Capítulo

05

Estructura
de un Programa

Mauricio Diéguez Rebolledo

Ania Cravero Leal

Departamento de Ingeniería de Sistemas
Facultad de Ingeniería, Ciencias y Administración

"Proyecto financiado por el Fondo de Desarrollo Educativo de la Facultad de Ingeniería, Ciencias y Administración de la Universidad de La Frontera"

Versión

1.0

TEMARIO

Estructura de un Programa

En este capítulo trataremos las reglas para implementar los algoritmos diseñados en los capítulos anteriores, utilizando un determinado Lenguaje de programación. Estudiaremos la estructura de los lenguajes y sus reglas sintácticas y semánticas básicas.

5.1 Introducción

Como ya has visto en los capítulos anteriores, el desarrollo de un programa computacional requiere de tres grandes etapas: Análisis, Diseño y Construcción. Los diseños que has creado anteriormente, a través de los algoritmos, ahora deben ser traducidos a un Lenguaje que el computador sea capaz de interpretar y ejecutar, recuerda que el objetivo de un programa consiste en proveer al computador de una serie de instrucciones para que éste pueda trabajar de forma autónoma. Luego, es de vital importancia que las instrucciones que le entregues al computador, estén en un formato que sea reconocible por él.

Para lograr esto, existen los denominados "Lenguajes de Programación", que nos permiten escribir el algoritmo diseñado en un lenguaje familiar al computador. A este proceso, se le denomina codificación. Existen muchos lenguajes de programación, desarrollados desde los años 50's, cada cual enfocado a ámbitos diferentes, como por ejemplo, científico, comercial, Web, entre otros.

La utilización de estos lenguajes, requiere, al igual que cualquier lenguaje, que se respeten las reglas sintácticas y semánticas propias (ver en sección 5.2.1 a), por lo tanto es necesario que conozcas estas reglas y las estructuras necesarias para construir tus programas.

Existen herramientas, denominadas IDE (ver capítulo 2), que permiten codificar un algoritmo en algún lenguaje de programación determinado. Además, estas herramientas, permiten detectar errores, modificar código y ejecutar un programa.

5.2 Lenguajes de Programación

5.2.1 Conceptos Básicos

a. Lenguaje de Programación:

Un lenguaje de programación, puede definirse como "un sistema notacional para describir computaciones en una forma legible tanto para la máquina como para el ser humano" (Louden, 2004).

De acuerdo a lo anterior podemos entender que un lenguaje de programación es un lenguaje conformado por un conjunto de símbolos y reglas sintácticas y semánticas, diseñado para representar las instrucciones que deben ser ejecutadas por el computador, es decir, un lenguaje de programación es el lenguaje a través del cual podemos comunicar al computador, las instrucciones que conforman nuestro algoritmo para su ejecución.

Para esto, debemos considerar que un lenguaje de programación está compuesto por:

- Un conjunto finito de símbolos que forman el denominado Léxico del lenguaje, el cual contiene el vocabulario permitido por el mismo.
- Un conjunto de "reglas gramaticales", denominada Sintaxis del lenguaje, con la cual se construyen las sentencias correctas en el programa.
- Por último la Semántica del lenguaje, reglas que asocian un significado, o definición de lo que se debe hacer, a la construcción del lenguaje.

5.2.2 Características del Lenguaje

Como se ha indicado en capítulos anteriores, en este libro se revisa el lenguaje de programación JAVA.

Lenguaje JAVA:

JAVA es un lenguaje de programación orientado a objetos, que nace como parte de un proyecto de Sun Microsystems, denominado Green, a principios de la década de los 90. Es un lenguaje basado en C++, pero con un modelo de objetos más simples y sin elementos de bajo nivel, como la manipulación de punteros.

En la actualidad Java se utiliza para el desarrollo de aplicaciones empresariales de gran escala, para aplicaciones Web, aplicaciones para aparatos domésticos y celulares, entre otros.

Una de las características más significativas de los programas construidos en JAVA, es que son multiplataforma, es decir, pueden ser ejecutados en cualquier sistema operativo.

5.3 Estructura y Sintaxis del Programa

5.3.1. Sintaxis

Una sintaxis define la forma en que deben ser escritos los programas, se pueden considerar como las "reglas gramaticales" del lenguaje. Como se explicó anteriormente, cada lenguaje posee una sintaxis propia, distinta a la de otros lenguajes. La sintaxis de un lenguaje está conformada por los siguientes elementos:

a. Identificadores:

Elemento léxico de un lenguaje de programación, que sirve para identificar elementos o entidades dentro de un programa. Se puede decir que el identificador es el nombre de un elemento. A considerar:

- Un identificador debe comenzar siempre con una letra
- Las comillas ("), el guión (-) y los espacios en blanco, no se utilizan en los identificadores

b. Palabras Claves:

Es un identificador, pero que ha sido definido por el lenguaje y no por el programador, por lo tanto no puede ser usado en un elemento definido por el programador.

c. Variables:

Elemento definido por el usuario que modifica su valor a través de la ejecución del programa. El nombre de una variable está dado por un identificador.

d. Constantes:

Elemento definido por el usuario que no modifica su valor a través de la ejecución del programa. El nombre de una constante está dado por un identificador.

e. Operadores:

Símbolos que indican cómo deben manejarse determinados elementos (operandos) dentro de un programa, es decir, como deben relacionarse estos elementos en una instrucción. Los operandos pueden ser constantes, variables o llamados a funciones. Existen operadores aritméticos, relacionales y lógicos

Tabla 1: operadores básicos

Tipo	Lenguaje JAVA
Aritméticas	
Suma	+
Resta Multiplicación	-
División	/
Resto de la división	%
Relacionales	
Mayor que	>
Mayor o igual que	>=
Menor que	<
Menor o igual que	<=
Igual	==
distinto	!=
Lógicos	
AND	&&
OR	||
Not	!

f. Expresiones:

Es la unión de los operandos con los operadores. Por ejemplo:

y = x + z	**x >= y**	**(x < z) && (y >= x + z)**

g. Funciones:

Son un conjunto de sentencias e instrucciones enfocadas a dar solución a un problema específico dentro del programa. Las funciones pueden ser utilizadas muchas veces durante la ejecución de un programa. Existen dos grupos de Funciones, Las funciones intrínsecas y las funciones definidas por el usuario.

Las primeras se refieren a aquellas funciones que están incorporadas en el lenguaje de programación, como por ejemplo la función raíz cuadrada o la función potencia, las cuales ya vienen definidas en el lenguaje de programación, por lo que no se necesitan que se construyan. Las segundas se refieren a aquellas funciones que construye un usuario para dar solución a un problema específico, por ejemplo una función que calcule el sueldo del trabajador a partir de sus datos (cargo, comisiones, bonos, descuentos, etc.).

La sintaxis de una función es:

Identificador_Función(argumentos)

El argumento de una función, son aquellos valores que la función necesita para calcular un valor.

Tabla 2: Ejemplos de Funciones Intrínsecas

Tipo	Lenguaje JAVA
Raíz Cuadrada	sqrt(valor)
Potencia	pow(valor)
Seno	sen(angulo)
Coseno	cos(angulo)

h. Inicio-Cierre:

Se refiere a la identificación del comienzo o término de un bloque de código que se encuentra dentro de una misma estructura. Ya sea el inicio-cierre de un programa, de un ciclo o de una estructura de control selectiva.

i. Indentación:

Se refiere al uso de la sangría dentro del código, para ordenar y mejorar la legibilidad del programa. Consiste en mover bloques de código hacia la derecha a través de espacios o tabulaciones.

Ejemplo 5.1:

```
Instrucción 1
Instrucción 2
        Instrucción 3
Instrucción 4
```

j. Comentarios:

Texto que se utiliza para describir determinados bloques de instrucciones dentro del programa. Por lo general se utilizan para explicar la función o tarea de un determinado código. Lo que se encuentra entre comentario no es revisado por el compilador.

En JAVA existen tres formas de realizar comentarios, denominados comentarios de implementación.

- Utilizando los símbolos //, este tipo de comentario abarca una línea, por ejemplo

```
// Este es un comentario de una línea
```

- Utilizando los símbolos /* */, este tipo de comentario abarca más de una línea, por ejemplo

```
/* comentario que abarca más de una línea,
el final lo marca el símbolo */
```

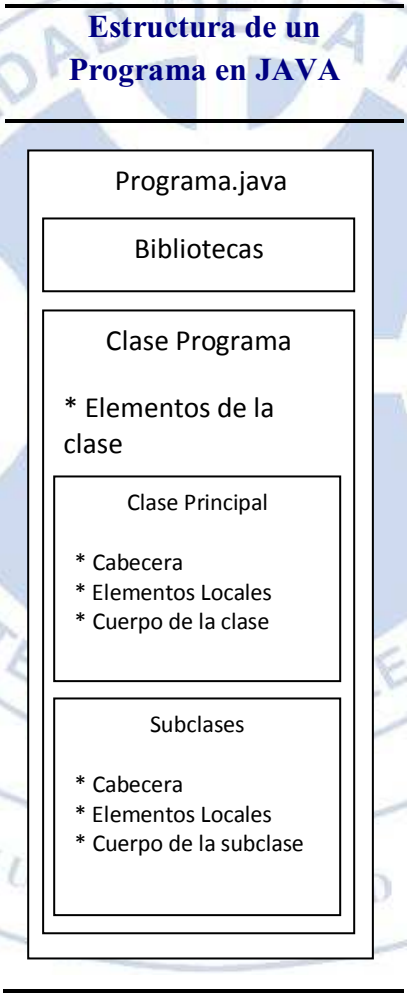
- Utilizando el símbolo /** hasta */, este tipo de comentarios sirve para documentar el programa y generar un reporte HTML utilizando herramientas de documentación como JAVADOC. Por ejemplo

```
/** Comentario estilo javadoc, se incluye
        Automáticamente en documentación HTML */
```

5.3.2. Estructura de un programa

Como se ha indicado, cada lenguaje de programación tiene sus características y formas de construcción propias (Léxico, Sintaxis y Semántica), a pesar de que tienen una historia en común. Esto se refleja en la estructura de desarrollo que plantea el lenguaje, como se aprecia la Fig. 5.1.

Figura 5.1: Estructura de Programas



Estructura de un Programa en JAVA

Dado que JAVA se basa en C++, su estructura es muy similar. Ésta se conforma de la siguiente manera:

a. Cabecera:

Una biblioteca es una colección de rutinas o funciones que el lenguaje de programación utiliza. En otras palabras, es el lugar donde se encuentran definidas las funciones intrínsecas del lenguaje.

Las bibliotecas están clasificadas por lo tipos de funciones que contienen, por ejemplo, existen bibliotecas de funciones matemáticas, de manejo de cadenas de caracteres, de tiempo, etc.

JAVA posee, lo que se denominan Bibliotecas de Clases o APIs (interfaces de programación de aplicaciones), que contienen las clases y sus métodos, que ya están definidas por el lenguaje de programación y que no necesitan ser construidas.

Tabla 3: Paquetes Básicos de APIs de Java

Paquetes de utilidades
java.lang: Fundamental para el lenguaje. Incluye clases como String o StringBuffer,
java.io: Para la entrada y salida a través de flujos de datos, y ficheros del sistema.
java.util: Contiene colecciones de datos y clases, el modelo de eventos, facilidades horarias, generación aleatoria de números, y otras clases de utilidad.
java.math: Clases para realizar aritmética
java.text: Clases e interfaces para manejo de texto, fechas, números y mensajes de una manera independiente a los lenguajes naturales.
java.security: Clases e interfaces para seguridad en Java
Paquetes para el desarrollo gráfico
java.applet: Para crear applets y clases que las applets utilizan para comunicarse con su contexto.
java.awt: Para crear interfaces con el usuario, y para dibujar imágenes y gráficos.
javax.swing: Conjunto de componentes gráficos que funcionan igual en todas las plataformas que Java soporta.
javax.accesibility: Da soporte a clases de accesibilidad para personas discapacitadas.
java.beans: Para el desarrollo de JavaBeans.
Paquetes para el desarrollo en red
java.net: Clases para aplicaciones de red.
java.sql: Paquete que contiene el JDBC, para conexión de programas Java con Bases de datos.

java.rmi: Paquete RMI, para localizar objetos remotos, comunicarse con ellos e incluso enviar objetos como parámetros de un objeto a otro.

org.omg.CORBA: Facilita la posibilidad de utilizar OMG CORBA, para la conexión entre objetos distribuidos, aunque esté codificados en distintos lenguajes.

org.omb.CosNaming : Da servicio al IDL de Java, similar al RMI pero en CORBA

Cabe hacer notar que cada paquete puede tener subpaquetes.

La declaración de bibliotecas se realiza según la siguiente expresión:

import nombre_biblioteca;

Ejemplo 5.2:

Ejemplo de declaración de una biblioteca:

```
import javax.swing.JOptionPane;

import javax.swing.*;

import java.awt.event.*;
```

b. Public Class

Un programa en JAVA puede contener un o más clases. Algunas vienen predefinidas y otras deben ser construidas por el programador. Una de estas clases debe ser designada como Clase Principal. La clase principal corresponde al programa construido por el usuario, incluye todas las clases que contendrá el programa. En esta parte se definen los atributos y métodos que contendrá el programa.

Una clase se construye bajo la siguiente sintaxis:

```
class nombreClase{
        <Declaración de miembros de la clase>
}
```

Nota

No olvidar que la clase principal, debe llevar el mismo nombre del archivo donde está contenida.

c. Atributo:

Atributo de una clase son aquellos datos que definen las características de una clase. Por ejemplo, si se está trabajando con una clase ventana, los atributos pueden ser: ancho, alto, color de fondo, etc.

d. Método:

Método de una clase son aquellas operaciones que se pueden realizar sobre la clase. Tomando como ejemplo la clase ventana, algunos métodos pueden ser: minimizar, maximizar, cerrar, modificar tamaño, etc.

Ejemplo 5.3:

Ejemplo de clase en JAVA

```
public class Persona{

// Atributos

private int idPersona;
private String nombre;
private int fechaNac;
private String genero;

// Métodos

setEdad (edad);

}
```

e. Método Principal o main:

Un programa en JAVA, también debe definir un método main o principal. El cual está contenido dentro de la clase principal, e indica el comienzo del programa para su ejecución.

Sintaxis:

```
public static void main(String [] args){
        <cuerpo del método>
}
```

f. Subclases

Las subclases, corresponden a aquellos métodos definidos por el usuario, que son necesarios para el desarrollo del programa y que están contenidos dentro del principal. Son llamadas por la clase principal cada vez que esta las necesita.

Sintaxis básica de una subclase:

[public] class nombre_clase

La definición de una clase como public es opcional, y significa que puede ser usada por cualquier clase en cualquier paquete. Si no lo es, solamente puede ser utilizada por clases del mismo paquete.

Estructuras de Entrada/Salida JAVA

Todos los lenguajes de programación definen ciertas estructuras que le permiten realizar el ingreso de datos al programa y el envío de información hacia el usuario u otros programas. Como recordarás, cuando construíamos algoritmos denominábamos a estas estructuras como Leer y Escribir.

Lenguaje JAVA:

- **Entrada:** la entrada estándar es la sentencia System.in (contenida en la biblioteca java.io). Esta sentencia permite leer un carácter.

  Otra manera de **leer un carácter** es utilizando una clase Reader, de la siguiente forma:

  InputStreamReader isr = new InputStreamReader(System.in);

  De esta manera, se está declarando una variable, llamada isr, que almacenará el caracter leído a través del System.in. La clase InputStreamReader, permite hacer la transformación del System.in a la variable isr.

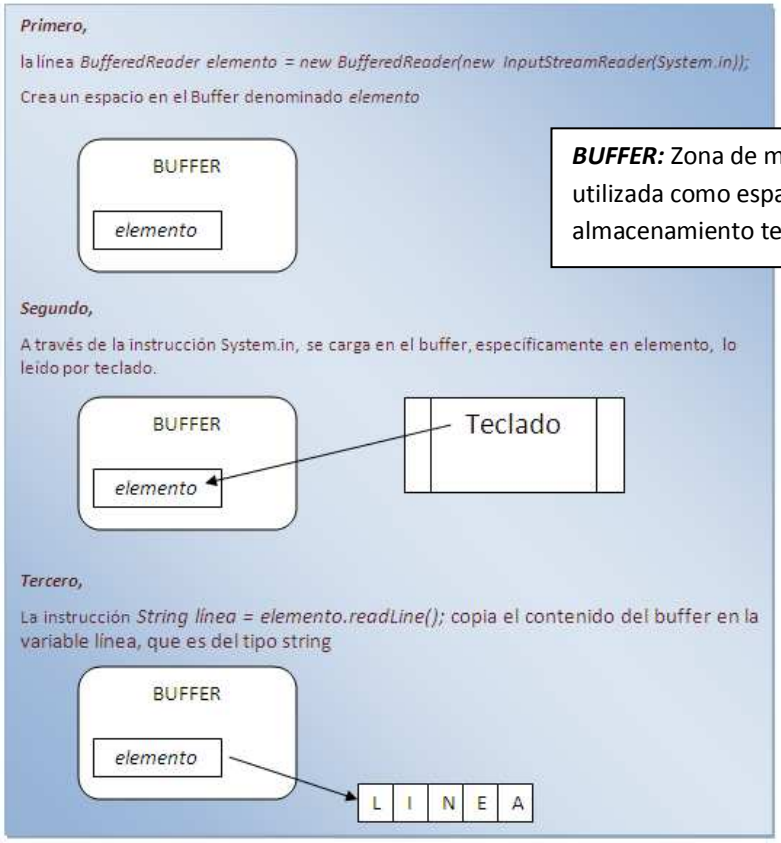
  Para leer una cadena de caracteres, se escribe de la siguiente forma:

  BufferedReader elemento = new BufferedReader(new InputStreamReader(System.in));

  String línea = elemento.readLine();

  La cadena de caracteres quedará almacenada en línea, como se explica en la figura 5.2.

Figura 5.2: cadena de caracteres



Para **leer un valor numérico**, se escribe de la siguiente forma:

```
BufferedReader elemento = new BufferedReader(new InputStreamReader(System.in));
valor = Integer.parseInt(elemento.readLine());
```

El número quedará almacenado en valor.

parseInt: Transforma un número leído como carácter en un valor numérico

Otra forma de ingresar datos es mediante la clase Scanner (contenida en el paquete **java.util.Scanner**). Este paquete permite ingresar datos de distintos tipos mediante los métodos:

- nextBoolean(), permite ingresar datos booleanos
- nextInt(), permite ingresar datos de tipo entero
- nextLong(), permite ingresar enteros más grandes
- nextFloat(), permite ingresar números reales
- nextDouble(), permite ingresar números reales más grandes
- nextString() or next(), permite ingresar caracteres
- nextLine(), permite ingresar una línea de texto

La sintaxis de esta clase es la siguiente:

Scanner intro = new Scanner(System.in);

En este caso la variable ***intro*** permitirá el ingreso de distintos tipos de datos mediante los métodos anteriores.

Por ejemplo, si se quiere leer un número entero → **num = intro.nextInt();**

Otra forma de ingresar datos es a través de las "cajas de mensajes", utilizando los objetos del tipo ***JoptionPane***, contenidos en el paquete ***javax.swing.JOptionPane***. Para el ingreso se utiliza la forma:

showInputDialog

Ejemplo 5.4:

Por ejemplo, si se quiere leer un entero:

```
n1 = JOptionPane.showInputDialog("Introduzca el Primer Numero");
num1 = Integer.parseInt(n1);
```

En este ejemplo, la primera sentencia lee lo introducido por teclado y lo guarda en n1, sin embargo la lectura se realizó como carácter. Para realizar la transformación a un número entero, se utiliza la sentencia **Integer.parseInt**. Quedando, finalmente, en la variable num1, el valor leído como un valor entero.

- **Salida**: la salida estándar es la sentencia System.out (contenida en la biblioteca *java.io*), su sintaxis es:

System.out.print("[Mensaje]" + [variable]);

La sentencia System.out permite enviar como salida un mensaje o el valor de una variable. También permite la salida de ambas en forma combinada.

Ejemplo 5.4:

System.out ("Este es un mensaje para pantalla!"); → Se observará en pantalla

Este es un mensaje para pantalla!

System.out ("El valor es: " + valor); → El valor es: 5

Si la variable valor es 5, entonces se observará en pantalla

Para la salida a través de "cajas de mensajes", se utiliza la forma:

showMessageDialog

Ejemplo 5.5:

JOptionPane.showMessageDialog(null, "El resultado es: " + valor);

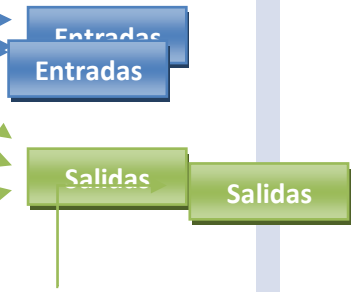
Ejemplo 5.6:

Ejemplo de Entrada/Salida

Lenguaje Java

Usando JOptionPane

```
import javax.swing.JOptionPane;

public class sumaJOptionPane
{

 public static void main (String args[])

 {

  String n1;
  String n2;
  int num1, num2, suma;

  n1 = JOptionPane.showInputDialog("Introduzca primer número: ");
  n2 = JOptionPane.showInputDialog("Introduzca segundo número: ");

  num1 = Integer.parseInt(n1);
  num2 = Integer.parseInt(n2);
  suma = num1 + num2;

  JOptionPane.showMessageDialog(null, "La suma es: "+ suma, "Sumar 2 números enteros", JOptionPane.PLAIN_MESSAGE);
 // n1 = intro.nextInt();
}
}
```



Manejo de excepciones en JAVA:

Una excepción es la indicación de un problema que ocurre durante la ejecución de un programa, pero que no es un problema que se produzca de forma habitual (Deitel, 2004). Por ejemplo, se puede generar una excepción para manejar la división por cero. Este problema, sólo se produce en aquellos casos cuando el numerador es cero.

El manejo de excepciones, permite salvaguardar el funcionamiento del programa a pesar de que se produzca la situación, de tal manera de evitar que se detenga la ejecución del programa si se presenta el problema.

Para el manejo de excepciones se utiliza la sentencia try-catch, de acuerdo a la siguiente sintaxis:

```
try {    // Código que pueda generar Errores ("Excepciones")
   } catch(Tipo1 id1) {       // Manejar " Excepciones " para la Clase Tipo1
   } catch(Tipo2 id2) {       // Manejar " Excepciones " para la Clase Tipo2
   }
```

Ejemplo 5.7:

```
public class ExcepcionApp {

    public static void main(String[] args) {

        String str1="12";
        String str2="0";
        int numerador, denominador, cociente;

        try{
            numerador=Integer.parseInt(str1);
            denominador=Integer.parseInt(str2);
            cociente=numerador/denominador;
            System.out.println(String.valueOf(cociente));

        }catch(NumberFormatException ex){
            System.out.println("Se han introducido caracteres no numéricos");

        }catch(ArithmeticException ex){
            System.out.println("División entre cero");

        }

    }
}
```

Instrucciones de la operación División, dentro de la sentencia try

Primera Excepción: Excepción de formato. En el caso de que se ingrese algún dato que no es numérico, a través de la sentencia `NumberFormatException`

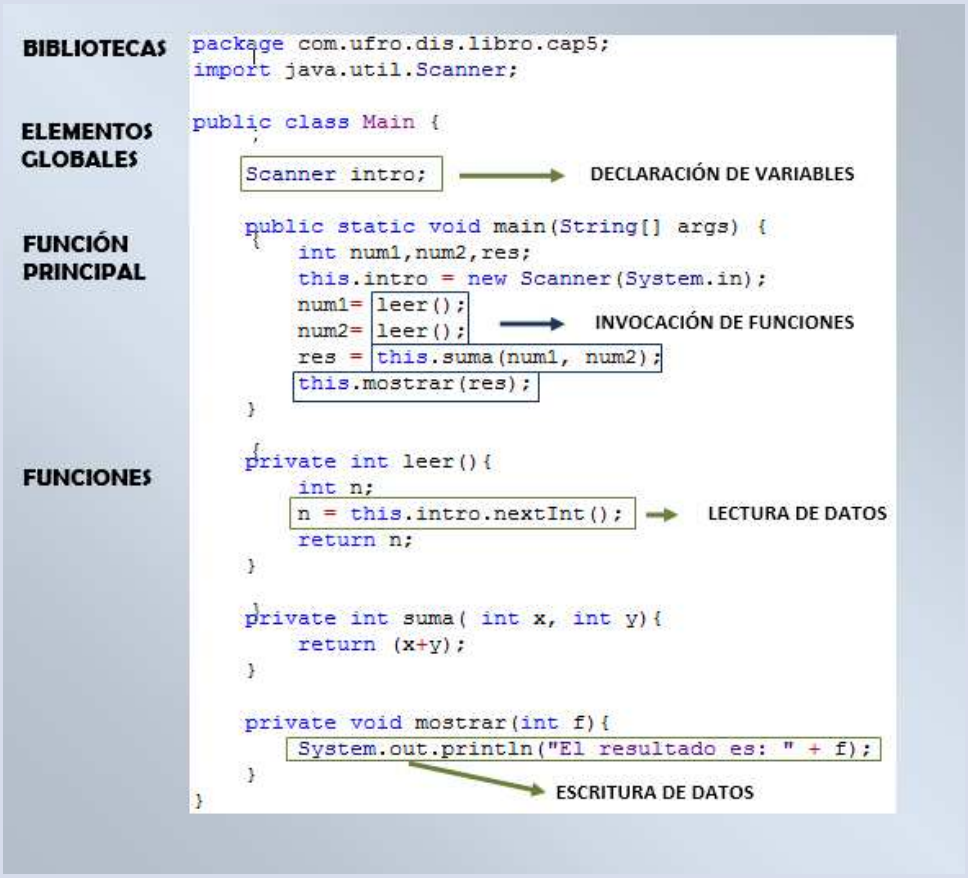
En este caso se despliega el mensaje "Se han introducido caracteres no numéricos"

Segunda Excepción: Excepción Aritmética. En el caso de que se produzca la división por cero, a través de la sentencia `NumberFormatException`.

En este caso se despliega el mensaje "División entre cero"

Ejemplo 5.8:

Ejemplo de Estructura de Programa en Java




```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

public class Main {

    Scanner intro;

    public static void main(String[] args) {
        int num1,num2,res;
        this.intro = new Scanner(System.in);
        num1= leer();
        num2= leer();
        res = this.suma(num1, num2);
        this.mostrar(res);
    }

    private int leer(){
        int n;
        n = this.intro.nextInt();
        return n;
    }

    private int suma( int x, int y){
        return (x+y);
    }

    private void mostrar(int f){
        System.out.println("El resultado es: " + f);
    }
}
```

5.4 Estructuras de control

Las estructuras de control, son aquellos bloques de código que sirven para controlar el flujo de ejecución dentro de un programa. Como ya se ha visto en capítulos anteriores, veremos dos tipos de estructuras de control, las estructuras de selección y las estructuras de Iteración. Un programa está conformado por la unión o interacción de estas estructuras.

5.4.1 Estructuras de Selección

Este tipo de estructuras permiten la selección de un bloque de instrucciones de acuerdo a una determinada condición. Existen tres tipos: Simple, Doble, Múltiple.

Lenguaje JAVA
Simple (estructura "si")
if (a > b) && (b > c)
Doble (estructura "si-sino")

if (a > b) && (b > c)
…..
else
…..

if (a > b) && (b > c) ….. else if (a < c) …..
Múltiple (estructura "switch-case")

switch (exp){
case constante1:
sentencia; break;
case constante2:
sentencia; break;
……..
default: sentencia;
}

Ejemplos de Estructuras Selectivas (En lenguaje Java)

Ejemplo 5.9: Estructura Selectiva simple

Programa que calcula el valor absoluto de un número ingresado por el usuario

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        float num;
        Scanner intro = new Scanner(System.in);

        System.out.println("Ingrese numero: ");
        num = intro.nextFloat();
        if(num < 0 ){
            num = (-1) * num;
        }
        System.out.println("El Valor absoluto es: " + num);
    }
}
```

Ejemplo 5.10: Estructura Selectiva doble

Programa indica si el valor ingresado esta o no dentro del intervalo [0,100]

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        float num;
        Scanner intro = new Scanner(System.in);

        System.out.print("Ingrese numero: ");
        num = intro.nextFloat();
        if((num >= 0) && (num <= 1000)){
            System.out.println("El valor esta dentro del intervalo");
        }else{
            System.out.println("El valor No esta dentro del intervalo");
        }
    }
}
```

Ejemplo 5.11: Estructura Selectiva múltiple

Programa entrega la relación entre dos números

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        int valor1, valor2;
        Scanner intro = new Scanner(System.in);

        System.out.print("Ingrese Primer valor: ");
        valor1 = intro.nextInt();
        System.out.print("Ingrese Segundo valor: ");
        valor2 = intro.nextInt();

        if(valor1 > valor2){
            System.out.println(valor1 + " es el mayor");
        }else if(valor 1 < valor2){
            System.out.println(valor2 + " es el mayor");
        }else{
            System.out.println("los valores son iguales");
        }
    }
}
```

5.4.2. Estructuras de Iteración

Este tipo de estructuras permite la repetición de un bloque de código, tantas veces como sea necesario. Existen tres tipos: Mientras, Hacer-Mientras, Para.

Lenguaje JAVA
Mientras (while)

while (var<num){
…..
}

Hacer-Mientras (do-while)
do { ….. }while(var<num);
Para (for)

for (i=0; i<num; i++){
…..
}

Ejemplos de Estructuras Iterativas (En lenguaje Java)

Ejemplo 5.12: Estructura iterativa Mientras

Programa entrega el mayor valor de una secuencia de números, usando un ciclo while

```
package com.ufro.dis.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        float num, max=0;
        int cont=0, n;
        Scanner intro = new Scanner(System.in);

        System.out.print("Cantidad de numeros a ingresar: ");
        n = intro.nextInt();

        while (cont < n){
            System.out.print("Ingrese numero: ");
            num = intro.nextInt();
            if (num > max){
                max = num;
                cont++;
            }
         }
         System.out.println("El numero mayor es: " + max);
    }
}
```

Ejemplo 5.13: Estructura iterativa Hacer-Mientras

Programa entrega el mayor valor de una secuencia de números, usando un ciclo do-while

```
package com.ufro.dis.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        float num, max=0;
        int cont=0, n;
        Scanner intro = new Scanner(System.in);

        System.out.print("Cantidad de numeros a ingresar: ");
        n = intro.nextInt();
        do{
            System.out.print("Ingrese numero: ");
            num = intro.nextInt();
            if (num > max){
                max = num;
                cont++;
            }
         }while (cont < n);
         System.out.println("El numero mayor es: " + max);
    }
}
```

Ejemplo 5.14: Estructura iterativa Para

Programa entrega el mayor valor de una secuencia de números, usando un ciclo for

```
package com.ufro.dis.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        float num, max=0;
        int cont, n;
        Scanner intro = new Scanner(System.in);

        System.out.print("Cantidad de numeros a ingresar: ");
        n = intro.nextInt();
        for (cont=0; cont < n; cont++){
            System.out.print("Ingrese numero: ");
            num = intro.nextInt();
            if (num > max){
                max = num;
            }
        }

         System.out.println("El numero mayor es: " + max);
    }
}
```

5.5. Ejercicios Resueltos

Ejercicio 5.1:

Construya un programa que pueda determinar cuál de dos números enteros ingresados por teclado, es el mayor.

Solución

Es importante darse cuenta que este problema posibilita tres respuestas: (i) El Primer número es el mayor, (ii) el segundo número es el mayor o (iii) los números son iguales. Para determinar la respuesta correcta, se debe construir una estructura del tipo si/sino que pueda discriminar, a través de una condición, la relación que existe en entre los números ingresados.

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int num1,num2; // Declaración de variables
        Scanner intro;

        //Inicializa las variables
        num1=0;
        num2=0;
        intro = new Scanner(System.in);

        //Lee ambos valores
        System.out.print("Ingrese el Primer Numero: ");
        num1 = intro.nextInt();
        System.out.print("Ingrese el Segundo Numero: ");
        num2 = intro.nextInt();

        //Compara ambos valores y entrega respuesta
        if (num1>num2)
            System.out.println("El Primero es Mayor");
        else{
           if (num1<num2)
            System.out.println("El Segundo es Mayor");
           else
            System.out.println("Los Numeros son Iguales");
        }
        //Genera una pausa para ver el resultado
        intro.next();
    }
}
```

Ejercicio 5.2:

Construya un programa que muestre los divisores de un número ingresado por el usuario.

Solución

Para este ejercicio se debe utilizar un ciclo que incremente un contador desde uno hasta que se igual al número ingresado. Por cada iteración, se determina el resto entre el número ingresado y el valor del contador en dicha iteración. Si el resto es cero, quiere decir que el valor del contador es divisor del número, por lo tanto se muestra en pantalla.

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;


class Main {

    public static void main(String[] args) {
        int num, cont;
        Scanner intro = new Scanner(System.in);

        num = intro.nextInt();
        cont = 1;
        while(cont <= num){
            if(num%cont == 0){
                System.out.println("Divisor: " + cont);
            }
            cont++;
        }
    }
}
```

Ejercicio 5.3:

Construya un programa que pueda determinar el año en que nació la persona más joven, de una cantidad de personas indicada por el usuario, comparando sólo sus años de nacimiento.

Solución

Dado que en este caso se refiere a una cantidad n de personas, se debe utilizar un ciclo que realice tantas iteraciones como años de nacimientos ingresará el usuario. Cada año ingresado se compara con el menor que se tiene hasta el momento, si el nuevo año es menor que el anterior, entonces se cambia.

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        int agnoNac, menor, personas, contador;
        Scanner intro = new Scanner(System.in);

        agnoNac=0;
        menor=0;
        contador=1;

        System.out.print("¿Cuantos personas desea ingresar?: ");
        personas = intro.nextInt();

        do{         //ciclo que lee el nuevo número y compara con el anterior
            System.out.print("Ingrese el año de nacimiento de la persona " + contador +": ");
            agnoNac = intro.nextInt();
            if (agnoNac > menor) {
                menor = agnoNac;
            }
            contador++;

        }while(contador<=personas);
        System.out.print("La persona mas joven nació en el año: " + menor);
    }
}
```

Ejercicio 5.4:

Considere un centro sismológico que lleva el registro de los movimientos telúricos que se producen en la región. Construya un algoritmo que sea capaz de llevar el registro de 10 movimientos. Debe recibir la magnitud del movimiento y entregar: (i) Magnitud Máxima, (ii) Magnitud Mínima y (iii) Magnitud Promedio.

Además debe clasificar lo movimientos de acuerdo a la siguiente tabla:

Rango movimiento	Clasificación
Menor 4 grados	Sismo Leve
Entre 4 y 7	Sismo Medio

Mayor a 7	Terremoto

Además, si el movimiento es de 7 grados o superior, deberá emitir una alerta de Tsunami.

Solución

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        int cont;
        float mag, prom, max, min;
        Scanner intro = new Scanner(System.in);
        cont = 1;
        prom = 0;
        mag =0;
        max = 0;
        min = 100;

        while (cont <= 10){
            System.out.print("Ingrese Magnitud evento: ");
            mag = intro.nextFloat();
            prom = prom + mag;
            if (mag < 4){
                System.out.println("El Sismo es Leve");
            }else if(mag < 7){
                System.out.println("El Sismo es Medio");
            }else{
                System.out.println("Es Un TERREMOTO!!!");
                System.out.println("ALERTA DE TSUNAMI!!!!!!!!!!!!");
            }
            if (mag > max){
                max = mag;
            }else if (mag < min){
                min = mag;
            }
            cont = cont + 1;
        }
        System.out.println("La Magnitud Maxima es: " + max);
        System.out.println("La Magnitud Minima es: " + min);
        System.out.println("El Promedio es: " + prom/10);
    }
}
```

Ejercicio 5.5:

Construya un programa que registre el consumo eléctrico de un edificio de 10 departamentos. Debe indicar: (i) Consumo Máximo, (ii) Consumo Mínimo y (iii) Consumo Promedio. Además, deberá indicar por cada departamento, el valor a pagar de acuerdo a su consumo. Indique el consumo total del edificio.

Tabla valores por intervalo de consumo

Lectura medidor	Precio
Entre 0 y 100 KW	$ 100 / KW
Entre 100 KW y 200 KW	$ 200 / KW
Superior a 200 KW	$ 500 / KW

Solución

Se deberá utilizar un ciclo que permita determinar el consumo menor y mayor, a través de comparaciones, y el consumo promedio, a través de la suma de todos los consumos divido por la cantidad de medidores, esto para los 10 departamentos.

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        int depto, consumo, consumoTotal, consumoMax, consumoMin;
        float factura, consumoPromedio;

        Scanner intro = new Scanner(System.in);
        depto = 1;
        consumoTotal = 0;
        consumoMax = 0;
        consumoMin = 1000;
        while(depto <= 10){
            System.out.print("Ingrese lectura del medidor para departamento " + depto + " : ");
            consumo = intro.nextInt();
            consumoTotal = consumoTotal + consumo;
            if(consumo > consumoMax){
                consumoMax = consumo;
            }
            if(consumo < consumoMin){
                consumoMin = consumo;
            }
            if(consumo < 101){
                factura = consumo*100;
                System.out.println("El valor para el departamento " + depto + " es: " + factura);
            }else if(consumo < 201){
                factura = consumo*200;
                System.out.println("El valor para el departamento " + depto + " es: " + factura);
            }else{
                factura = consumo*500;
                System.out.println("El valor para el departamento " + depto + " es: " + factura);
            }
```

```
            depto = depto + 1;
        }
        consumoPromedio = consumoTotal/depto;
        System.out.println("El Consumo Mínimo fue: " + consumoMin);
        System.out.println("El Consumo Máximo fue: " + consumoMax);
        System.out.println("El Consumo Promedio fue: " + consumoPromedio);
    }
}
```

Ejercicio 5.6:

Considere un equipo de futbol que compite en una liga regional, que consiste en 10 partidos. Construya un programa que pueda manejar los datos históricos del equipo, a partir de los goles realizados en un partido. Deberá mostrar: (i) Goles a Favor, (ii) Goles en contra, (iii) Puntos, (iv) Cantidad de Partidos Ganados/Empatados/Perdidos y (v) Diferencia de goles.

Además debe clasificar al equipo de acuerdo a la siguiente tabla:

PUNTOS	CALSIFICACIÓN
Menos de 10 puntos	Liguilla de Promoción
Entre 10 y 20 puntos	No liguilla
Mayor a 20 puntos	Liguilla de Campeonato

Solución

```
package com.ufro.dis.libro.cap5;
import java.util.Scanner;

class Main {

    public static void main(String[] args) {
        int partidos, ptos, gf, gc, gft, gct, pe, pp, pg;
        Scanner intro = new Scanner(System.in);
        partidos = 1;
        ptos = gf = gc = gft = gct = pe = pp = pg = 0;
        while(partidos <= 10){
            System.out.println("Partido " + partidos);
            System.out.print("Ingrese Goles de su Equipo: ");
            gf = intro.nextInt();
            System.out.print("Ingrese Goles del otro Equipo: ");
            gc = intro.nextInt();
            gft = gft + gf;
            gct = gct + gc;
            if(gf < gc){
                System.out.println("Partido Perdido");
                pp = pp + 1;
            }else if(gf > gc){
                System.out.println("Partido Ganado");
```

```
                ptos = ptos + 3;
                pg = pg + 1;
            }else{
                System.out.println("Partido Empatado");
                ptos = ptos + 1;
                pe = pe + 1;
            }
            partidos = partidos + 1;
        }
        System.out.println("");
        System.out.println("Estadisticas Finales....");
        System.out.println("");
        System.out.println("Partidos Ganados: " + pg);
        System.out.println("Partidos Empatados: " + pe);
        System.out.println("Partidos Perdidos: " + pp);
        System.out.println("Goles a Favor: " + gft);
        System.out.println("Goles en Contra: " + gct);
        System.out.println("Diferencia de goles: " + (gft - gct));
        System.out.println("");
        System.out.println("Puntos Totales");
        if(ptos < 10){
            System.out.println("Liguilla de Promocion");
        }else if(ptos <20){
            System.out.println("No Liguilla");
        }else{
            System.out.println("Liguilla de Campeonato");
        }
    }
}
```

5.6. Ejercicios Propuestos

Ejercicio 5.7:

Traduzca a lenguaje JAVA, los ejercicios de los capítulos 3 y 4

Ejercicio 5.8:

Construya un programa que a partir de las coordenadas de dos puntos, pueda calcular la ecuación de la recta que los une y la distancia que los separa.

Ejercicio 5.9:

Construya un programa que le permita jugar al Loto. Que permita ingresar los números comprados por el usuario y comparándolos con los números sorteados, que indique la cantidad de aciertos y el premio obtenido.

Ejercicio 5.10:

Hacer un programa que nos permita introducir un número por teclado y sobre él se realicen las siguientes operaciones: comprobar si es primo, hallar su factorial o imprimir su tabla de multiplicar.

Ejercicio 5.11:

Dada una lista de X notas de un alumno, mostrar por pantalla: Promedio, Cantidad de notas aprobaras, Cantidad de notas reprobadas, La nota más alta y La nota más baja. Indicar si el alumno aprueba o no la asignatura, considerando que los requisitos son: tener un promedio mayor o igual a 4.0 y cantidad de notas reprobadas es inferior al 50%.

Ejercicio 5.12:

Cree un programa que permita construir el valor de un sándwich creado a la medida del cliente. El valor base del sándwich es de $300 lo que incluye solo el pan. Si el sándwich es churrasco al valor base se suman $400, si es lomo de cerdo $350, si es ave $250 y si es atún es $350. Si desea agregar uno de los siguientes ingredientes se suman $150 adicionales por cada uno: lechuga, tomate, champignon, palta, palmitos, queso, tocino, jamón, huevo frito, cebolla. Si el cliente elige más de 3 ingredientes, el precio por ingrediente es de $120, y si elige más de 5 el precio es de 100 por ingrediente. Un cliente no puede pedir solo el pan, o sea no se puede vender un sándwich de $300.

Ejercicio 5.13:

Construya un programa que registre la llegada a la meta de los corredores de la maratón de Santiago. Considere que los corredores pertenecen a 3 equipos. Debe registrar las 10 primeras posiciones y entregar un resumen que contenga la siguiente información: (i) Cantidad de corredores por cada equipo, que llegó dentro de las primeras posiciones y (ii) Cantidad total en premios por equipo

Tabla resumen de premios

Posiciones	Premio
Primer lugar	$ 100.000
Segundo lugar	$ 80.000
Tercer lugar	$ 60.000
Entre el 4º y el 6º lugar	$ 40.000
Entre el 7º y el 10º lugar	$ 20.000
Superior al 10º lugar	No recibe Premio

5.7 Comentarios Finales

En este capítulo hemos tratado las reglas para implementar los algoritmos diseñados en los capítulos anteriores, utilizando el determinado Lenguaje de programación Java. También estudiamos la estructura de dicho lenguaje y sus reglas sintácticas y semánticas básicas.

5.8 Referencias

Louden, Kenneth "Lenguajes de Programación: Principios y Prácticas" Ed. Thomson, 2º edición, 2004.H., Deitel, P., Deitel, "Como programar en JAVA", Ed. Pearson, Quinta Edición, 2005