Programación Orientada a Objetos.

Práctica 03.

Objetivos:

- Practicar las estructuras de control de los programas orientados a objetos.
- Construir ciclos anidados utilizando funciones de manipulación de Strings.
- Proseguir con el desarrollo del sistema bancario.

Antes de iniciar los ejercicios, baje de la página del curso, sección Material, el archivo Practica03.zip al escritorio y descomprímalo para obtener el archivo necesario para esta práctica.

Ejercicio 1. Utilización de estructuras de control.

1. Cree un nuevo proyecto de Netbeans llamado Practica03 de tipo Java Application. Deseleccione la casilla Create Main Class.

2. Dentro del proyecto Practica03, cree un paquete llamado ejercicio1.

3. En el paquete ejercicio1 escriba una clase de Java llamada FooBarBoz que examine los 50 primeros enteros positivos y los imprima uno por línea. Para cada número que sea múltiplo de 3, se debe imprimir en la misma línea que el número, la palabra “foo”. Para cada número que sea múltiplo de 5, la palabra “bar” y para cada múltiplo de 7, la palabra “boz”.

La salida del programa debe ser así:

1
2
3 foo
4
5 bar
6 foo
7 boz
8
9 foo
10 bar
11
12 foo
13
14 boz
15 foo bar
16

etc. etc.

Nota. Se sugiere utilizar el método **System.out.print()**; es muy parecido a System.out.println(), la diferencia es que el primero no inserta \n antes de imprimir, mientras que println() sí lo hace.

Ejercicio 2. Uso de ciclos anidados.

1. Dentro del proyecto Practica03, cree un nuevo paquete llamado ejercicio2.

2. En este paquete, cree una nueva clase de Java llamada Buscador.

3. En el programa Buscador, escriba un método llamado haySubString(String subTexto, String texto) que busque una cadena específica (subTexto) dentro de otra cadena (texto). El método debe entregar true si la cadena buscada existe dentro de la otra cadena. Por ejemplo:

   haySubString("El", "El gato sobre el tejado...) debe regresar true.
   haySubString("La", "El gato sobre el tejado...) debe regresar false.

4. En el mismo paquete, escriba un programa de prueba llamado TestBuscador cuyo método main cree un objeto de clase Buscador y pase varios strings y substrings al método haySubString() e imprima el resultado del método haySubString().

Notas.

a. Puede utilizar los métodos **charAt(int index)** y **length()** de la clase String. Consulte como funcionan en la documentación de la API de java.

b. En la API de Java existe un método llamado **contains()** que realiza precisamente la función solicitada. En este ejercicio no se permite usar dicho método.

Ejercicio 3. Modificaciones a métodos del Sistema Bancario.

1. Abra el proyecto de netbeans BankLtd que se ha venido desarrollando hasta ahora.

2. Borre la clase TestBanking de la versión anterior del proyecto.

3. Modifique la clase Account para incluir la siguiente nueva funcionalidad:

   a) El método deposit() debe entregar el valor booleano true siempre. (Todos los depósitos son aceptados).

   b) El método withdraw() debe checar ahora que la cantidad del retiro no sea mayor que el saldo de la cuenta, y regresar false si el retiro causara sobregiro de la cuenta. Si no es así, se deberá afectar el saldo y regresar true.

4. Copie la nueva versión del archivo TestBanking.java proporcionado en el directorio mod04/ejercicio3 al proyecto Practica03, paquete banking.

5. Estudie el program TestBanking sobre todo en lo concerniente al formato de la variable amount.

6. Ejecute el programa TestBanking y examine el resultado.

7. Modifique la cantidad a retirar (variable amount) en el programa TestBanking, ejecútelo de nuevo y cheque que efectivamente no permita retiros cuando el cliente no tiene fondos suficientes.

Diagrama de Clases.