Tema 2: Tipos, expresiones y control de flujo

Objetivo: El alumno conocerá los elementos básicos del lenguaje, así como los aspectos teóricos relacionados.

2.1 Generalidades

Tipado de lenguajes

El análisis de tipos tiene por objetivo asegurar que el uso de los objetos definidos es correcto: esto es, que su uso se atiene a la semántica de su definición.

La asociación de un tipo a un objeto se le conoce como tipado.

Trata de evitar que puedan utilizarse abstracciones de maneras no previstas

2.1 Generalidades

Tipado de lenguajes

Encontramos 3 Clasificaciones para el tipado de los lenguajes

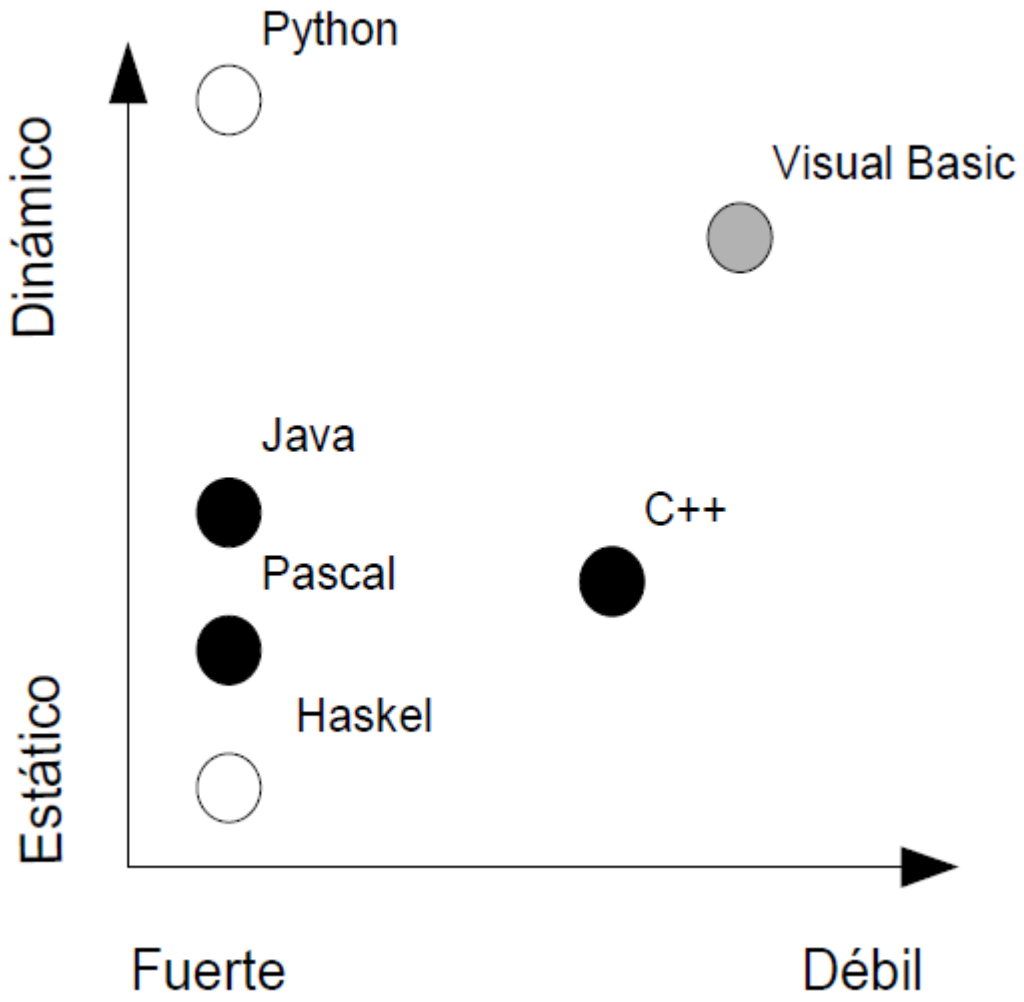
- ✓Tipado Fuerte
- ✓Tipado débil

2.1 Generalidades

- ✓Tipado Estático
- ✓Tipado Dinámico

- ✓Tipado explícito
- ✓Tipado implícito

2.1 Generalidades



2.1 Generalidades

Identificadores

- Un identificador es el nombre de algún componente de programa, clase, método, constante, variable o a cualquier otro elemento del programa que necesite nombrarse.

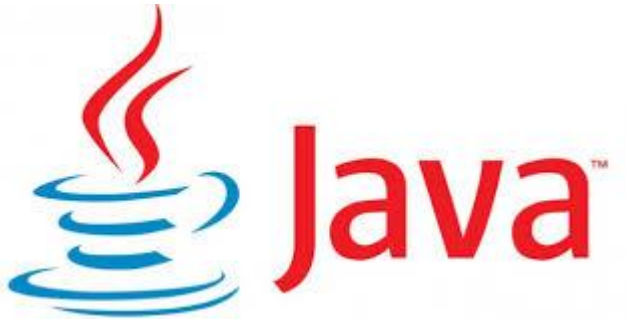
2.1 Generalidades

- Los identificadores en válidos en java se componen de letras, números, guion bajo y el signo $, y no pueden iniciar con un número
- Por convención, los nombres de las clases comienzan con una letra mayúscula…

2.1 Generalidades

Palabras reservadas

Las palabras reservadas son identificadores reservados predefinidos que tienen un significado especial y no se pueden utilizar como identificadores en sus programas.



https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

2.1 Generalidades

Palabras reservadas

abstract	char	float	interface	public
boolean	class	for	long	static
break	do	if	new	super
byte	else	implements	package	try
case	extends	import	private	void
catch	final	int	protected	while

2.1 Generalidades

Comentarios

```
1  /*************************************************
2   * HolaMundo.java
3   * @author Edgar
4   * El programa imprime un saludo
5   *************************************************/
6  public class HolaMundo {
7
8      /**
9       * @param args the command line arguments
10      */
11     public static void main(String[] args) {
12         System.out.println("Hola Mundo");
13     }
14
15 } //Fin de la clase
16
```

2.1 Generalidades

Descripción de una clase y objetos

La forma general de definir una clase es la siguiente:

[Modificadores] **class** *[IdentificadoDeLaClase]*{

//atributos

//constructores

//métodos

}

2.1 Generalidades

Ejemplo

```
class Point{
    int x;
    int y;

}
```

```
class Alumno{
    long numCuenta;
    String nombre;
    String apellido;
    int edad;

}
```

```
class Perro{
    String nombre;
    String color;
    String raza;
    int edad;
}
```

2.1. Generalidades

Referencias o instancias

Se utiliza la palabra new para instanciar una clase, seguido del nombre de la clase y paréntesis

```
Point p = new Point()
Point miPunto = new Point()
Alumno alu1 = new Alumno()
Perro miPerro = new Perro()
```

2.1 Generalidades

```
class TestPoint {
    public static void main(String[] args){
        System.out.println("Creando un punto...");
        Point p = new Point();
        System.out.println("Inicializando atributos");
        p.x=4;
        p.y=10;
        System.out.println("Imprimiendo objeto");
        System.out.println("Punto p("+ p.x + "," + p.y + ")");

    }

}
```

2.2 Tipos de datos

Tipos primitivos

Java cuenta con 8 tipos primitivos

byte	8 bit	signed
short	16-bit	signed
int	32-bit	signed C2
long	64-bit	signed C2

2.2 Tipos de datos

Tipos primitivos

Java cuenta con 8 tipos primitivos

float	32-bit	IEEE-754
double	64-bit	IEEE-754
boolean	1-bit?	true/false
char	16-bit	UTF-16

2.2 Tipos de datos

Los tipos primitivos tienen valores de inicialización por defecto

Tipo de dato	Valor de inicialización
boolean	false
byte, short, int, long	0
float, double	0.0
objetos (referencias)	null

2.2 Tipos de datos

Promoción Numérica

Al realizar operaciones con tipos de datos primitivos Java automáticamente realiza la promoción correspondiente siguiendo 4 reglas

1.- Si dos valores tienen diferente tipo, se promueve uno de los dos, al más grande.

2.- Si uno es entero y otro flotante, automáticamente se promociona a punto flotante

2.2 Tipos de datos

Promoción Numérica

3.- Los más pequeños (byte, short, char) se promueven a entero en operaciones aritméticas

4.- El resultado de las operaciones será el mismo que el de los operandos que se involucren

2.2 Tipos de datos

Ejemplos

```
int x = 1;                  x*y?
long y = 33;

double x = 39.21;           x+y?
float y = 2.1f

short x = 10;               x/y?
short y = 3;
```

2.2 Tipos de datos

Moldeado o Casting

Entre tipos primitivos se realiza cuando se quiere indicar explícitamente que el resultado es un tipo de dato “menor”

1)
```
float x = 55.4
float y = 100.0
int z = (int)(x*y);
```

2)
```
short x = 10;
short y = 3;
short z = (short)x*y;
```

2.2 Tipos de datos

Operadores Unarios

Realizan las tareas más simples como incrementos, decrementos, cambio de signo o negación de un valor “boolean”

Post unarios	`expresión++, expresión--`
Pre unarios	`++expresión, --expresión`
unarios	`+, -, !`

2.2 Tipos de datos

Operadores Aritméticos

Java cuenta con los operadores de las operaciones convencionales

Adición, sustracción, multiplicación

División, módulo

2.2 Tipos de datos

Operadores relacionales

Devuelven un valor “boolean”

`<`	Estrictamente menor que
`<=`	Menor o igual
`>`	Estrictamente mayor que
`>=`	mayor o igual
`instanceof`	determina si un objeto es una instancia de un determinado tipo

2.2 Tipos de datos

Operadores de asignación

```
x += 3;          x = x + 3;
x -= 4;          x = x - 4;
x *= y;          x = x * y;
x /= 4;          x = x / 4;
x %= 16;         x = x % 16;
x *= y + 1;      x = x * (y +1);
```

2.2 Tipos de datos

Precedencia de operadores

Unarios	++a, a++, !var
Multiplicativos	*, /, %
Aditivos	+, -
Relacionales	<,>,=>,<=, ==
Lógicos	&, !, ^
Asignación	=, +=, -=

2.2 Tipos de datos

Evaluar la siguiente expresión

```
int a = 10;
int b = 20;
int c = 5;
c += 10 + --a * b - 30 / c % 4 ;

Resultado:
                    c = 193
```

2.2 Tipos de datos

Evaluar la siguiente expresión

```
int a = 5;
int b = 10;
int c = 15;
c += 5 + --a * b++ - 10 * 2 + ++b ;
Resultado:
                    c = 52
```

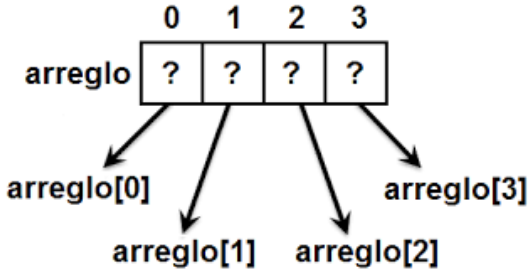
2.3 Arreglos

Colecciones de elementos del mismo tipo.

Declaración de arreglos en Java:

```
tipo_dato[] identificador
int [] arreglo1;
long [] arreglo2;
```

Lo que ella piensa…



Lo que tu piensas…

2.3 Arreglos

La creación de los arreglos se realiza con la palabra "**new**"

```
arreglo1 = new int[20];
arreglo2 = new long[10];

String [] telefonos new String[10];
int calificaciones[]= new int[6];
```

2.3 Arreglos

En java, un arreglo es considerado un objeto.

Los arreglos son referencias por lo que un arreglo contiene la dirección del primero de sus elementos.

Inicialización:

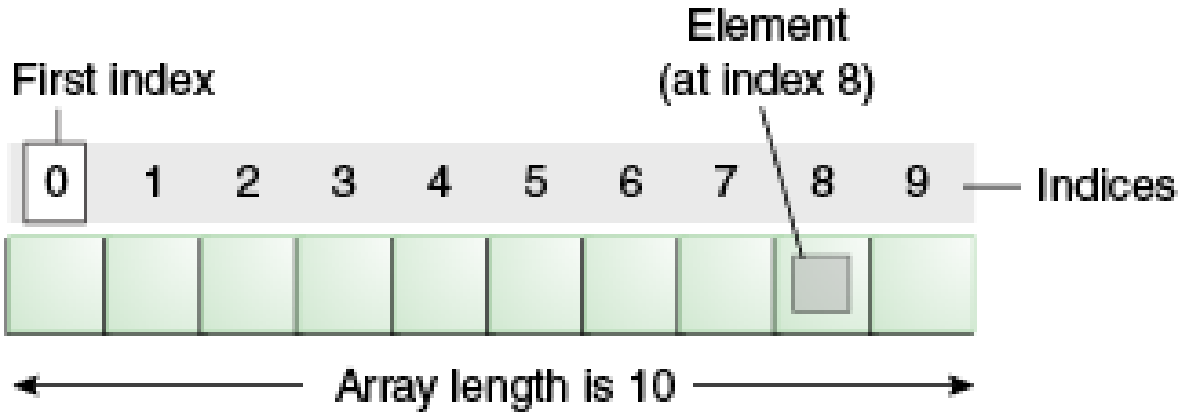
```
int [] numPrimos = {1,3,5,7,11,13};
String[] animales = {“perro”,“gato”,“ratón”};
```

2.3 Arreglos

Usando arreglos

El manejo de arreglos se realiza a través de sus índices.

Al ser objetos cuentan con atributos y métodos, uno de los más importantes es *length()*



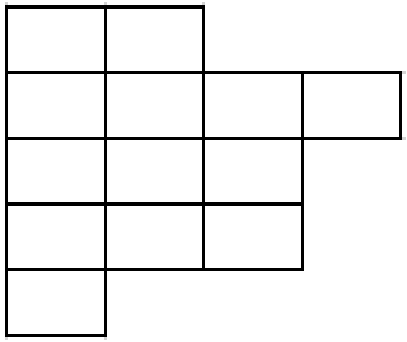
2.3 Arreglos

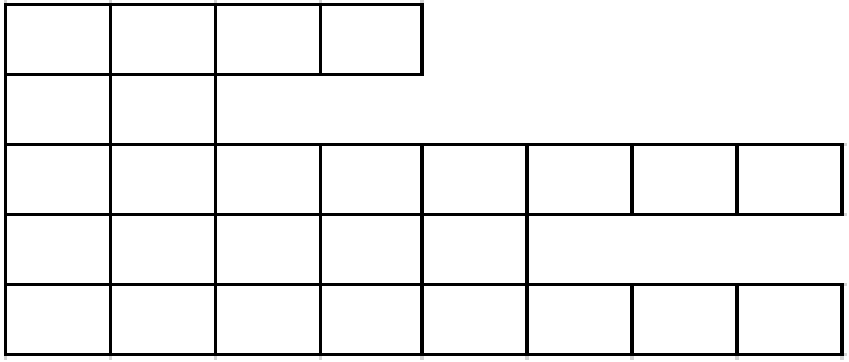
```
String animales = {“perro”,“gato”,“ratón”};
System.out.println(animales.length)
System.out.println(animales[0]);
System.out.println(animales[2]);
int size = animales.length;
animales[animales.length];
animales[animales.length-1];
```

2.3 Arreglos

Arreglos irregulares

En java, es posible crear arreglos irregulares:





2.4 Tipos y ámbito de las variables

Constantes

Son elementos cuyo valor no puede ser modificado.

Para declarar una constante en Java, se utiliza la palabra reservada final seguida del tipo de dato

```
final double PI = 3.141519;
final long VELOCIDAD_LUZ = 299792458;
```

2.4 Tipos y ámbito de las variables

Variables estáticas.

Este tipo de variables son compartidas por todos los objetos de la clase.

El valor de la variable es el mismo para todos los objetos y si alguno la modifica en el flujo del programa, la modificación afecta a todos los objetos de esa clase.

2.4 Tipos y ámbito de las variables

Variables estáticas.

- ✓ Una variable estática pertenece a una clase, no un objeto
- ✓ De ahí surge la diferencia entre variable de clase y variable de instancia
- ✓ Las variables estáticas son inicializadas solo una vez
- ✓ Se puede acceder directamente y no necesita la referencia de un objeto

2.5 Estructuras de Selección

2.5.1 Estructura if

Funciona de manera similar a otros lenguajes de programación

```
if (expresionBooleana){
   // Hacer esto
}
```

2.5 Estructuras de Selección

2.5.1 Estructura if-else

```
if (expresionBooleana){
   // si es verdadero…
}
else{
   // si es falso…
}
```

2.5 Estructuras de Selección

2.5.2 El operador Ternario (? :)

Es el único que toma tres operandos

```
expBooleana ? expresión1 : expresión 2;
```

El primer operador es una expresión booleana, en el segundo y el tercero puede haber cualquier expresión que regrese un valor.

El operador ternario es básicamente un if-then condensado

2.5 Estructuras de Selección

```
int x, y = 10;
if(y  > 5) {
     x = 2 * y;
} else {
     x = 3 * y;
}
```

```
int y = 10;
int x = (y > 5) ? (2 * y) : (3 * y);
```

2.5 Estructuras de Selección

2.5.2 El operador switch

```
switch (variable){
   case exp1:
        //acciones;
        break;
   case exp2:
        //acciones;
        break;
   default:
        //acciones;
}
```

2.5 Estructuras de Selección

Tipos de datos que soporta la estructura switch:

- int
- byte
- short
- char
- String

Los valores de cada una de las opciones (case) deben ser del mismo tipo de dato del switch, además de ser conocidos en tiempo de compilación

2.5 Estructuras de Selección

```
int dayOfWeek = 5;
switch(dayOfWeek) {
  default:
    System.out.println("Weekday");
    break;
  case 0:
    System.out.println("Sunday");
    break;
  case 6:
    System.out.println("Saturday");
    break;
}
```

2.5 Estructuras de Selección

```
int dayOfWeek = 5;
switch(dayOfWeek) {
  case 0:
    System.out.println("Sunday");
  default:
    System.out.println("Weekday");
  case 6:
    System.out.println("Saturday");
    break;
}
```

2.5 Estructuras de Repetición

- **while**

```
while(booleanExpression) {

      // Body

}
```

- **do-while**

```
do {

      // Body

} while (booleanExpression);
```

2.5 Estructuras de Repetición

- **ciclo “for”**

```
for(inicio;boolExpr;actualiza){
    //instrucciones
}
```

2.5 Estructuras de Repetición

- **for-each**

Un ciclo for-each es una versión modificada del ciclo “for” tradicional. Puede usarse siempre que es necesario iterar a través de todos los elementos en una colección de datos.

```
for(tipoDato variable : collection){
    //instrucciones
}
```

2.5 Estructuras de Repetición

ejemplo

```
int[] arr = {1,2,3,4,5,6,7,8,9,10};
for(int valor : arr){
     System.out.println(valor*2);
}
```