1. ESTRUCTURA DE UN PROGRAMA EN JAVA

Un programa Java está compuesto por una o más clases. Es obligatorio que uno de los métodos de la clase principal sea el método ***main().*** Un método es un conjunto de instrucciones que realizan una o más acciones [1]. Java también permite que desde un programa se pueda hacer uso de otros programas escritos previamente, para hacer esto posible hacemos uso de la instrucción ***import*** seguida del nombre del programa (archivo) que queremos incluir en nuestro programa.

1.1 NOMBRE DEL PROGRAMA

Es el nombre que identifica al programa principal. Este nombre debe ser el mismo de la clase principal. Ejemplo: ProgramaAreas.java. En la clase principal debe estar el método main().

1.2 INCLUSIÓN DE OTROS ARCHIVOS

A través de la instrucción ***import*** se pueden incluir archivos que contienen clases y métodos previamente definidos. La línea siguiente permite que un programa java pueda utilizar las clases de entrada y salida.

```
Import java.io.*;
```

1.3 MÉTODO main()

Método encargado de inicializar la ejecución de un programa en java [2]. El argumento de ***main()*** es un arreglo de cadenas que recibe datos de la línea donde se ejecute el programa. Sin importar que no haya argumentos en la línea de ejecución es obligatorio especificar ***String[].*** Para este método, java reglamenta la siguiente sintaxis:

```
public static void main(String[] args)
        {
                declaraciones locales;
                instruccion_1;
                instruccion_2;
                instruccion_n;
        }
```

1.4 OTROS MÉTODOS

Son métodos adicionales al método main() que son definidos por el usuario dentro de una clase para realizar tareas específicas. Los métodos en java son subprogramas que pueden devolver un valor único, un conjunto de valores o no devolver valores y realizar una acción particular.

```
static double calcularArea()
{
                declaraciones locales;
                instruccion_1;
                instruccion_2;
                instruccion_n;
        }
```

1.5 COMENTARIOS

Son líneas que introducidas por los programadores pero que no son tenidas en cuenta al momento de compilar y ejecutar un programa. En java se pueden utilizar los símbolos siguientes para indicar un comentario:

```
//Comentario para una línea
/*Comentario
   Para
   Varias
   Líneas */
```

1.6 PUNTO Y COMA (;)

En java cada instrucción debe finalizar con un punto y coma (;) Es posible tener varias instrucciones en una sola línea e instrucciones de varias líneas.

```
Instruccion_n;
Instrucción_1; instrucción_2; instrucción_n;

System.out.println("Este ejemplo muestra como se puede
crear una instrucción java en varias líneas. Aquí el
triple de "+x+ "es "+ 3*x);
```

1.7 EJEMPLO DE UN PROGRAMA EN JAVA

```
import java.io.*;
public class Ejercicio1
{
      public static void main(String[]args)
      {
            Cuadro Cuadro1 = new Cuadro();
            System.out.println("Area ="+Cuadro1.
            calcularArea());
      }
}
class Cuadro
      {
            double alto = 5;
            double largo = 10;
            double calcularArea()
                  {
                        return alto * largo;
                  }
      }
```

1.8 PAQUETES (package)

Los paquetes son una agrupación de clases. Los paquetes están localizados en folders del disco que tienen el mismo nombre que el paquete. Para hacer uso de las clases de un paquete se utiliza la instrucción import, la cual sigue la siguiente sintaxis:

```
Import  nombrePaquete.nombreClase;
```

Ejemplo:

```
import java.io.*;          //Incorpora todas las clases del paquete java.io
Import java.util.Date;     //Incorpora la clase Date del paquete java.util
```

1.8.1 PAQUETE java.lang

Es un paquete especial que contiene todos los elementos básicos para la elaboración de un programa. Debido a su importancia el compilador

siempre lo incluye lo cual evita que tenga ser incorporado por el programador al programa.

1.9 DECLARACIÓN DE CLASES

Un programa en Java está organizado como una colección de clases. El programa debe tener al menos una clase, puede ser considerada como la clase principal, con el método main() y si es necesario otros métodos y variables[1]. Para declarar una clase en Java se hace con la siguiente sintaxis:

```
tipo_acceso class Nombre_Clase
```

tipo_acceso	Indica el tipo de acceso de la clase (Es opcional)
class	Palabra reservada de java para hacer la declaración de una clase.
Nombre_Clase	Nombre con el que se identifica una clase dentro de un programa.

```
//Ejemplo:  Programa con una clase única
public class Cuadrado {
      public static void main(String[]args)
            {
                  int numero = 3;
                  System.out.println("El cuadrado de
                  "+numero+" es "+numero*numero);
            }

}

//Ejemplo:  Declaración de clases adicionales y
Declaraciones dentro de una clase
public class Ejemplo
{
      public static void main(String[]args)
            {
                  //Creación de un objeto de la clase Suma
                  Suma Suma1=new Suma();
```

[1] JOYANES AGUILAR, Luis. Programación en Java 2. Pg 105.

```
                System.out.println("El total de la suma es
                "+Suma1.sumarValores());
            }
}
class Suma
    {
        int valor1, valor2; //Variables que se pueden
        utilizar en //cualquier método de la clase
        public int sumarValores()
            {
                int valor3, total; //Variables
                locales al método
                valor1 = 1000;      //y solo pueden
                utilizarse
                valor2=valor3=200;  // dentro de él
                total=valor1+valor2+valor3;
                return total;
            }
    }
```

1.10 PROGRAMACIÓN DEL MÉTODO main()

El método ***main()*** es el encargado de inicializar un programa. Todos los programas deben contener uno y sólo un método ***main()***, si se declaran dos o más métodos ***main()*** se generará un error así estos hayan sido creados en diferentes clases.

Sintaxis del método main()

```
public static void main(String[ ] args)
        {
                Instruccion_1;
                Instruccion_2;
                .....
                Instruccion_n;
        }
```

Encabezado: public static void main(String[ ] args)

El método ***main()*** tiene como argumento una lista o arreglo de cadenas que le dan la posibilidad de introducir cadenas de caracteres desde la línea de ejecución.

1.11 DEFINICIÓN DE OTROS MÉTODOS

Un programa java, además de tener un método main() desde el que se inicializa el programa, puede tener otros métodos definidos por el usuario según las necesidades que tenga para dar solución a un determinado problema. Estos métodos pueden estar definidos dentro de la clase principal del programa, si es que el programa es muy pequeño, pero para programas extensos, es recomendable que estos métodos se definan dentro de otras clases. Cada método pertenece a la clase en la cual es definido y se invoca desde esta misma clase.

1.12 ESTRUCTURA DE UN MÉTODO

```
tipo_de_retorno nombreMetodo(parámetros)
        {
                Variables internas del método;
                Instrucciones del método;
                return valor;
        }
```

tipo_de_retorno	*Corresponde al tipo de dato devuelto por el método.*
nombreMetodo	*odos los métodos deben llevar un nombre que los identifique. Se sugiere que este nombre identifique la tarea realizada por el método.*
parámetros	*Conjunto de parámetros que recibe el método.*

1.13 MÉTODOS QUE NO RETORNAN VALORES

No todos los métodos retornan valores para que sean tomados por otros métodos. Para poder hacer esto, es necesario anteponer al nombre del método la palabra reservada *void,* la cual indicará al compilador que el método que se está definiendo no retornará ningún tipo de valor (int, double, char, long, float, etc.).

El siguiente ejemplo muestra un programa donde se crea una nueva clase llamada Valores, en la cual se definen las variables a, b, y c de tipo int junto con el método *visualizarValores()* de tipo *void.* Este método permite visualizar los contenidos de las variables en un punto dado del programa. En la clase principal *OperadorAsigClase*, se define el método

main(), en el cual se crea una instancia de la clase Valores con el nombre *MiValor*. También se le asignan nuevos valores a las variables del nuevo objeto y podemos ver la forma como se hace el llamado al método *visualizarValores()*

```
public class OperadorAsigClase
{
      public static void main(String[]args)
      {
            Valores MiValor = new Valores();
            MiValor.visualizarValores(); //Llamado al método //
                                          visualizarValores()
            MiValor.a = MiValor.b;      //Asignación de valores
                                        entre //variables
            MiValor.c = MiValor.a;
            MiValor.visualizarValores();
            MiValor.a = MiValor.b = MiValor.c = 5000;
                                        //Asociatividad por
                                        //derecha
            MiValor.visualizarValores();
      }
}

class Valores
{
      int a = 1000;
      int b = 2000;
      int c = 3000;
      void visualizarValores()
      {
            System.out.println("El valor de a es "+a);
            System.out.println("El valor de b es "+b);
            System.out.println("El valor de c es "+c);
      }
}
```

Una vez ejecutado este programa la salida será la siguiente:

```
El valor de a es 1000
El valor de b es 2000
El valor de c es 3000
El valor de a es 2000
El valor de b es 2000
```

```
El valor de c es 2000
El valor de a es 5000
El valor de b es 5000
El valor de c es 5000
```

1.14 CREACIÓN Y EJECUCIÓN DE UN PROGRAMA EN JAVA

Java contempla los siguientes pasos para el proceso de creación y ejecución de un programa.

1. ***Escribir el programa(Código fuente):*** Para este paso se hace necesario tener un editor de texto, en el que el programador digitará y grabará las instrucciones que conforman el programa. Una vez escrito el programa se debe guardar con un nombre bajo el cual se identificará este programa. Los programas fuente deben tener el mismo nombre de la clase principal y se guardarán con la extensión .java (programa1.java)

2. ***Compilar el programa(Código fuente):*** El programa fuente escrito por el programador, no es entendido por la máquina, es por esto que se hace necesario traducir este programa a un formato entendible por la máquina. Java a través de su compilador convierte el código fuente en Byte Code, que es un código intermedio entre el código fuente y el lenguaje de máquina. Este Byte Code podrá ejecutarse en cualquier máquina independiente de la plataforma gracias al trabajo realizado por la Máquina Virtual de Java, la cual se encarga de adecuar este código a las diferentes plataformas sobre las que el programa puede ser ejecutado. Un programa compilado tendrá el mismo nombre del programa fuente pero con la extensión .class (programa1.class). Es de resaltar que si el programa hace referencia a clases de otros paquetes, estas son incluidas en el Byte Code durante el proceso de compilación.

3. ***Ejecución del programa(archivo.***class) El archivo .class es ejecutado por la Máquina Virtual de Java. Este proceso consta de tres etapas: primero el archivo .class es cargado en la memoria, luego se hace la verificación y finalmente se ejecutan cada una de las instrucciones contenidas el archivo .classs.

1.15 ERRORES EN JAVA

En los lenguajes de programación orientados a objetos el concepto de error se cambia por el de ***excepción,*** de esta manera cuando se diseña una aplicación en Java, uno de sus objetivos elementales es el de agregar el tratamiento de fallos o errores que se pueden producir durante su ejecución. "A estos errores o condiciones anormales que ocurren durante la ejecución de un segmento de código se les denomina ***excepciones"*** [3].

Son tres los tipos de errores que se pueden presentar al momento de compilar un programa en Java.

1. ***Errores de Sintáxis:*** Son aquellos que se presentan por no seguir las reglas propias del lenguaje. Entre estos se encuentran el mal uso de palabras reservadas, omisión de símbolos tales como el punto y coma al final de una instrucción, inserción de símbolos en lugares donde no deben ir, por ejemplo, la inclusión de un punto y coma al final de la definición de un método.

2. ***Errores lógicos:*** Son aquellos que se presentan por el hecho de hacer un análisis incompleto o mal hecho de un problema. Estos errores son difíciles de encontrar ya que el compilador no los detecta al momento de hacer la compilación y ejecución del programa. Entre los más frecuentes están: fórmulas de cálculo incorrectas y requerimientos mal definidos.

3. ***Errores de regresión:*** Son aquellos que se producen cuando se corrigen otros errores. Es importante que cuando se hagan correcciones sobre el programa fuente, se analicen los posibles efectos del cambio o cambios hechos para que el programa no llegue a presentar otros errores inesperados.

1.16 TIPOS DE DATOS EN JAVA

Se denominan tipos de datos al universo de valores que pueden ser asignados a una variable. Java posee ocho tipos de datos, los cuales son resumidos en la siguiente tabla:

Tipo de dato	Tamaño en bits	Rango	
		Valor mínimo permitido	Valor máximo permitido
Byte	8	-128	127
short	16	-32.768	32.767
Int	32	-2.147.483.648	2.147.843.647
Long	64	-263	263 - 1
Float	32	10^{-46}	10^{38}
double	64	10^{-324}	10^{308}
Char	16	'\0000'	'\FFFF'
Boolean	1	false	true

Como todos los lenguajes modernos de programación, Java soporta diferentes tipos de datos los cuales son utilizados para declarar variables y crear arreglos [4]

1.17 VARIABLES

Un programa en lenguaje Java, gestiona una serie de datos los cuales pueden ser fijos o variables, pueden estar también incorporados por defecto al programa o ser obtenidos en el momento que este es ejecutado. Estos datos se identifican a través de nombres simbólicos denominadas en general "variables o constantes". Estos nombres identifican espacios de memoria en los que se sitúan los datos antes de ser utilizados por el procesador en una operación elemental, tal como un cálculo aritmético, una comparación o el traslado de un dato de un lugar a otro [5]

Se denomina variable al nombre de una posición de memoria cuyo valor almacenado puede cambiar a lo largo del programa. Los nombres de las variables en java son dados por el programador, teniendo en cuenta que existen algunos símbolos del lenguaje que no pueden ser utilizados para tal fin, entre los que se encuentran los operadores matemáticos, los separadores (. , : ;), así como un conjunto de palabras reservadas del lenguaje que tienen significado propio para el compilador..

abstract	default	goto	native	static	void
boolean	do	if	new	super	volatile
break	double	implements	null	switch	while
byte case	else	import	package	synchronized	
catch	extends	instance	private	this	
char	final	of	protected	throw	
class	finally	int	public	throws	
const	float	interface	return	transient	
continue	for	long	short	try	

1.18 DECLARACIÓN DE UNA VARIABLE

Para declarar una variable es necesario especificar el tipo de datos que en ella serán almacenados y el nombre con que será reconocida dentro del programa. Su sintaxis es la siguiente:

```
Tipo_Dato  nombreVariable;
```

1.19 ÁMBITO DE UNA VARIABLE

Es indispensable declarar las variables antes de poder utilizarlas. En java las variables deben declararse ya sea dentro de una clase, al principio de un método o bloque de código o en el punto de utilización.

1.19.1 Variables declaradas dentro de una clase

Este tipo de variables recibe el nombre de ***variables miembro de la clase***. Su declaración se hace por dentro de las llaves **{}** que delimitan la clase y al mismo nivel de los métodos que pertenecen a dicha clase. Hecha la declaración de esta forma, estas variables estarán disponibles para todos los métodos de la clase. En el siguiente ejemplo podemos ver como los métodos calcularDefinitiva*(), calcularAcumulado() y calcularFaltante()* pueden hacer uso de las variables que han sido definidas dentro de la clase *Alumno*.

```
class Alumno
      {
            double Acumulado, Faltante, Definitiva;
            double nota1 = 3.5;
            double nota2 = 2.0;
```

```
        double nota3 = 1.5;
        double calcularDefinitiva()
                {
                        Definitiva = (nota1+nota2+nota3)/3;
                        return Definitiva;
                }
        double calcularAcumulado()
                {
                        Acumulado = Definitiva * 0.5;
                        return Acumulado;
                }
        double calcularFaltante()
                {
                        Faltante = (2.95 - Acumulado)*2;
                        return Faltante;
                }
}
```

1.19.2 Variables declaradas al principio de un método o bloque de código

Este tipo de variables recibe el nombre de variables locales y su ámbito o alcance está dado por el lugar donde fueron declaradas; si fueron declaradas al inicio de un bloque, serán reconocidas exclusivamente dentro de ese bloque y si fueron declaradas dentro de un método serán reconocidas exclusivamente dentro de ese método.

En el siguiente programa cuyo objetivo es calcular el valor de pago correspondiente que tiene que hacer un cliente a una cuota de un préstamo bancario, se puede observar la declaración de ***capital*** y ***cuotas*** como variables miembro de la clase ***Intereses,*** la declaración de ***intereses, pago*** y ***diasmora*** como variables locales del método ***calcularPago()*** e ***interesmora*** como variable local del bloque ***if*** perteneciente al método ***calcularPago().***

```
public class VariablesLocales
{
      public static void main(String[]args)
      {
            Intereses cliente1 = new Intereses();
            System.out.println("Valor a pagar "+cliente1.
            calcularPago());
```

```
        }
}
class Intereses
{
        double capital = 5000000; //Variable miembro de la clase
        int cuotas = 12;          //Variable miembro de la clase
        double calcularPago()
        {
                double interes = 0.015;  //Variable local al método
                double pago;             //Variable local al método
                int diasmora = 4;
                if (diasmora > 0)
                {
                        double interesmora = 0.03;  //Variable
                                                    local al bloque
                        System.out.println("Valor a pagar por
                        mora "+(diasmora*capital*interesmora)/30);
                }
                pago = (capital * interes)+(capital/cuotas);
                return pago;
        }
}
```

1.19.3 Variables declaradas en el punto donde se utilizan

Son también variables locales y son de gran utilidad en la construcción de bucles repetitivos, su alcance está limitado al bloque donde son declaradas.

En el siguiente ejemplo cuyo objetivo es convertir la distancia dada en kilómetros entre dos ciudades, se puede observar como en el método ***convertirMetros()*** se declara la variable ***metros*** justo en el punto donde es utilizada.

```
public class VariablesPunto
{
        public static void main(String[]args)
        {
                Kilometros tunjapaipa = new Kilometros();
                System.out.println("Distacia Tunja-Paipa en
                metros: "+tunjapaipa.convertirMetros());
        }
}
```

```
class Kilometros
{
      double km = 36.5;
      double convertirMetros()
      {
            double metros = km * 1000;  //Variable declarada
                                        en el punto //de
                                        utilización.
            return metros;
      }

}
```

1.20 MODIFICADORES PARA DECLARAR VARIABLES DE CLASE

Java permite los modificadores public, protected y private para las variables de clase. Las variables public tienen como ámbito cualquier clase del programa o paquete en que se encuentre. Las variables de tipo protected tienen como ámbito la clase en que fueron declaradas y en las clases derivadas de esta. Las variables de tipo private tienen como ámbito los métodos pertenecientes a la clase en que se declararon.

Las variables miembro de una clase que no tienen modificador son asumidas como variables públicas, es decir, su ámbito será todo el programa o paquete en que se encuentren. Para poder tener acceso a estas variables desde otra clase es necesario tener como argumento una referencia a un objeto de la clase en que la variable fue declarada.

En el siguiente programa cuyo objetivo es determinar el monto máximo a prestar a un cliente de un banco dependiendo de su salario, se puede ver cómo en los métodos mostrarCliente() y calcularMonto() de la clase Préstamo se pasa como argumento el objeto cliente1 de la clase Cliente; lo cual quiere decir que el método recibe como argumento un objeto de la clase Cliente con todas sus variables y métodos. Una vez recibidos estos objetos referencia, su utilización dentro del método respectivo se hace a través del operador punto. (`cliente1.cedula o cliente1.salariobasico)`

Para poder pasar objetos como argumentos hacia un determinado método, primero deben haber sido creadas sus respectivas instancias.

```
public class SistemaBancario
{
      public static void main(String[]args)
      {
          Cliente JuanPablo = new Cliente();
          Prestamo PrestamoConsumo = new Prestamo();
          System.out.println("JUAN PABLO");
          System.out.println("Salario Básico "+JuanPablo.
          salariobasico);
          System.out.println("Sueldo Neto "+JuanPablo.
          calcularSueldo());
          PrestamoConsumo.mostrarCliente(JuanPablo);
          System.out.println("Monto préstamo
          "+PrestamoConsumo.calcularMonto(JuanPablo));
      }
}
class Cliente
{
      double cedula = 7567340;
      double salariobasico = 2500000;
      double calcularSueldo()
      {
            double salud = salariobasico * 0.04;
            double pension = salariobasico * 0.03;
            double sindicato = salariobasico * 0.01;
            double sueldoneto = salariobasico-salud-
            pension-sindicato;
            return sueldoneto;
      }

}
class Prestamo
{
      double codigo = 1000;
      void mostrarCliente(Cliente cliente1)
      {
            System.out.println("Cédula: "+cliente1.cedula);
      }
      double calcularMonto(Cliente cliente1)
      {

            double monto = 2 * cliente1.salariobasico;
            return monto;
      }

}
```

1.21 CONSTANTES

Definimos una constante como un espacio de memoria que puede almacenar un tipo de dato pero cuyo valor no puede cambiar durante su existencia. Para hacer su definición se debe anteponer al tipo de datos que almacenará, la palabra *final*.

```
final tipo_datos nombre = VALOR;
```

```
class Longitud
{
	public double distanciaKm = 35.9;
	double convertirMetros()
	{
		final double FACTOR = 1000;
		return FACTOR * distanciaKm;
	}
}
```