Programación Avanzada. Curso 2011
Juan Manuel Fernández Peña
Lectura de datos y Main

En este documento se analiza la manera de leer datos desde la consola de ejecución y desde archivos de texto. Adicionalmente se introduce el uso del MÉTODO Main.

El ejemplo utiliza las clases relacionadas con el Banco que se presentan en el archivo "Ejemplo de agregación".

Lectura desde la consola.
La consola de ejecución, herencia de MS-DOS, la hemos empleado para mostrar datos usando el método System.out.println(). System.out representa la consola de salida; en forma similar, System.in representa la consola de entrada. Desgraciadamente fue definida para leer caracteres de uno en uno, lo cual resulta incómodo. Java ofrece diversos mecanismos que toman como entrada la consola y permiten un manejo más sencillo. En este ejemplo se utiliza la clase Scanner, que agrupa los datos y los regresa en unidades separadas (el separador básico es el espacio, pero se pueden definir otros). Scanner proporciona una serie de funciones que permiten tomar los elementos uno a uno, como cuerdas o como números, saber si existen más elementos, etc.

Lectura desde archivo de texto.
La consola de entrada es un caso particular de los archivos de texto, por lo cual se puede usar el mismo mecanismo de arriba para leer en archivos. La única diferencia es que debe crearse un objeto File que se conecta al archivo deseado.

Si no se especifica otra cosa, el archivo de datos debe estar en el mismo directorio donde se ejecuta el programa. Si se usa Eclipse, será el folder del proyecto; si se ejecuta a pié, será el folder donde se halla la clase inicial o la jarra que contiene el programa.

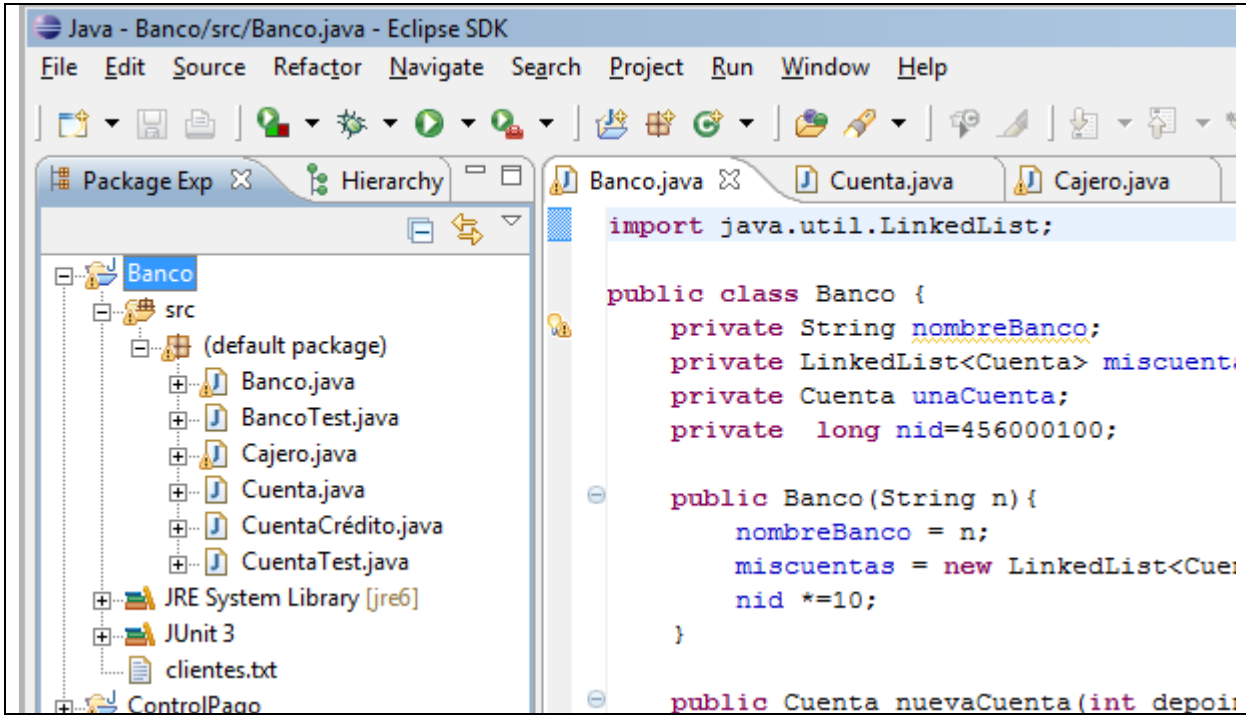
Para el ejemplo usaremos un archivo de texto llamado "Clientes.txt" en el folder del proyecto:

Método Main.
Cualquier clase de Java puede ser la clase inicial para lanzar una aplicación. Para que pueda hacerlo se le agrega un método de tipo void llamado "main" con un arreglo de strings como parámetro, como herencia de formas antiguas de lanzar programas:

```
public static void main(String[] args)
```

Note la palabra 'static', que indica que es único, para toda la clase. La clase inicial ejecuta primero que nada su método main, el cual gobernará el funcionamiento de la aplicación. El directorio donde se encuentra esa clase inicial será el directorio base de la aplicación.

Aunque el método main puede incluir mucho código, por sus características es inadecuado; es como realizar un programa tradicional de hace treinta años. Por ello su contenido debe reducirse al mínimo, que incluye crear un objeto inicial y cederle el control, ya sea a través de su constructor o de un método.



Observe el archivo Clientes.txt en el Folder Banco (casi hasta abajo)

Ejemplo Cajero.

En el ejemplo de abajo se construyó una clase inicial que representa un cajero que atiende pedidos de depósitos y retiros de un conjunto de cuentas. La clase Cajero tiene funciones de interfaz con los seres humanos y, por ello, se encarga de la entrada y salida. Al mandarse a ejecutar Cajero, el main crea un objeto Cajero y le cede el control a través de su constructor; este carga una lista de cuentas desde un archivo de texto y luego entra en un ciclo de atención a clientes, quienes le dan una orden seguida de un identificador de cuenta y una cantidad, como “Deposita 4560001004 500” o “Retira 4560001005 250”. También acepta la orden “Fin” que suspende su operación y “Ayuda” que lista los comandos disponibles.

```
import java.io.File;
import java.util.Scanner;

public class Cajero {

	private Banco ban;
	private int numctas = 0;

	public Cajero(String nomban, String lista){
		ban = new Banco(nomban);
		if (cargaCuenta(lista)>0)
			atiende();
	}

	private int cargaCuenta(String lis){
		int res=-1;
		String lin;
		Scanner lee;
		String  id;
		int sald;
		Cuenta kue;

		try {
			File lect = new File(lis);
			lee = new Scanner(lect);
			while (lee.hasNext()){
				sald = lee.nextInt();
				kue = ban.nuevaCuenta(sald);
				numctas++;
				System.out.println("Cuenta "+numctas+"
con id: "+kue.getIdentificador()+" tiene $"+kue.getSaldo());
			}
			} catch (Exception ee) {
				System.out.println("Problema al abrir el
archivo de entrada "+ee);
			}
			res = numctas;
		return res;
	}

	private void atiende(){
		Scanner conso = new Scanner(System.in);
		String opera, ayuda, id;
		int kant;
		Cuenta kue;
		ayuda = "Operaciones: Deposita, Retira, Ayuda, Fin";
		System.out.println(" ");
```

En el método cargaCuentas se lee un archivo de texto que contiene listas de saldos iniciales de las cuentas que serán creadas y quedarán a cargo del Banco.

Note el uso de File para conectarse al archivo,
El Scanner para separar elementos
Las funciones hasNext() para saber si hay más elementos, nextInt() para tomar el siguiente elemento como número entero

El método atiende crea un Scanner que se conecta a la consola de entrada

```
            System.out.println("     *** Cajero listo ***");
            System.out.println(ayuda+"\n");
            while (true){
                  opera = conso.next();
                  if (opera.equals("Fin")) {
                        System.out.println("     *** Cajero se
cierra ***");
                        break;
                  }
                  if (opera.equals("Ayuda"))
                        System.out.println(ayuda+"\n");
                  else{
                  id = conso.next();
                  kant = conso.nextInt();
                  System.out.println("Leido "+opera+", "+id+",
"+kant);
                  if (opera.equals("Deposita")){
                        kue=ban.buscaCuenta(id);
                        kue.deposita(kant);
                        System.out.println("Cuenta "+id+" nuevo
saldo $"+kue.getSaldo());
                  }
                  else
                        if (opera.equals("Retira")){
                              kue = ban.buscaCuenta(id);
                              System.out.println("cuenta "+id+"
retira "+kue.retira(kant)+" quedando saldo= $"+kue.getSaldo());
                        }
                        else
                              System.out.println("   >>>
Instrucción no reconocida <<<");
                  }
            }
      }

      /**
       * @param args
       */
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            Cajero atm = new Cajero("Banco de Veracruz",
"clientes.txt");
      }

}
```

Lugo tiene un ciclo infinito donde espera clientes
Trae una cuerda que representa la operación deseada

Si es “Fin” se rompe el ciclo (break)

En deposita o retira lee otra cuerda que es el id del banco para esa cuenta
Y luego un entero que es la cantidad

Finalmente se conecta con la clase Banco y le pide actuar

El método main se reduce a crear el cajero inicial y dejar en sus manos el control de la aplicación