PROGRAMACIÓN ORIENTADA A OBJETOS
Herencia e interfaces
ADEMAS Java desde consola
2019-01
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 : Use collective ownership. 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 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

`En este laboratorio simularemos un gimnasio: BodyTic`

Conociendo [En lab03.doc y bodyTic.asta]

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

Arquitectura general. [En lab03.doc y automataasta]

1. Consulten el significado de las palabras package e import de java. Explique su uso con un ejemplo de este programa.
2. ¿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.
3. Revisen el contenido del directorio de trabajo y sus subdirectorios. Describa su contenido. ¿Qué coincidencia hay entre paquetes y directorios?

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

1. Usando ingeniería reversa. Presente el diseño actual de la **capa de aplicación.**
2. ¿Qué observaciones haría al respecto?

Ciclo 1. Iniciando con los deportistas normales [En lab03.doc y *.java]

(NO OLVIDE BDD - MDD)

1. Estudie la clase Salon ¿Qué tipo de colección usa para albergar los elementos? ¿Puede recibir deportistas? ¿Por qué?
2. Estudie el código de la clase Deportista, ¿qué otros componentes software la definen? ¿cómo?

3. Por ser una Persona ¿qué saben hacer? ¿qué no puede hacer distinto? ¿qué deben aprender a hacer? Justifique su respuesta.
4. Por comportarse como un EnSalon, ¿qué sabe hacer? ¿qué no puede hacer distinto? ¿qué debe aprender a hacer? Justifique su respuesta.
5. Considerando lo anterior, un Deportista ¿de qué color es? ¿cómo se mueve? ¿cómo decide? ¿cómo inicia ? Justifiquen sus respuestas.
6. Construyan el método que atiende el botón **`Entren:`** el método llamado entrada() de la clase Salon. En este método creen dos deportistas edward y bella. Ejecuten el programa, ¿Cómo quedan os deportistas? Capturen una pantalla significativa.
7. Construyan el método que atiende el botón **`Inicien:`** el método llamado inicio() de la clase Salon. Ejecuten el programa Ejecute el programa y pídales que entren e inicien. ¿Cómo se comportan los deportistas? Capturen una pantalla significativa.
8. Construyan el método que atiende el botón **`Paren:`** el método llamado parada() de la clase Salon. Ejecute el programa y pídales que entren, inicien y paren ¿Cómo se comportan los deportistas? Capturen una pantalla significativa.
9. Construyan el método que atiende el botón **`Decidan:`** el método llamado decision() de la clase Salon. Ejecute el programa y pídales que entren, inicien, decidan y paren ¿Cómo se comportan los deportistas? Capturen una pantalla significativa.
10. Construyan el método que atiende el botón **`Salgan:`** el método llamado salida() de la clase Salon. Ejecute el programa y pídales que entren, inicien, decidan, paren y salgan ¿Cómo se comportan los deportistas? Capturen una pantalla significativa.

Ciclo 2. Incluyendo a los deportistas avanzados [En lab03.doc y automataasta]

(NO OLVIDE BDD - MDD)

El objetivo de este punto es recibir en el salón deportistas avanzados. Los deportistas avanzados, vestidos de naranja, se mueven más, sólo paran después de dos órdenes y, cuando les piden que deciden, siempre hacen ejercicio.

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.
2. Adicionen una pareja de deportistas avanzados en la fila 3, llámenlas neo y trinity, Ejecute el programa y pídales que entren, inicien, decidan y paren. Capturen pantallas significativas. ¿Qué pasa?
3. Ahora, los avanzados quieren sorprender con su resistencia; es decir, sólo son paran muy pocas veces. ¿Qué modificaría para lograr este comportamiento? ¡Hágalo!
4. Nuevamente ejecute el programa y pídales a todos que entren, inicien, decidan y paren. Capturen una pantalla significativa. ¿Qué pasa?

Ciclo 2. Incluyendo a los deportistas habladores [En lab03.doc y automataasta]

(NO OLVIDE BDD - MDD)

El objetivo de este punto es recibir en el salón deportistas habladores. Los deportistas habladores están vestidos de naranja; al ejercitarse saltan (van subiendo los pies lentamente y luego los baja) y estándo arriba se mueven a la derecha, y habla poco; y cuando descansan hablan mucho.

1. ¿Cuáles son las extensiones necesarias en el diseño? ¿y los cambios? ¡Hágalos! ¿cuáles métodos se sobre-escriben *(overriding)*?
2. Implemente al deportista hablador.
3. 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?

Ciclo 3. Adicionando bolas de pilates [En lab03.doc, bodyTic.asta y *.java]

El objetivo de este punto es incluir en el `Salon` bolas de pilates especiales de colar (sólo vamos a permitir un tipo de bolas). Los bolas se encienden cuando inician, se apagan cuando paran y decien de manera estándar.

(NO OLVIDE BDD – MDD)

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.
2. Para aceptar este elemento , ¿debe cambiar en el código del `Salon` en algo? ¿por qué?
3. Adicionen dos b`olas` cerca en las esquinas del `Salon`, llámenlas `suroeste` y `noreste`, ¿Escriba la prueba correspondiente.
4. Nuevamente ejecute el programa y pídales a todos que entren, inicien, decidan y paren. Capturen una pantalla significativa. ¿Qué pasa?

Ciclo 4. Nueva Deportista: Proponiendo y diseñando

El objetivo de este punto es permitir recibir en un nuevo tipo de deportista (NO OLVIDE BDD - MDD)

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

El objetivo de este punto es permitir recibir en un nuevo elemento (no deportista) en el `Salon`.

(NO OLVIDE BDD - MDD)

1. Propongan, describan e Implementen un nuevo tipo de elemento
2. 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. [En lab03.doc, bodyTic.asta , *.java, bodyTic.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. ejecútenlo ¿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 Capturenn el contenido de su directorio

```
bodyTic
     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.

```
bodyTic
       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 Capturen el contenido del directorio que modificó.

Comandos de java [En lab03.doc]

1. Consulten 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. Capturen 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 `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 comando y que los archivos compilados deben quedar en los directorios respectivos.
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. 3Tenga 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)
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?
4. ¿Cuál consideran fue el mayor logro? ¿Por qué?
5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?
6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?