Programación Orientada a Objetos.

Práctica 01.

Objetivos.

- Conocer en términos generales la documentación de la API de Java y poder examinar algunas clase de la misma.
- Crear una clase sencilla para probar conceptos básicos como la abstracción y el encapsulamiento.
- Crear la estructura inicial de un sistema bancario que se usará como ejemplo en el resto del curso.

Ejercicio 1. Utilización de la Documentación de la Java API.

1) Utilizando un navegador de Internet, abra la página "**Java™ Platform, Standard Edition 8 API Specification"** que se encuentra en el URL:

http://docs.oracle.com/javase/8/docs/api/

Nota. Es recomendable descargar esta documentación para poder utilizarla aún cuando no se cuente con conexión a Internet.

2) Seleccione el paquete java.text de la lista del frame superior del lado izquierdo.

3) Seleccione la clase NumberFormat en la lista de clases del frame inferior izquierdo.

4) Lea la descripción de la clase en el frame del lado derecho para aprender a utilizar esta clase, que será usada en ejercicios posteriores.

5) Familiarícese con la documentación de la API de Java, dedicando tiempo a leer la descripción de otras clases que le parezcan interesantes.

Ejercicio 2. Abstracción y encapsulamiento simples.

Versión 1.

1) Baje de la página del curso, sección Material, el archivo Practica01.zip al escritorio y descomprímalo para obtener el material necesario para esta práctica.

2) Cree un nuevo proyecto de Netbeans llamado Practica01 de tipo Java Application. **Deseleccione la casilla Create Main Class**.

3) Dentro del proyecto Practica01, cree un paquete llamado version1.

4) En el paquete version1, cree una clase llamada Vehicle con los siguientes elementos:

    a. Dos atributos públicos de tipo double: load, que representa el peso del vehículo y maxLoad, que es la carga máxima que puede tener.
    b. Un constructor público que reciba un argumento de tipo double. En el cuerpo del constructor inicialice el atributo maxLoad con el valor del argumento pasado al constructor.
    c. Dos métodos públicos (getters): getLoad() que entregue el atributo load y getMaxLoad() que entregue el atributo maxLoad.

    Se asume que los datos están expresados en Kilogramos.

5) Incluya en el proyecto Practica01 paquete version1 el programa TestVehicle.java proporcionado en el subdirectorio Practica01/ejercicio2/version1 copiándolo directamente (fuera de Netbeans) al directorio c:\...\Practica01\src\version1.
Estudie el código del programa. Note que encuentra problemas cuando se añade la última caja al vehículo porque el código no checa si se excede la carga máxima.

6) Ejecute el progarma TestVehicle y observe el resultado.

Versión 2.

1) En el proyecto Practica01 cree un nuevo paquete llamado version2.

2) Dentro de este paquete, cree otra clase llamada Vehicle con los siguientes elementos:

    a. Modifique los atributos load y maxLoad para que sean privados.

b. Añada el método public boolean addBox() que tome como argumento el peso de la caja en Kilogramos y verifique que el añadir este peso no exceda la carga máxima. Si ocurre esta violación, la caja se rechaza entregando un valor booleano false. Si no hay problema el método debe entregar true.

Se asume que los datos están expresados en Kilogramos.

3) Incluya en el proyecto Practica01 paquete version2 el nuevo programa TestVehicle.java proporcionado en el subdirectorio Practica01/ejercicio2/version2 de la misma manera que en el ejercicio anterior y estudie su contenido. Note que el programa a diferencia del de la versión 1, no puede modificar el atributo load directamente, sino usar el método addBox, que regresa un valor de verdadero o falso, el cual es desplegado.

4) Ejecute el programa TestVehicle y observe el resultado.

Versión 3.

Ahora, suponga que se van a hacer ciertos cálculos para determinar el programa de mantenimiento de los vehículos, estos cálculos se deben hacer con los pesos de la carga expresados en newtons en vez de kilogramos.

1) En el proyecto Practica01 cree un nuevo paquete llamado version3.

2) Dentro de este paquete, cree otra clase llamada Vehicle, copiando la desarrollada en la versión 2.

3) Modifique el constructor y los métodos getLoad, getMaxLoad y addBox para convertir los parámetros que vienen en kilogramos a newtons. La conversión se debe hacer llamando a los métodos privados kiloToNewts y newtsToKilos que reciben como argumento el peso en kilogramos y entregan como resultado el peso en newtons.

4) Escriba los métodos privados kiloToNewts y newtsToKilo que lleven a cabo las conversiones. Para convertir kilos a newtons multiplique el peso por 9.8. Para convetir newtons a kilos divida entre 9.8. Note que ahora los datos internamente están en newtons aunque los parámetros se reciben en kilogramos.

5) Incluya en el proyecto Practica01 paquete version3 el nuevo programa TestVehicle.java proporcionado en el subdirectorio Practica01/ejercicio2/version3 de la misma manera que en el ejercicio anterior y estudie su contenido. Note que el programa es idéntico al de la version 2.

6) Ejecute el progarma TestVehicle y observe el resultado. No debe haber ninguna diferencia en la salida, lo cual demuestra que los cambios hechos a la implementación de Vehicle no afectan la interfaz pública.

Ejercicio 3. Creación de la estructura básica de un Sistema Bancario.

En este ejercicio se crea una primera versión de la clase Account, que modela las cuentas de un Banco. Se proporciona un programa TestBanking que crea un objeto cuenta de clase Account con referencia account. TestBanking inicializa el saldo de la cuenta y hace algunas transacciones simples. Finalmente, despliega el saldo de la cuenta.

1) Cree un nuevo proyecto de Netbeans llamado BankLtd de tipo Java Application. **Deseleccione la casilla Create Main Class**.

2) Dentro del proyecto BankLtd, cree un paquete llamado banking.

3) En este paquete cree el archivo Account.java que implemente la clase Account con las siguientes características:

    a. Un atributo privado de tipo double llamado balance.
    b. Un constructor público que reciba un argumento initBalance que inicialice el atributo balance.
    c. Un método public, getBalance() que entregue el atributo balance.
    d. Un método público void, deposit(), que reciba una cantidad y la sume al saldo.
    e. Un método public void, withdraw(), que reciba una cantidad y la reste del saldo.

4) Incluya en el proyecto BankLtd paquete banking el programa TestBanking.java proporcionado en el subdirectorio Practica01/ejercicio3 de la misma manera que en el ejercicio anterior y estudie su contenido.

5) Ejecute el programa TestBanking y observe el resultado. Note que en esta primera versión del sistema bancario no se hace ningún tipo de verificación en cuanto a las cantidades a retirar de manera que la cuenta se podría sobregirar.

Diagrama de Clases.