PROGRAMACIÓN EN JAVA

Materia PJA-001 material creado por el Ing. Jim Requena

Fundamentos de Java I (primera parte)

Materia PJA-001 material creado por el Ing. Jim Requena

Objetivos de la sesión

- Comprender los tipos de datos en Java
- Aprender a declarar y usar variables
- Entender los operadores básicos
- Practicar la entrada y salida básica en Java

Tipos de datos en Java

Java tiene dos categorías de tipos de datos:

1. Tipos primitivos
2. Tipos de referencia

Tipos primitivos

- byte: 8 bits, ej: 127
- short: 16 bits, ej: 32767
- int: 32 bits, ej: 2147483647
- long: 64 bits, ej: 9223372036854775807L
- float: 32 bits, ej: 3.14f
- double: 64 bits, ej: 3.14159265359
- boolean: true o false
- char: 16 bits, ej: 'A'

Declaración de variables

Sintaxis: `tipo nombreVariable = valor;`

Ejemplos:

```
int edad = 25;
double altura = 1.75;
boolean esEstudiante = true;
char inicial = 'J';
```

Tipos de referencia

- Objetos de clases
- Arrays

```java
String nombre = "Juan";
int[] numeros = {1, 2, 3, 4, 5};
```

```java
// Clase
String nombre = new String("Java");

// Array
int[] numeros = new int[5];

// Clase personalizada
MiClase objeto = new MiClase();

// Interfaz
List<String> lista = new ArrayList<>();

// Enumeración
enum DiaSemana { LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO }
DiaSemana dia = DiaSemana.LUNES;
```

Tipos de referencia enum

Puntos clave sobre enums:

1. Son tipos seguros: solo puedes asignar uno de los valores predefinidos.
2. Tienen un método `values()` que devuelve un array de todos los valores.
3. Tienen un método `valueOf(String)` que convierte un String al enum correspondiente.
4. Pueden tener campos, constructores y métodos.
5. Son útiles en switch statements.
6. Mejoran la legibilidad del código al usar constantes nombradas en lugar de números mágicos.

Tipos de referencia enum

```
// Enumeración
enum DiaSemana { LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO }
DiaSemana dia = DiaSemana.LUNES;
```

La primera línea define el enum `DiaSemana` con sus constantes. La segunda línea crea una variable `dia` de tipo `DiaSemana` y le asigna el valor `LUNES`.

Los enums son útiles cuando tienes un conjunto fijo de valores relacionados.

1. Ejemplo con colores:

```
enum Color { ROJO, VERDE, AZUL }
```

```
Color miColor = Color.VERDE;
System.out.println("Mi color favorito es " + miColor);
```

Tipos de referencia enum

2. Enums en estructuras de control:

```
enum EstadoSemaforo { VERDE, AMARILLO, ROJO }

EstadoSemaforo estado = EstadoSemaforo.AMARILLO;

switch (estado) {
    case VERDE:
        System.out.println("Puedes avanzar");
        break;
    case AMARILLO:
        System.out.println("Prepárate para detenerte");
        break;
    case ROJO:
        System.out.println("Detente");
        break;
}
```

Tipos de referencia enum

3. Enums con métodos y constructores:

```
enum Planeta {
    MERCURIO(3.303e+23, 2.4397e6),
    VENUS(4.869e+24, 6.0518e6),
    TIERRA(5.976e+24, 6.37814e6);

    private final double masa;   // en kilogramos
    private final double radio;  // en metros

    Planeta(double masa, double radio) {
        this.masa = masa;
        this.radio = radio;
    }

    public double gravedad() {
        double G = 6.67300E-11;
        return G * masa / (radio * radio);
    }
}

System.out.println("La gravedad en la Tierra es " +
Planeta.TIERRA.gravedad() + " m/s^2");
```

Operadores en Java

- Aritméticos: +, -, *, /, %
- Relacionales: ==, !=, <, >, <=, >=
- Lógicos: &&, ||, !
- Asignación: =, +=, -=, *=, /=, %=

El operador == compara valores para tipos primitivos y referencias de memoria para objetos. Es importante entender esta distinción para evitar errores sutiles en la comparación de objetos.

Ejemplos de operadores

```java
int a = 5, b = 3;
int suma = a + b;  // 8
boolean esMayor = a > b;  // true
boolean condicion = (a > 0) && (b < 5);  // true
a += 2;  // a ahora es 7
```

El operador == en Java

Es un operador de comparación que se utiliza para verificar si dos valores son iguales. Sin embargo, su comportamiento es diferente dependiendo de si se está usando con tipos primitivos o con tipos de referencia.

1.Para tipos primitivos:

El operador == compara el valor real de las variables.

```
int a = 5;
int b = 5;
boolean sonIguales = (a == b);   // sonIguales será true
```

Entrada y salida básica

Salida:

```
System.out.println("Hola, mundo!");
System.out.print("Esto no tiene salto de línea");
```

Entrada (requiere importar Scanner):

```
import java.util.Scanner;

Scanner scanner = new Scanner(System.in);
System.out.print("Ingrese su nombre: ");
String nombre = scanner.nextLine();
```

Ejercicio práctico

Crear un programa que:

1. Pida al usuario su nombre y edad
2. Calcule en qué año el usuario tendrá 100 años
3. Imprima un mensaje con esta información

Solución del ejercicio

```java
import java.util.Scanner;
import java.time.Year;

public class Edad100 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Ingrese su nombre: ");
        String nombre = scanner.nextLine();

        System.out.print("Ingrese su edad: ");
        int edad = scanner.nextInt();

        int anioActual = Year.now().getValue();
        int anio100 = anioActual + (100 - edad);

        System.out.println(nombre + ", tendrás 100 años en el año " + anio100);
    }
}
```

Tarea

1. Modifica el programa para que también calcule cuántos años faltan para que el usuario tenga 100 años
2. Agrega validación para asegurarte de que la edad ingresada sea un número positivo
3. Sube tu solución al repositorio del curso

Recursos adicionales

- Documentación de Java sobre tipos de datos: docs.oracle.com/javase/tutorial/java/nutsandbolts/datat y
- Tutorial sobre operadores en Java: www.w3schools.com/java/java_operators.asp

¡Gracias!