Unidad 2

DIPLOMATURA EN PROGRAMACION JAVA

Ejercicios

Capítulo 3

Control del Programa y Vectores

Capítulo 3

Ejercicio 1

En este ejercicio se utilizará un ciclo y sentencias de control de flujo

Crear una aplicación que realiza 50 ciclos e imprima en una línea separada cada valor de la variable de control del ciclo. Se deberá imprimir además de este valor para cada múltiplo de 3 “mTres”, para cada múltiplo de 5 “mCinco” y para cada múltipo de 7 “mSiete”

Por ejemplo:

1
2
3 mTres
4
5 mCinco
6 mTres
7 mSiete
8
9 mTres
10 mCinco
11
12 mTres
13
14 mSiete
15 mTres mCinco
16

etc...

1. Crear un espacio de trabajo en el directorio Capítulo 3 y llamarlo Ejercicio 1
2. Crear un proyecto Java y llamarlo Ejercicio 1 también
3. Crear una única clase de Java con un método main en el cuál se maneje el ciclo for y donde la variable de iteración se llame número. Llamarla Ciclos
4. Dentro del ciclo for, realizar salidas por consola utilizando el método System.out.print cuando sea conveniente. Este método a diferencia de println no imprime un carácter de nueva línea.
5. Dentro del ciclo for utilizar una sentencia if para determinar cuándo deben y cuando no imprimirse “mTres”, “mCinco” y “mSiete”. Sugerencia: el operador % calcula el resto de la división entera.

6. Por último dentro del mismo ciclo for, imprimir el carácter de nueva línea. Esto lo puede hacer utilizando el carácter de control "\n"

Ejercicio 2

En este ejercicio se adquirirá experiencia en declarar, crear y manipular vectores de una dimensión de tipos primitivos

Realizar los siguientes pasos:

1. Crear un espacio de trabajo en el directorio Capítulo 3 y llamarlo Ejercicio 2
2. Crear un proyecto Java y llamarlo Ejercicio 2 también
3. Crear una clase de aplicación (con el método main) y llamarla PruebaDeVectores. En el método main() declarar dos variables de referencia a vector de enteros llamadas vector1 y vector2.
4. Inicializar el vector1 con un bloque de inicialización (valores entre llaves) que contenga los ocho primeros números primos: 2, 3, 5, 7, 11, 17 y 19.
5. Mostrar por pantalla el contenido de vector1: Utilizar el método imprimeVector que se muestra a continuación:

```
public static void imprimeVector(int[] vector) {
	System.out.print('<');
	for (int i = 0; i < vector.length; i++) {
	// Imprimir un elemento
		System.out.print(vector[i]);
	// Imprimir una coma para delimitar si no es el último elemento
		if ((i + 1) < vector.length) {
			System.out.print(", ");
		}
	}
	System.out.print('>');
}
```

6. Compilar y ejecutar el programa
7. Asignar la variable vector1 a la variable vector2. Modificar cada elemento de vector2 para que su contenido sea igual a su número de índice (Ej: vector2[0]=0), teniendo en cuenta que consta de siete elementos.
8. Imprimir vector1 y verificar que pasó con su contenido.

Ejercicio 3

En este ejercicio se adquirirá experiencia en declarar, crear y manipular vectores de dos dimensiones de tipos primitivos

Realizar los siguientes pasos:

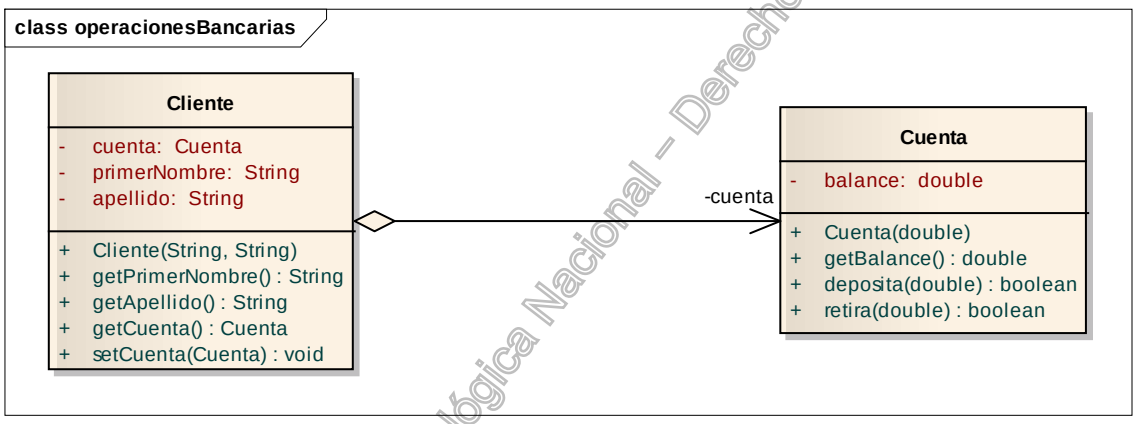
1. Crear un espacio de trabajo en el directorio Capítulo 3 y llamarlo Ejercicio 3

2. Crear un proyecto Java y llamarlo Ejercicio 3 también
3. Crear una clase de aplicación (con el método main) y llamarla PruebaDeVectores. Declarar una variable llamada matriz de tipo entero (int[][] – un vector de vectores de enteros)
4. Asignar elementos a la matriz de manera de producir la siguiente salida:

```
matriz[0] es <>
matriz[1] es <0>
matriz[2] es <0, 2>
matriz[3] es <0, 3, 6>
matriz[4] es <0, 4, 8, 12>
```

Ejercicio 4

Se debe modificar los métodos deposita y retira para retornar un valor booleano que especifique si la transacción fue satisfactoria



1. Abrir el espacio de trabajo del directorio Capítulo 3\Ejercicio 4
2. Modificar la clase cuenta para que condicione los métodos deposita y retira
   a. Modificar el método deposita para retornar true (significa que el depósito fue satisfactorio)
   b. Modificar el método retira para que verifique que la cantidad que se intenta retirar no sea mayor al balance actual. Si la cantidad es menor que el balance, descontar dicha cantidad y retornar `true`, caso contrario no descontar nada de balance y devolver `false`

Compilar y ejecutar el main de la clase PruebaOperacionesBancarias para verificar que la salida sea la siguiente:

```
Creando al cliente Juan Perez.
Creando una cuenta con 500.00 de balance.
Retira 150.00:true
Deposita 22.50:true
```

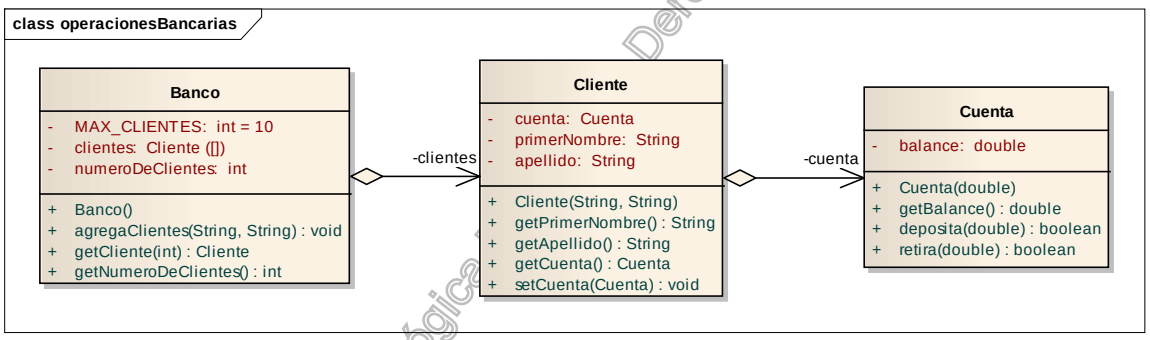
```
Retira 247.62:true
La cuenta tiene un balance de 124.88
Cliente [Perez, Juan] tiene un balance de 124.88
Intentando retirar 200:false
La cuenta tiene un balance de 124.88
Cliente [Perez, Juan] tiene un balance de 124.88
```

Ejercicio 5

En este ejercicio se implementará la multiplicidad que existe en la asociación entre un banco y sus clientes

Realizar los siguientes pasos:

1. Abrir el espacio de trabajo del directorio Capítulo 3\Ejercicio 5. Los errores que figuran se deben a las modificaciones que deberán realizarse.
2. Crear la clase Banco dentro del paquete operacionesBancarias. Cuando se cree un objeto del tipo banco se creará una relación entre él y los objetos del tipo Cliente. Esta se implementa como una agregación en un vector de objetos del tipo Cliente.



3. Agregar dos atributos a la clase Banco: clientes (un vector de objetos del tipo Cliente) y numeroDeClientes (un entero con el que se sabe cuál es el próximo cliente en el vector)
4. Agregar un constructor público que inicialice el vector cliente con una cantidad de elementos superior a 5.
5. Añadir el método agregaClientes(), en él se deberá construir los objetos del tipo Cliente con sus respectivos parámetros (nombre y apellido), colocarlo dentro del vector cliente e incrementar en uno el atributo numeroDeClientes
6. Agregar el método de acceso getNumeroDeClientes() el cual retorna el atributo numeroDeClientes
7. Agregar el método getCliente() el cual retornará el objeto del tipo Cliente según el índice que se pase como parámetro.
8. Compilar y ejecutar el programa.
9. Verificar que la salida sea:

```
Cliente [0] es Juan Pérez
Cliente [1] es Pedro García
Cliente [2] es Oscar Toma
Cliente [3] es Maria Soley
```