

PROGRAMACIÓN ORIENTADA A OBJETOS

Herencia e interfaces



ADEMÁS Java desde consola

2020-2

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. 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 : Coding  Use **collective ownership**
Coding  Only one pair **integrates code at a time**
6. 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.
- ➔ En la entrega inicial 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

En esta aplicación simularemos la situación de la isla Tortuga en donde un grupo de piratas están buscando un tesoro.

Conociendo [En lab03.doc y IslaTortuga.asta]

1. En el directorio descarguen los archivos contenidos en `islaTortuga.zip`. Revisen el código de la aplicación a) ¿Cuántos paquetes tiene? b) ¿Cuántas clases tiene en total? c) ¿Cuál es la clase con la que se ejecuta la aplicación? ¿Por qué?
2. Ejecuten el programa. ¿Qué funcionalidades ofrece? ¿Qué hace actualmente? ¿Por qué?

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

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

1. Consulte el significado de las palabras `package` e `import` de java. ¿Qué es un paquete? ¿Para qué sirve?
2. Revise el contenido del directorio de trabajo y sus subdirectorios. Describa su contenido. ¿Qué coincidencia hay entre paquetes y directorios?
3. Revisando las clases que contiene, ¿cuál es el propósito de cada uno de los paquetes encontrados?
4. Adicione el diseño de 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 Diagrama de Paquetes)

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

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?
3. Estudie la clase **Isla**. ¿Qué tipo de colección se usa para albergar los elementos? ¿Puede recibir piratas? ¿Por qué?
4. Estudie la clase **Persona**; ¿qué atributos pueden usar otras clases? ¿que atributos pueden usar sus subclases? ¿qué métodos no pueden cambiar las personas?
5. Estudie el código de la clase **Pirata**; ¿qué atributos pueden usar otras clases? ¿que atributos pueden usar sus subclases? ¿qué métodos no pueden cambiar los piratas? ¿qué métodos pueden cambiar parcialmente los piratas?
6. Estudie el código de la clase **EnIsla**; ¿qué atributos pueden usar otras clases? ¿qué métodos **deben** implementar las clases que están en isla?

Ciclo 1. Actuan y descansan los Piratas [En lab04.doc y *.java]

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

1. Estudie el código de la clase **Pirata** : ¿de qué color es? ¿qué mensaje tiene? ¿cómo actúan? ¿qué hacen cuando paran? ¿cómo deciden?. Explique claramente dónde está la información de las respuestas de cada una de las preguntas.
2. En el método **algunosEnIsla** de la clase **Isla** cree dos piratas en diferentes posiciones y acondiciónelos al isla llámelos **jack** y **elizabeth**. Ejecute el programa y capture la pantalla. ¿Qué pasa ahora? ¿Pídale que actuen? ¿Qué pasa? ¿Por qué?
3. 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.
4. En este punto vamos a construir el método que atiende el click del botón **Paren** de la interfaz: el método llamado **paren()** 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.
5. 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.

Ciclo 2. Incluyendo a los Piratas rebeldes [En lab04.doc, IslaTortuga.asta y *.java]

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

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, actuan 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)?
2. Adicione una pareja de piratas necios, llámelos **henry** y **corina**, ejecute el programa y pídale a todos que actúen y que paren. Capture la pantalla. ¿Qué pasa?
3. Nuevamente ejecute el programa y pídale a todos que actuen, decidan y paren. Capture la pantalla. ¿Qué pasa?

Ciclo 3. Adicionando palmeras [En lab03.doc, IslaTortuga.asta y *.java]

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.

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

1. ¿Qué debemos hacer para tener palmeras en la isla?
2. Construya la clase `Palmera` para poder añadirla en el `Isla`. ¿qué cambios incluyó Para aceptar este elemento , ¿debe cambiar en el código del `Isla`. en algo? ¿por qué?
3. Adicionen cuatro palmeras en las esquinas del `Isla`, llámenlas `superiorDerecha`, `superiorIzquierda`, `inferiorDerecha` y `inferiorIzquierda`. Ejecuten el programa pídale a todos que actúen, decidan y paren. Capturen la pantalla. ¿Qué pasa? ¿es correcto?

Ciclo 4. Creando un nuevo pirata: el minucioso [En lab03.doc, IslaTortuga.asta y *.java]

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.

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

1. Implemente este nuevo pirata. ¿cuáles métodos se sobre-escriben (overriding)?
2. Adicione una pareja de piratas minuciosos, llámelos `ada` y `turing`, ejecute el programa y pídale a todos que actúen y que paren. Capture la pantalla. ¿Qué pasa?
3. Nuevamente ejecute el programa y pídale a todos que actúen, decidan y paren. Capture la pantalla. ¿Qué pasa?

Ciclo 5. Nuevo pirata: Proponiendo y diseñando

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

(NO OLVIDE BDD - MDD)

1. Propongan, describan e implementen un nuevo tipo de pirata.
2. 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.

Ciclo 6. Nuevo elemento: Proponiendo y diseñando

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

(NO OLVIDE BDD - MDD)

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 capture las pantallas correspondientes.

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

1. Revise las opciones de `BlueJ` para empaquetar su programa entregable en un archivo .jar. Genere el archivo correspondiente.
2. Consulte el comando `java` para ejecutar un archivo jar. Ejecutelo ¿qué pasa?
3. ¿Qué ventajas tiene esta forma de entregar los proyectos? Explique claramente.

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.
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
    aplicacion
    presentacion
    pruebas
```

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

Estructura de proyectos java [En lab03.doc]

En java los proyectos se estructuran considerando tres directorios básicos.

```
automata
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 consulte y capture el contenido del directorio que modificó.

Comandos de java [En lab03.doc]

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

```
javac
java
javadoc
jar
```

2. Cree una sesión de consola y consulte en línea las opciones de los comandos java y javac. Capture las pantallas.
3. Busque la opción que sirve para conocer la versión a que corresponden estos dos comandos. Documente el resultado.

Compilando [En lab03.doc]

1. Utilizando el comando javac, **desde el directorio raiz (desde 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.
2. Revise 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 que abrir para empezar a navegar por la documentación? Ábralo y capture la pantalla.

Ejecutando [En lab03.doc]

1. Empleando el comando `java`, desde el directorio raíz, 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: <http://junit.sourceforge.net/doc/cookbook/cookbook.htm>
3. Pegue en su documento el resultado de las pruebas

Empaquetando [En lab03.doc]

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

RETROSPECTIVA

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