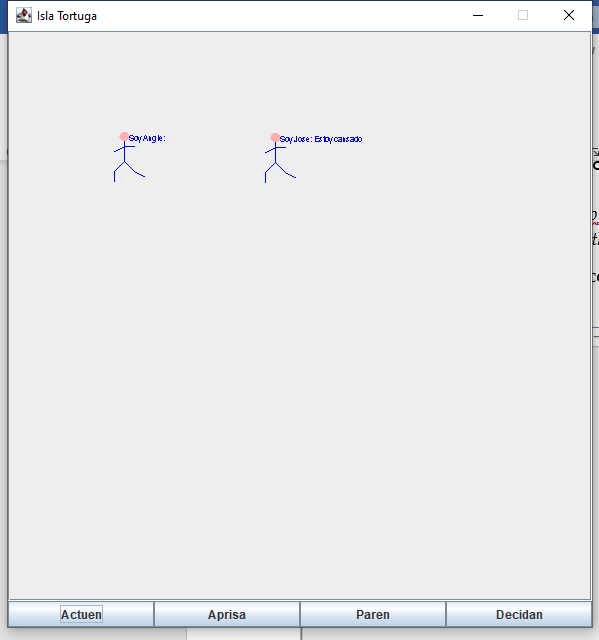
Los piratas perezosos, vestidos de azul, sólo actúan cada dos órdenes seguidas además de que tienen una cantidad especifica de veces que pueden actuar, cuando se les pide que decidan, siempre paran. En un principio no dicen nada, pero cuando se les acaba las veces que van a actuar paran y sus palabras son “Estoy cansado”.

1. Incluyan una pareja de ellos con el nombre de ustedes. Ejecute el programa con dos casos significativos. Explique la intención de cada caso y capture las pantallas correspondientes.

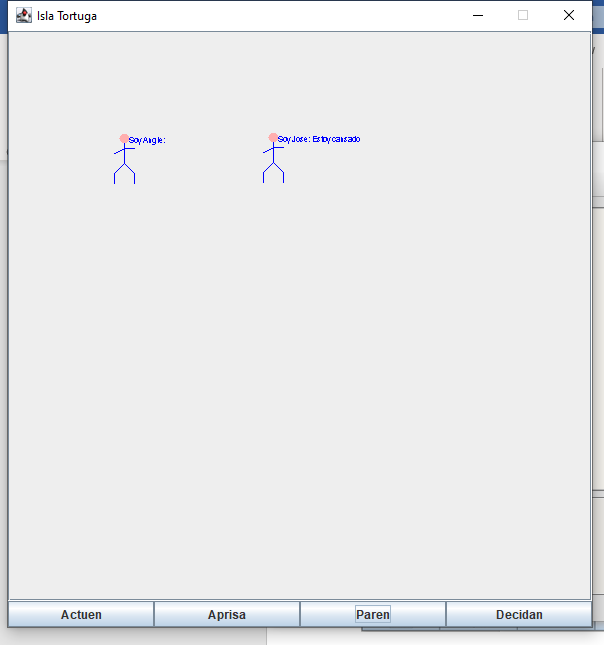
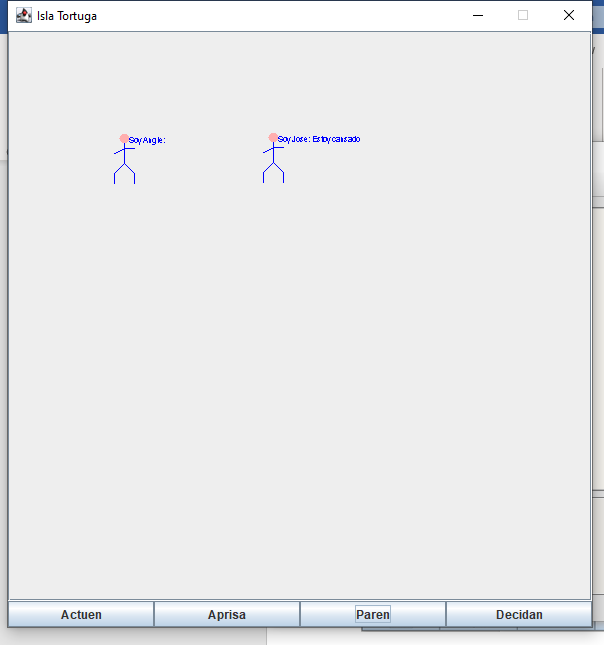
Se crearon la siguiente pareja:

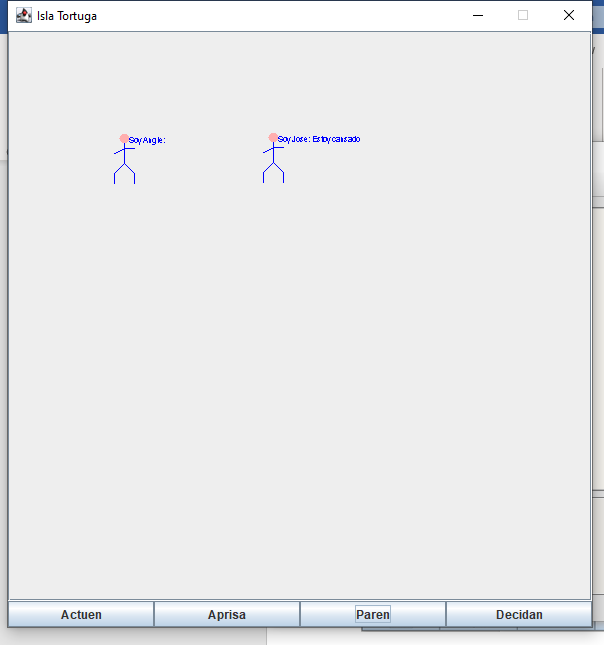
* + *Perezoso angie = new Perezoso(this, "Angie", 100, 400, 2);*
  + *Perezoso jose = new Perezoso(this, "Jose", 250, 400, 1);*

Se les va a pedir que actúen tres veces (para que realicen la acción una vez) y con esto *jose* debería decir que está cansado.

Después se les pedirá que decidan 3 veces, los dos no se van a mover en ninguna de las órdenes.



**Nuevo elemento: Proponiendo y diseñando**

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

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

Lagos: los lagos son localizaciones en las cuales los piratas perezosos podrán descansar si se llegan a encontrar cansados, una vez descansan la cantidad de acciones que pueden realizar antes de cansarse nuevamente incrementa en uno. Los lagos generalmente están calientes (Azul oscuro) y solo se podrá descansar en ellos cuando su temperatura disminuya (Azul claro). El tiempo de enfriamiento de un lago es aleatorio y puede durar como máximo 10 turnos.

Para descansar en un lago el pirata simplemente debe acercarse a este.



**Métodos agregados:**

**Lago:**

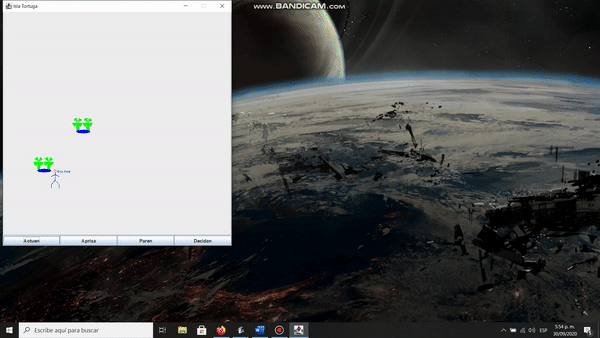
* public int getEnfriamiento()

**Isla:**

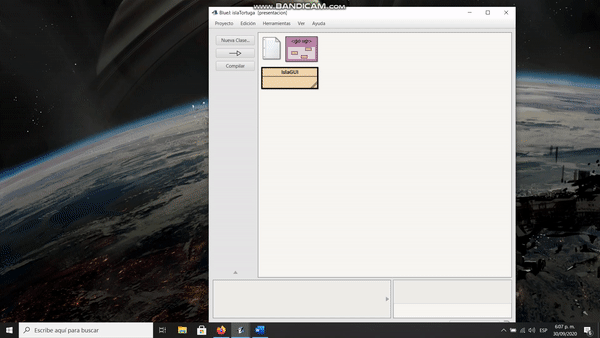
* public ArrayList<EnIsla> elementosCercanos(EnIsla elemento)

1. Incluyan dos de ellos con los nombres semánticos. Ejecuten el programa con dos casos significativos. Explique la intención de cada caso y capture las pantallas correspondientes.

* Los piratas pueden descansar en un lago:



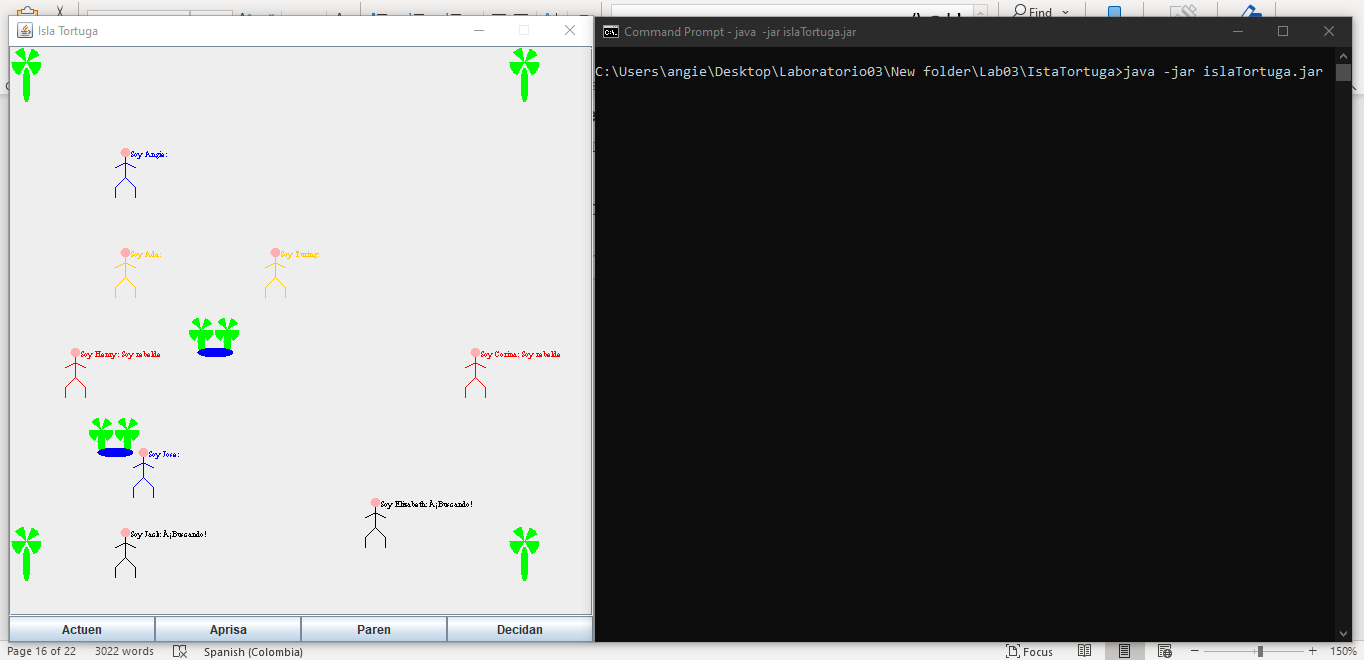
* Los lagos se enfrían cada cierto tiempo:



**Empaquetando la versión final para el usuario**

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. Ejecútelo ¿qué pasa?

Se abre la ventana grafica de la isla con los elementos que se habían puesto previamente en el método *algunosEnIsla.*



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

Además de ser una forma mucho más compacta de entregar la solución se ve claramente la correcta funcionalidad de esta.

También es más portable ya que solo es necesario que se tenga instalado la máquina virtual de Java (JVM).

**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**

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

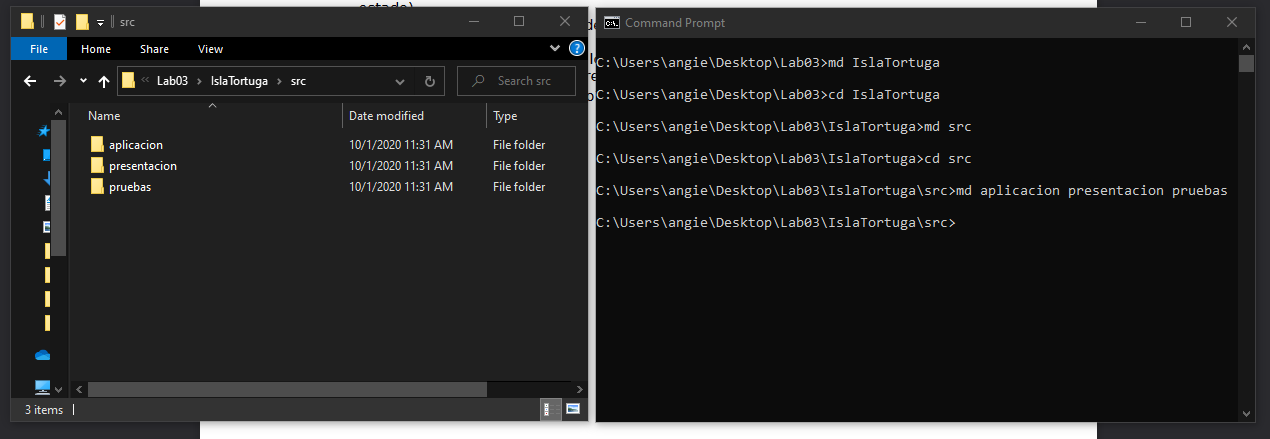
1. Investiguen los comandos para moverse en la estructura de directorios: crear, borrar, listar su contenido y copiar o eliminar un archivo.
   * **Crear:** *md* <nombredecarpeta>
   * **Borrar:** *del <archivo o carpeta> (*elimina el archivo o archivos de la carpeta, más no la carpeta
   * **Listar:** *dir (*lista los archivos en la carpeta*), Tree* <carpeta> (lista completa del árbol de directorios de una ruta concreta)
   * **Copiar:** *copy <origen> <destino> (*copia un archivo en la carpeta destino*), XCOPY <origen> [destino] [/A | /M] [/D[:fecha]] [/P] [/S [/E]] [/V] [/W] (*es una copia extendida, copia varios archivos, directorios completos*)*
   * **Eliminar:** *rd <directorio> (*elimina el 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 *IslaTortuga*

*src*

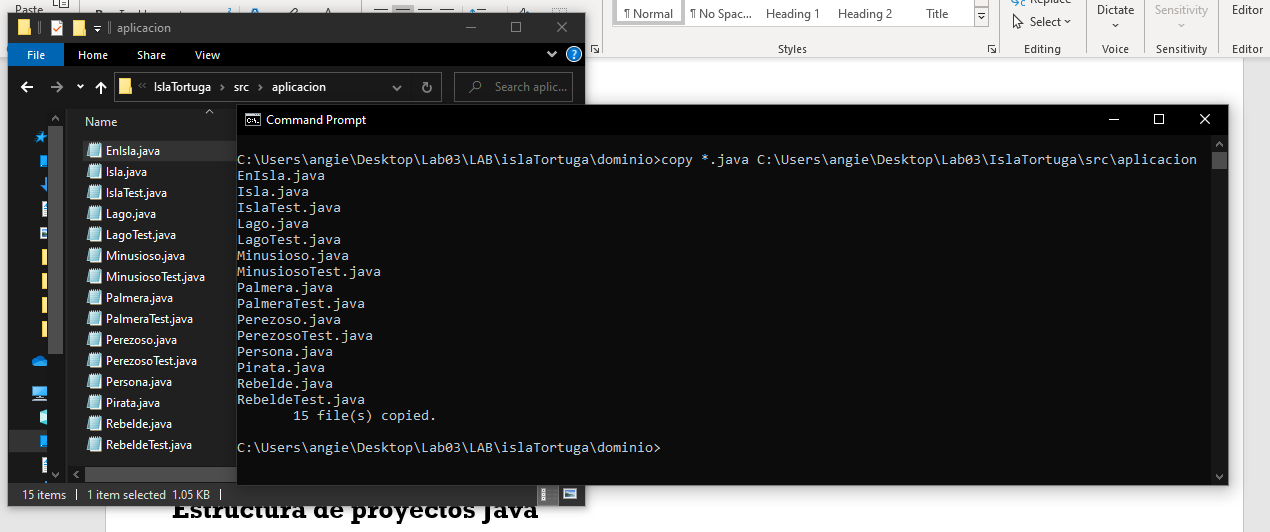
*aplicación*

*presentación*

*pruebas*



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



**Estructura de proyectos Java**

En java los proyectos se estructuran considerando tres directorios básicos.

*IslaTortuga*

*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:***Contiene los .java del proyecto, organizados internamente por paquetes
* ***Bin:***Contienes los .class del proyecto, organizados internamente por paquetes
* ***Docs:***Contiene la documentación del proyecto

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

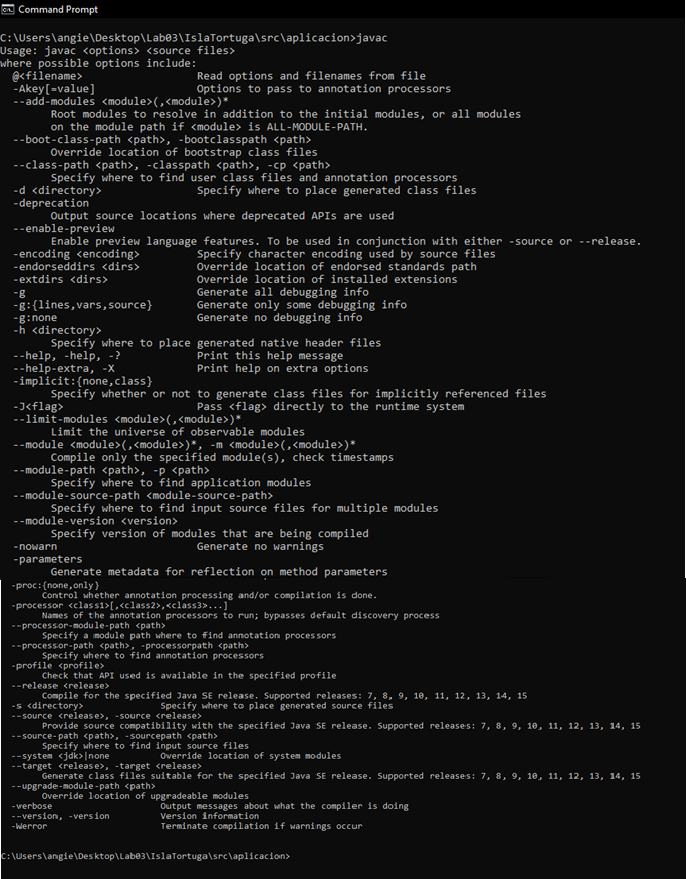
No se debería copiar nada, la carpeta bin contiene los archivos .class que son generados automáticamente al compilar el proyecto

**Comandos de Java**

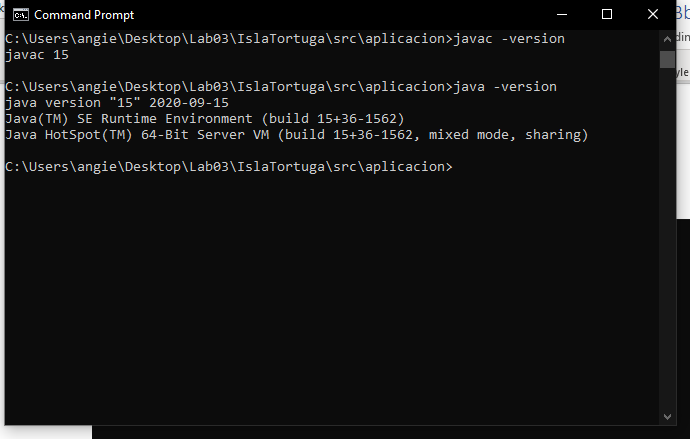
1. Consulte para qué sirven cada uno de los siguientes comandos:

* *javac:* Compila un archivo .java
* *java:* Ejecuta un archivo java
* *javadoc:* Genera la documentación de un archivo .java
* *jar:* Empaqueta un proyecto java

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



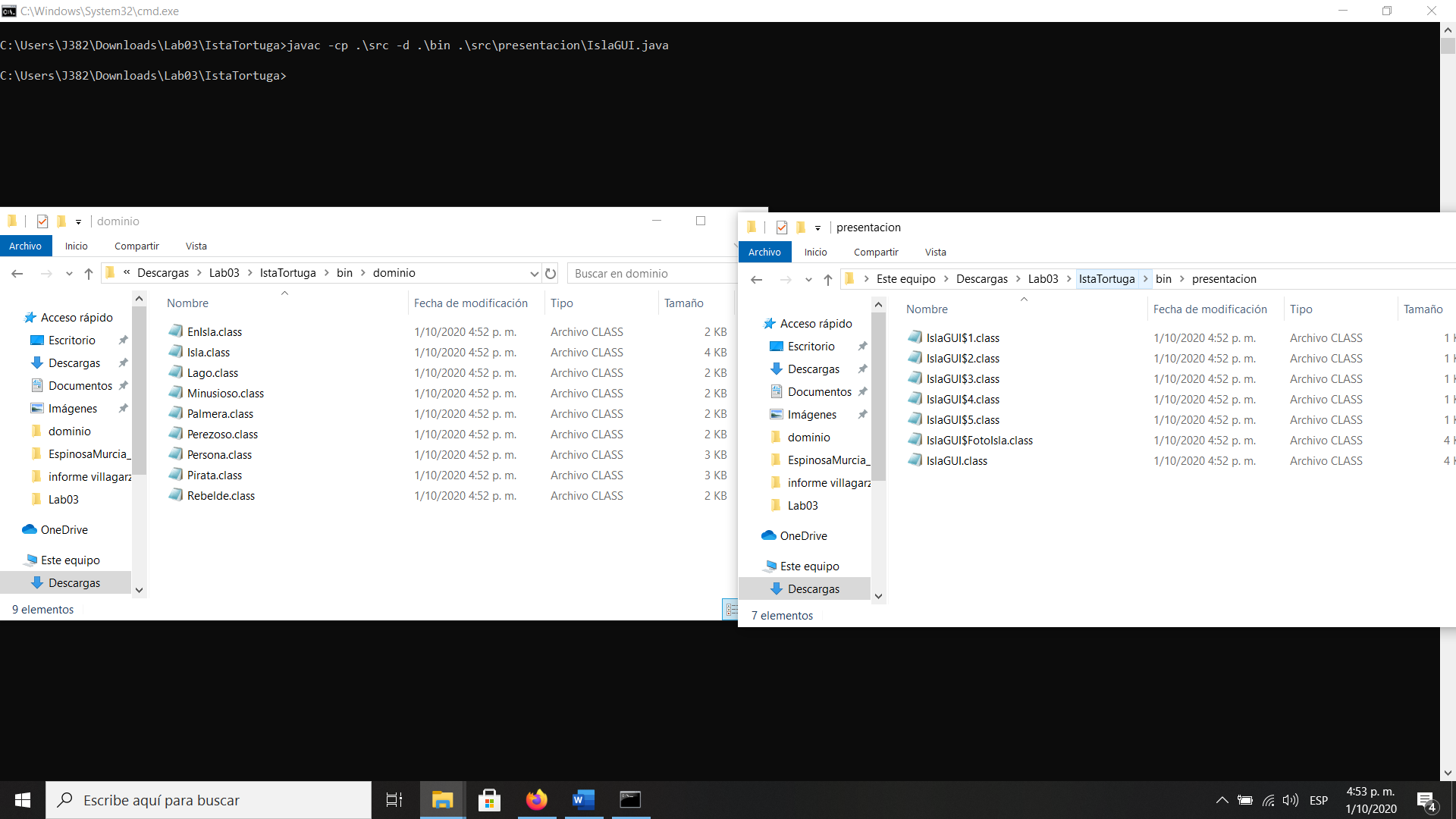
1. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.



**Compilando**

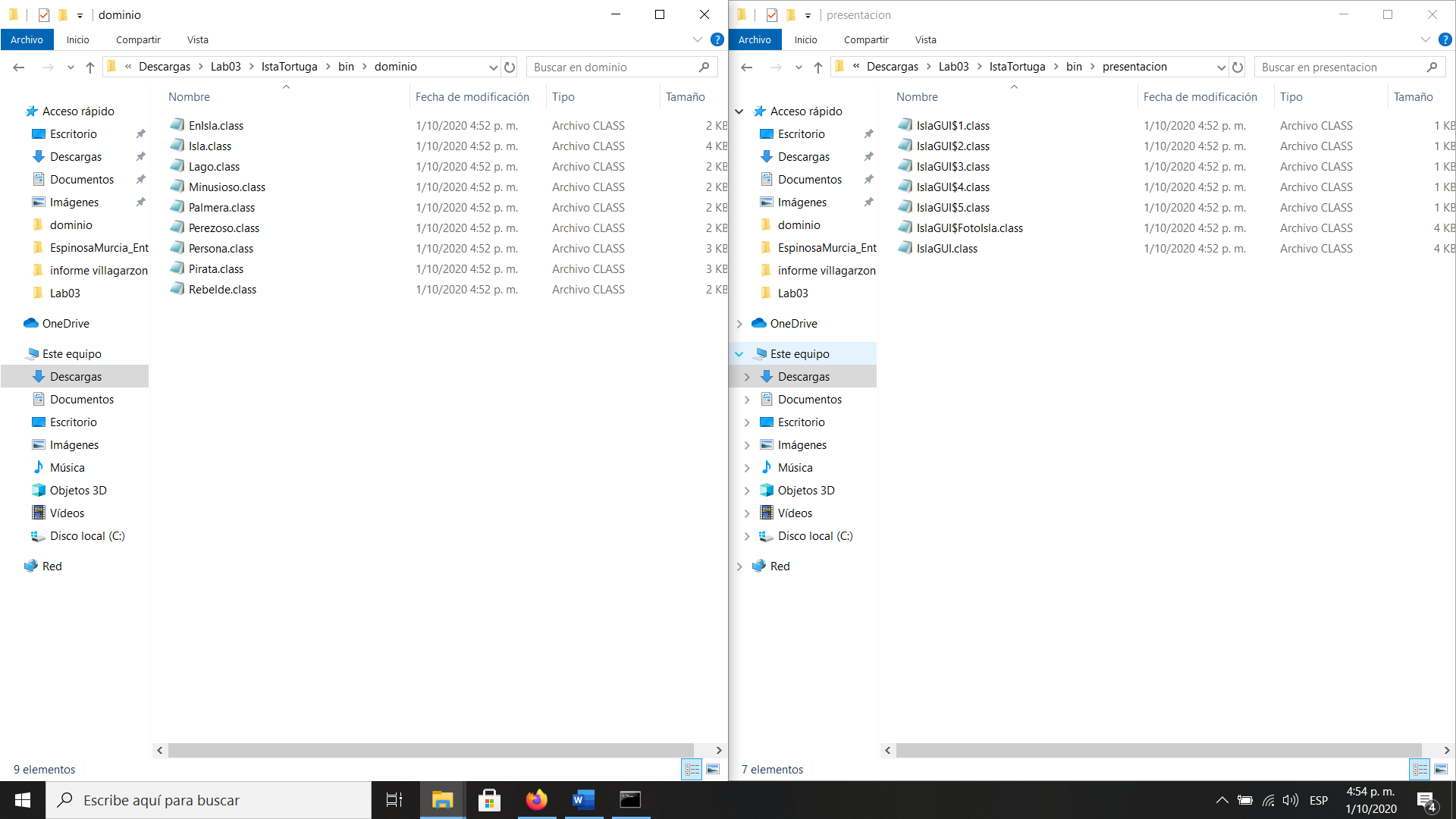
1. Utilizando el comando *javac*, desde el directorio raíz (desde automata con una sola instrucción), compile el proyecto. ¿Qué instrucción completa tuvo que dar a la consola para compilar TODO el proyecto? Tenga presente que se pide un único comando y que los archivos compilados deben quedar en los directorios respectivos.

Se uso: *javac -cp .\src -d .\bin .\src\presentacion\IslaGUI.java*



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

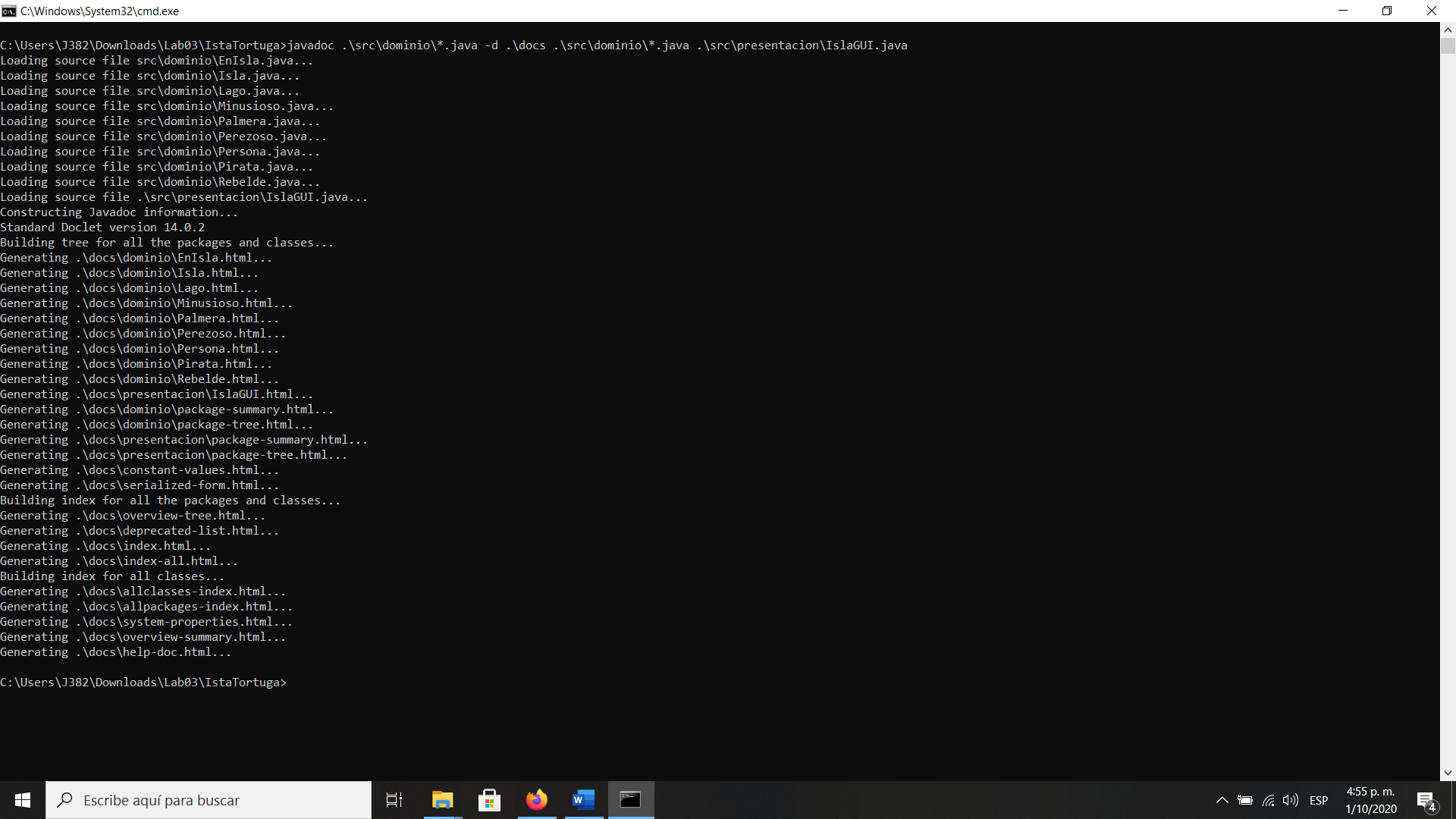
Los nuevos archivos tienen extensión .class, se encuentran en la carpeta bin organizados por paquetes



**Documentando**

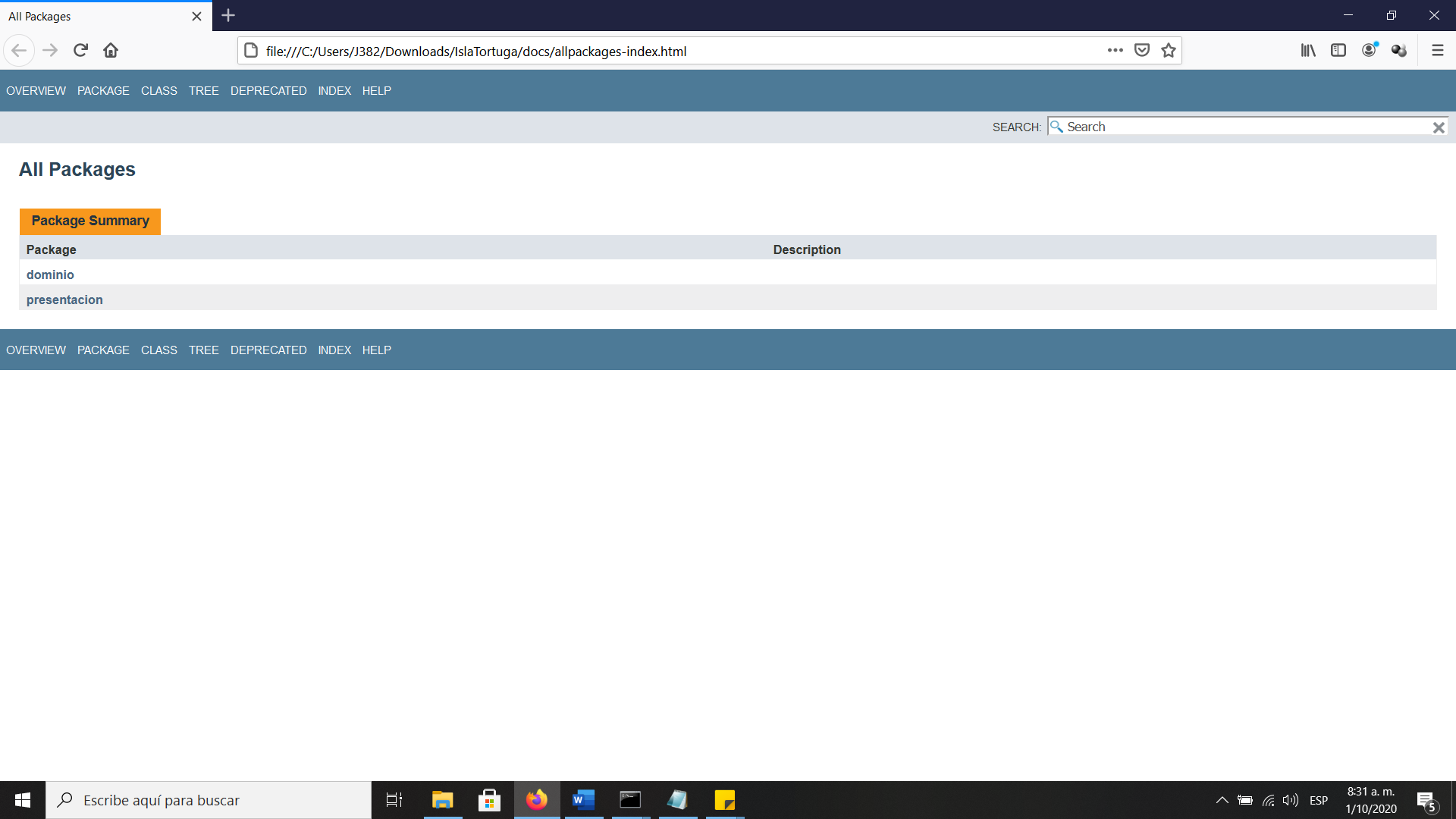
1. Utilizando el comando *javadoc*, desde el directorio raíz, genere la documentación (API)en formato html, en este directorio. ¿cuál es el comando completo para generar esta documentación?

Para generar toda la documentación se usa: *javadoc .\src\dominio\\*.java -d .\docs .\src\ \dominio\\*.java .\src\presentacion\IslaGUI.java*



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

Se debe abrir ***allpackages-index.html***



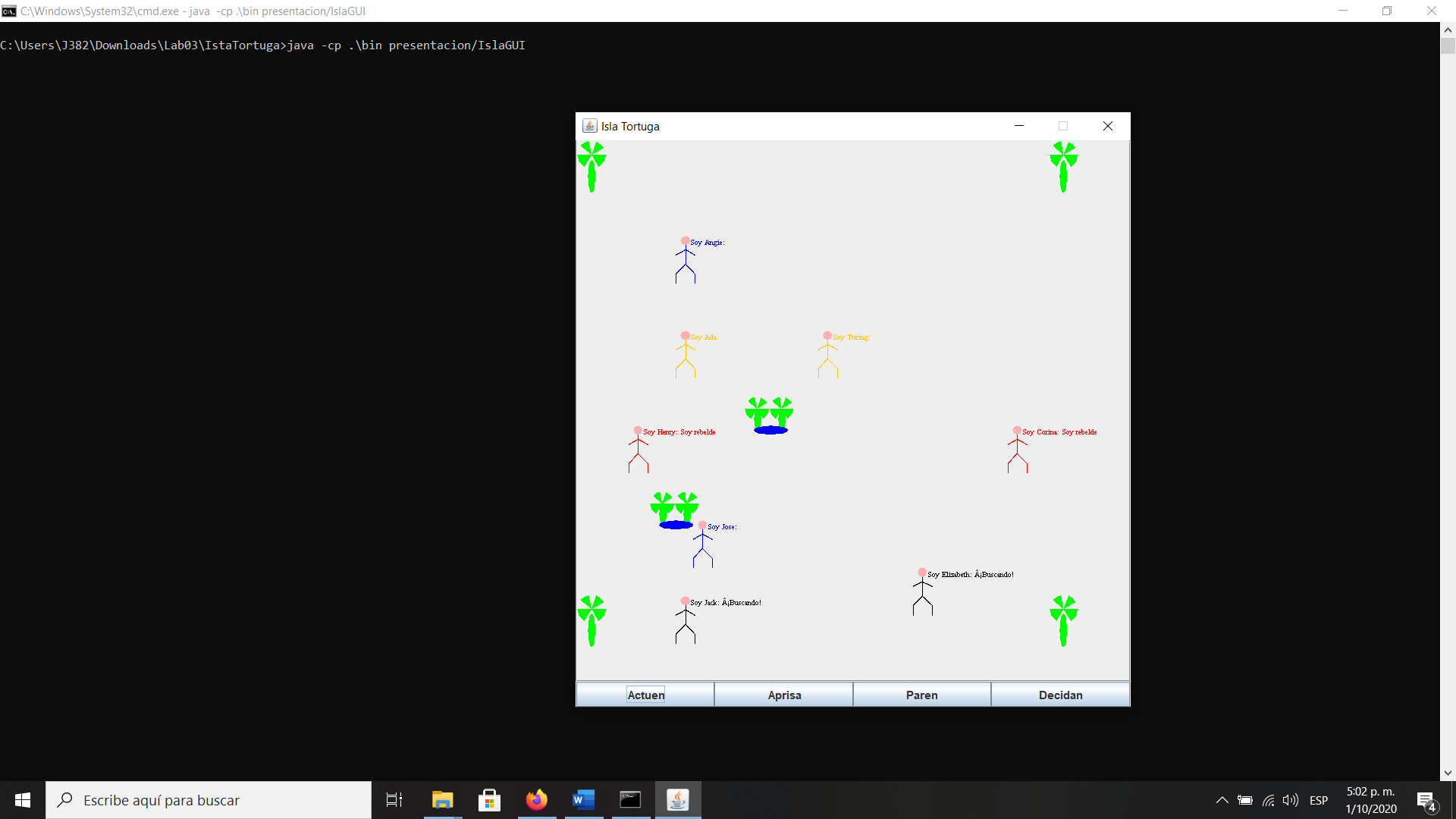
**Ejecutando**

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

**El comando es**: *java -cp .\bin presentacion/IslaGUI*

***cp:***Le indica a java en donde buscar las clases

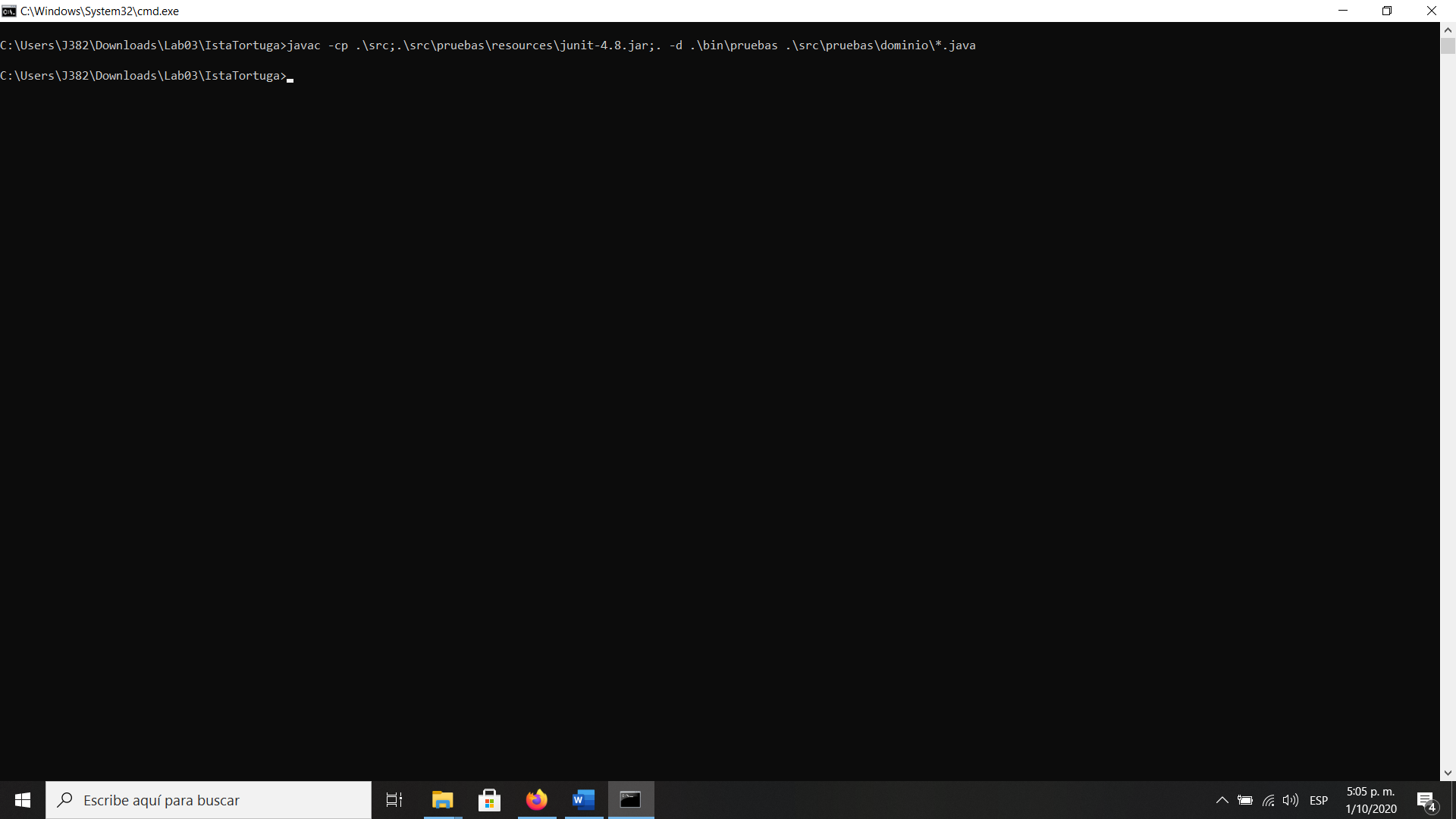
*presentacion/IslaGUI:* Es la clase que contiene el método main()



**Probando**

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?

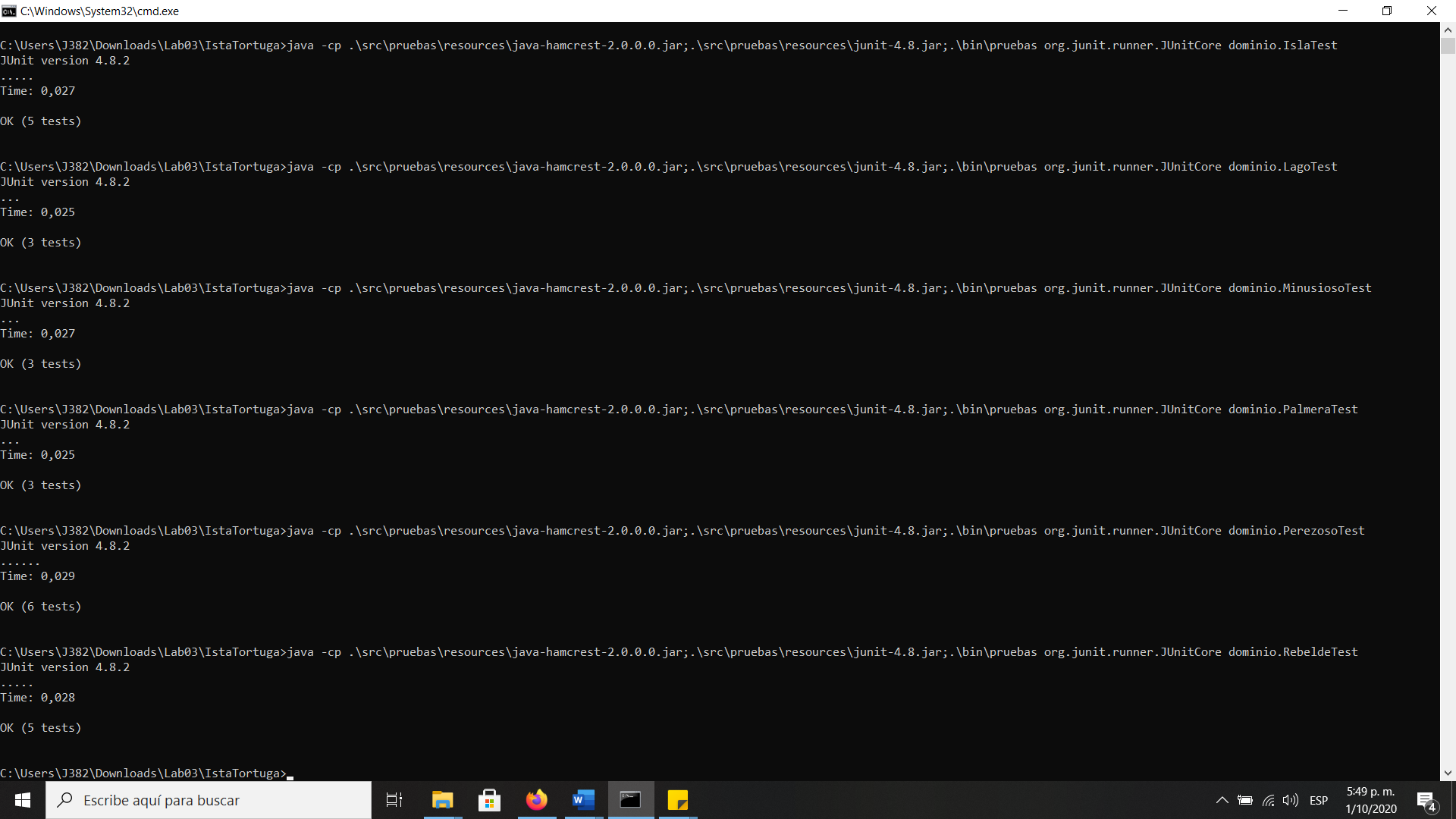
**Se utilizo:** *javac -cp .\src;.\src\pruebas\resources\junit-4.8.jar;. -d .\bin\pruebas .\src\pruebas\dominio\\*.java*



1. Ejecute 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>

**Se utilizó:** *java -cp .\src\pruebas\resources\java-hamcrest-2.0.0.0.jar;.\src\pruebas\resources\junit-4.8.jar;.\bin\pruebas org.junit.runner.JUnitCore dominio.RebeldeTest*

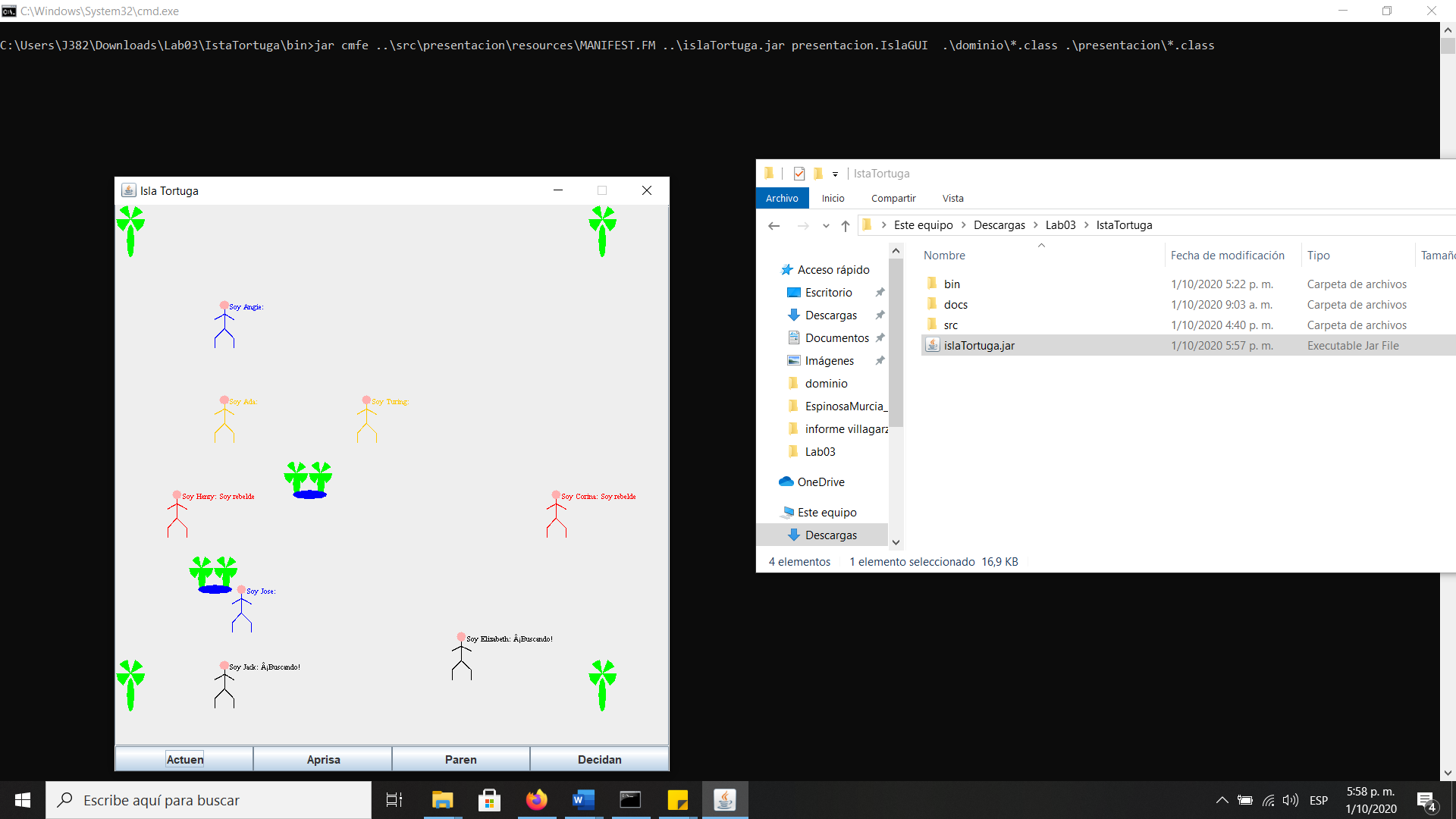
1. Pegue en su documento el resultado de las pruebas



**Empaquetando**

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*?

**Se empaqueto con la instrucción:** *jar cmfe ..\src\presentacion\resources\MANIFEST.FM ..\islaTortuga.jar presentacion.IslaGUI .\dominio\\*.class .\presentacion\\*.class* **ejecutada desde el directorio** *IslaTortuga\bin*



1. ¿Cómo se ejecuta el proyecto empaquetado?

Se puede ejecutar dando doble click en él o desde consola con el comando: *java -jar “nombre”.jar*

**RETROSPECTIVA**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?

Se trabajo con programación a pares aproximadamente 15 horas, 15 horas Angie Medina, Jose Perez.

1. ¿Cuál es el estado actual del laboratorio? ¿Por qué?

Consideramos que el laboratorio se realizó en su totalidad

1. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?

Integrates code at a time, pues no se generan problemas de versiones que quitan tiempo en un futuro

1. ¿Cuál consideran fue el mayor logro? ¿Por qué?

Completar de **Bluej A Consola** pues nuestro contacto previo con la línea de comando es poco y muchas cosas resultaron nuevas

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

Consideramos que trabajamos bien como equipo, nos comprometemos a mejorar el uso de las practicas xp vistas hasta ahora y empezar a usar las que se vieron en este laboratorio.