Análisis del lenguaje JAVA

William David Hidalgo Rotavisqui.

Universidad del Valle.

Resumen

Este documento presenta un análisis del lenguaje java, donde se realiza un análisis de su sintaxis y su semántica como también su paradigma base que es el orientado a objetos, de este se describe los principales conceptos que bordan este paradigma como lo es el encapsulamiento, la herencia, el polimorfismo y la abstracción.

Introducción

El lenguaje de programación java es uno de los principales en el mercado y la industria este fue creado por n creado por Sun Microsystems. Java es un lenguaje que es creado con base al paradigma de programación orientado a objetos. Los programas bajo este paradigma funcionan por medio de la interacción entre instancias de clases.

1. Lenguaje java

Este es un lenguaje de programación de alto nivel, el cual para ser compilado necesita del javac.exe, un programa que se encuentra contenido en el Java Development Kit (JDK) que es el software que contiene las herramientas necesarias para la creación de programas Java. Por otra parte una de las características de este lenguaje es que puede ser compilado e interpretado en paralelo. El compilador es el encargado de convertir el código fuente de un programa d a código bytecode y la interpretación de este código se realiza utilizando la máquina virtual Java (JMV).

Java tiene como característica base la implementación del paradigma orientado a objetos, el cual describiremos más adelante.

2. Sintaxis y semántica

2.1. Identificadores

Los identificadores de las variables en java son una secuencia de caracteres las cuales pueden ser letras de A-Z y a-z o dígitos del 0-9. Estos identificadores deben iniciar por una letra o existe la excepción de poder utilizar los caracteres $ y _.

```
Ejemplos: cantidad, Cantidad,
    _Cantidad o $Cantidad.
```

Por otra parte existen palabras reservadas como tal del lenguaje por lo cual estas no pueden ser utilizadas como identificadores.

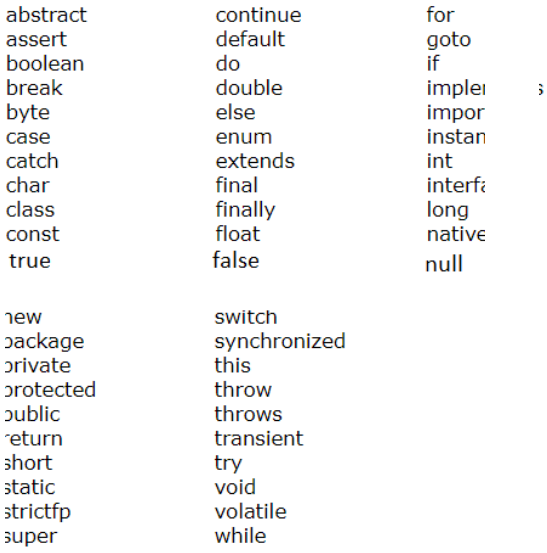

Fig.1. Palabras reservadas de java.

2.3 Valores

Los valores de variables están formados por una secuencia de caracteres, este valor no puede ser cambiado en algún punto de la

computación, este es un valor constante.

Existen diferentes tipos de valores constantes lo cuales pueden ser:

- Un solo carácter, este debe estar entre comillas simples y pueden ser alfanumérico o caracteres especiales (‘a’,’1’ o ’#’).
- Cadena de caracteres alfanuméricos y caracteres especiales, estos deben estar entre comillas simples (“hola”, “C0M03stas!” o “123Probando”).
- Numéricos (1,2.3, 19999999999)
- Booleanas (true o false).

2.4 Variables

En java se utiliza el término variables como la localidad en la memoria con la cual yo puedo almacenar datos.

Variable = Identificador asociado a otro identificador de variable o un valor constante.

Java es un lenguaje es tipeado, por lo que se debe especificar el tipo de dato para cada una de las variables que se vayan a utilizar. Este tipo de dato se utiliza en el momento en que se va a declarar una variable.

```
Tipodato + Identificadorvariable
```

2.5. Tipos de datos:

Tipo	Descripción
byte	Entero byte
short	Entero corto
int	Entero
long	Entero largo
float	Punto flotante
double	Punto flotante de doble presición
char	Un solo carácter
boolean	Un valor booleano

2.6. Operadores

Los operadores son símbolos que nos permiten combinarlos con variables para así poder crear expresiones. En java encontramos 4 tipos de operadores:

Operador	Significado	Ejemplo
	Operadores aritméticos	
+	Suma	a + b
-	Resta	a - b
*	Multiplicación	a * b
/	División	a / b
%	Módulo	a % b
	Operadores de asignación	
=	Asignación	a = b
+=	Suma y asignación	a += b (a=
-=	Resta y asignación	a -= b (a=a
*=	Multiplicación y asignación	a *= b (a=a
/=	División y asignación	a / b (a=a /
%=	Módulo y asignación	a % b (a=a
	Operadores relacionales	
==	Igualdad	a == b
!=	Distinto	a != b
<	Menor que	a < b
>	Mayor que	a > b
<=	Menor o igual que	a <= b
>=	Mayor o igual que	a >= b
	Operadores especiales	
++	Incremento	a++ (posti ++a (pre
--	Decremento	a-- (postde --a (prede
(tipo)expr	Cast	a = (int) b
+	Concatenación de cadenas	a = "cad1"
.	Acceso a variables y métodos	a = obj.va
( )	Agrupación de expresiones	a = (a + b

Fig.2. Lista de los operadores en java.

2.7. Expresiones

Las expresiones es la combinación de identificadores de variables, valores y operadores.

Ejemplo:

```
Exp=Identificadorvariable + Operador
+Valor (a=3 o b+2).

Exp= Identificadorvariable +
Operador+ Identificadorvariable  (a+b
o c=b).
```

Estructuras condicionales

Existen estructuras de sentencias condicionales como el **if** el cual evalua una expresión y si se obtiene un valor de verdad se realiza algo, también se incluye el si no **else** como opcional a la primera condición:

```
 if ( Exp){   código a ejecutar si se
        cumple la condición }

}   else   { código a ejecutar si se
cumple la condición }
```

Otra estructura condicional la sentencia **switch**, la cual compara el resultado de una expresión con una serie de valores diferentes y en el caso que se cumpla ejecuta el código que se encuentre dentro del case.

```
switch( resultdo exp o variable ){
case valor1:
   Código a ejecutar;
         break;

case  valorN:
   Código a ejecutar;
         break;
default:
}
```

2.8. Ciclos

Encontramos como estructura para realizar ciclos a la sentencia **for,** la cual utilizamos cuando deseamos repetir un fragmento de código varias veces. Este ciclo está compuesto por una inicialización de una variable la cual no permite controlar las iteraciones del ciclo. Una condición que es una exp que evalúa en cada iteración la variable que controla el ciclo y un incremento que es el que define como se modifica la variable en cada iteración.

```
for(inicialización;condición;incremen
to){
   Código a ejecutar;
}
```

Otros ciclos que podemos encontrar en java son while y do-while, estos iteran varias veces hasta que se cumpla su condición de salida.

```
while (condición){
   Código a ejecutar;
}
```

El ciclo do-while evalúa la condición al final del ciclo lo cual asegura que se ejecute por lo menos una vez el código que se encuentra dentro de ciclo.

```
do{
   código a ejecutar;
}while (condición);
```

2.9. Azúcar sintáctico:

Encontramos un ejemplo de azúcar sintáctico sobre el condicional if, este se le denomina if ternario y está compuesto por una **condición** luego el símbolo **?**, después el código que queremos que se ejecute si se cumple la condición y luego va el carácter **:,** y después el código que queremos que se ejecute si no se cumple la condición.

```
   condición ? código a ejecutar :
          código a ejecutar;
          c =  a > b ? 3 : 4
```

Otro ejemplo de azúcar que encontramos en java es la declaración de variables en paralelo.

```
int a,b,x,y=2;
```

2.10. Abstracción lingüística

Nos encontramos algo Abstracción lingüística como lo es la estructura de ciclo **foreach** o for extendido, este nos brinda una mayor utilidad cuando deseamos recorrer listas de objetos.

```
for( TipoVariable

nombreVariableTemporal :

nombreColección ) {

    Código a ejecutar;

  }
```

3. Paradigma Orientado a objetos.

La base del lenguaje java es el paradigma orientado a objetos, donde toda la programación es llevada a la interacción entre objetos de clases.

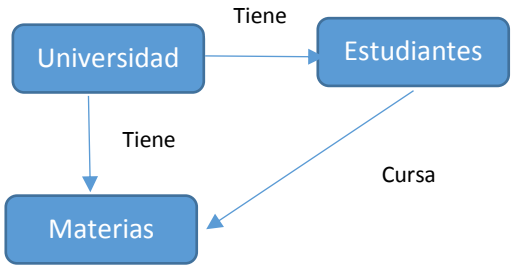

Fig.3. Diagrama conceptual de interacción de instancias de clases.

Un objeto es una encapsulación genérica de datos los cuales tienen un estado y un comportamiento. Pero estos objetos son instancias que se crean a partir de las clases, donde se especifican los métodos y las variables que definen las características comunes de todas las instancias de objetos.

Las clases nos permiten crear varias instancias de objetos donde cada instancia obtiene los atributos y métodos de la clase de donde son instanciados.

Definición de la estructura de una clase:

```
class nombreClase /* Declaración de
la clase */
{

 /* Aquí va la definición de
variables atributos y métodos */

}
```

3.1 Métodos

En java existe un método llamado Constructor, este es un tipo específico de método que debe llevar el mismo nombre que la clase, y se utiliza cuando se desean crear objetos de la clase, ósea para instanciar un objeto.

Existen métodos que retornan un objeto o métodos que no retornan, estos últimos se les debe agregar la palabra reservada **void**.

Definición estructura de método:

```
<Modificado_Acceso>
valorRetorno(Tipo de dato o una
Clase) nombreMetodo( <lista de
argumentos> )
   {
      /* Cuerpo del método */

sentencia return valorRetorno (tipo
dato o Objeto de clase) ;
}
```

El acceso a los métodos se hace por medio de la previa creación de un objeto y utilizando el operador punto (.).

```
nombreObjeto.nombreMetodo( <lista de
      argumentos opcionales> );
```

Los atributos también so accedidos por medo del operador (.).

```
nombreObjeto.nombreVariable;
```

3.2 Encapsulamiento

En la estructura del método nos encontramos con un término llamado modificador de acceso, esto nos introduce a uno de los conceptos utilizados en la POO, el **encapsulamiento**, este busca de alguna forma controlar el acceso a los datos que conforman un objeto (métodos y atributos), dando así un nivel de seguridad de acceso a los atributos o métodos.

Modificadores de acceso:

Prívate: Cualquier elemento de una clase puede ser accedido únicamente por la misma.
Public: El elemento puede ser accedido desde cualquier clase.
Protected: Los elementos pueden ser accedidos desde su mismo paquete y desde cualquier clase que extienda la clase en que se encuentra, independientemente de si esta se encuentra en el mismo paquete o no.

3.4. Herencia

La herencia es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos. Java solo permite la **herencia simple**, la cual solo permite definir nuevas clases a partir de una sola clase. Este concepto de herencia permite definir la estructura jerárquica entre clase padre y clase hija, donde la clase padre para la clase hija es la súper clase, y la clase hija para la clase padre es llamada subclase.

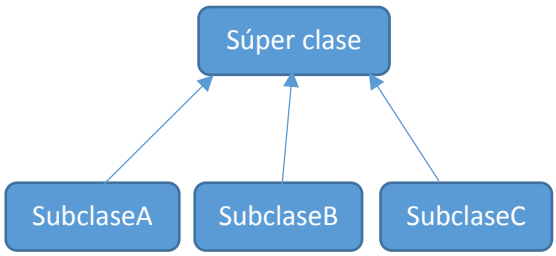

Fig.4. Diagrama que representa el concepto de herencia.

Una Superclase puede tener varias Subclases pero una Subclase solo puede tener una Superclase.

```
public class Animal
{
int cantPatas;
public int contarPatas(){..}

}
```

```
public class Perro extends Animal {

String  Raza;
public int contarPatas(){..}

}
```

Se utiliza la palabra reservada **extends** para indicar de qué clase hereda.

3.5. Polimorfismo

Permite definir distintos comportamientos para un método dependiendo de la clase sobre la que se realice la implementación. Nos permite enviar mensajes sintácticamente iguales a objetos de tipos distintos.

3.6. Abstracción

Otro de los conceptos principales de la POO es la abstracción la cual se enfoca en la visión externa de un objeto, separa el comportamiento específico de un objeto utilizando una interfaz, esta muestra solo comportamiento específico del objeto. Se utiliza la palabra reservada **implements** para poder implementar la abstracción.

```
public interface  Mascota
{
public abstract void vacunar();

}

public class Perro extends Animal
implements Mascota {

String  Raza;
public int contarPatas(){..}
public int vacunar(){..}
}
```

3.7. Instancias de objetos

Para realizar la instancia de un objeto en java se hace uso de los métodos constructores definidos en la clase. Esta instancia puede hacerse con parámetros de entrada que son los atributos de la clase o sin parámetros.

Estructura de instancia:

Constructor sin parámetros:

```
Clase nombreObjeto = new  Clase();
```

Constructor con parámetros:

```
Clase nombreObjeto = new Clase(Tipodato Atributo1, TipoDato Atributo2);
```

El operador new crea una instancia de una clase asignando la cantidad de memoria necesaria de acuerdo al tipo de objeto.

En java se puede crear un objeto de una clase referenciado a otro objeto de la misma clase ya existente.

```
Ejemplo; Animal a1= new Animal ();
         Animal a2 = a1;
```

Todo lo que afecte a **a1** afecta a **a2** ya que están referenciados en el mismo espacio de memoria.

4. Conclusiones

A pesar de java ser uno de los principales lenguajes de programación en el mercado, vemos que está limitado por el paradigma orientado objetos, como por ejemplo una de las principales es el manejo del concepto de la concurrencia, este no es natural en java, se debió adaptarlo y utilizar el concepto de monitores los cuales son complejos de programar. Al utilizar hilos en java existe la limitación de no poder implementarlos de manera no determinística ya que definir un hilo dependiente de un valor que se liga después de la ejecución del hilo causaría un error, ya que en este lenguaje no se puede suspender el hilo hasta que se ligue el valor que necesita para continuar.

Java está ubicado en el grupo de la programación imperativa y alejándose cada vez más de los beneficios del paradigma declarativo, como por ejemplo la ventaja del tipamiento dinámico lo cual hace que el lenguaje sea mucho más flexible y se desperdicie tiempo en compilación.

Por otra parte java al utilizar POO adquiere ciertas ventajas como la fiabilidad ya que se puede dividir el problema en partes más pequeñas y probarlas de manera independiente que permitirá aislar mucho más fácilmente los posibles errores. También cierta ventaja en la reusabilidad clases en distintas partes del programa y la mantenibilidad para abstraer el problema, ya que son programas más sencillos de leer y comprender.