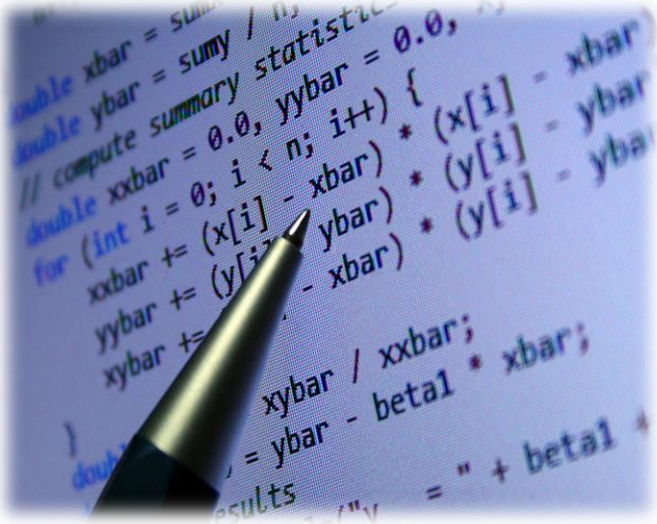
PROGRAMACIÓN ORIENTADA A OBJETOS

TEMA: USO DE DECIMALES EN JAVA



DOCENTE: Ing. Mónica Bonilla, MsC.
ESTUDIANTE: Ariel Alejandro Calderón

11-09-2024

UEB

El Uso de Decimales en Java: Precisión y Rendimiento en el Cálculo Numérico

Tipos de Datos para el Manejo de Decimales en Java

a. float y double

En Java, los tipos primitivos *float* y *double* son los más utilizados para el manejo de números con decimales en operaciones que no requieren una precisión extrema.
Si bien estos tipos son eficientes en términos de rendimiento y espacio, no son apropiados para cálculos financieros o cualquier aplicación donde se necesite precisión exacta, debido a los **errores de redondeo** inherentes al uso de punto flotante.
Por ejemplo, operaciones aritméticas aparentemente sencillas, como sumar 0.1 y 0.2, pueden dar resultados inesperados:

```
public class DecimalExample {
    public static void main(String[] args) {
        double a = 0.1;
        double b = 0.2;
        System.out.println(a + b); // Resultado inesperado: 0.30000000000000004
    }
}
```

b. BigDecimal

Para situaciones en las que se requiere precisión numérica exacta, como en cálculos financieros, Java proporciona la clase *BigDecimal*. Esta clase forma parte del paquete *java.math* y está diseñada para evitar los problemas de precisión asociados con los tipos de punto flotante. *BigDecimal* permite representar números con una **precisión arbitraria**, utilizando un objeto que maneja internamente los decimales como una cadena de caracteres y realiza las operaciones aritméticas de manera precisa.

Un ejemplo:

```
import java.math.BigDecimal;

public class BigDecimalExample {
    public static void main(String[] args) {
        BigDecimal a = new BigDecimal("0.1");
        BigDecimal b = new BigDecimal("0.2");
        BigDecimal result = a.add(b);
        System.out.println(result); // Resultado exacto: 0.3
    }
}
```

Ventajas de BigDecimal

- **Precisión:** Es ideal para aplicaciones que requieren una precisión exacta, como el cálculo de impuestos, intereses bancarios y cualquier operación financiera.
- **Escalabilidad:** BigDecimal permite la representación de números con una cantidad arbitraria de cifras decimales, lo que es crucial en cálculos científicos.

Desventajas de BigDecimal

- **Rendimiento:** Debido a que BigDecimal no es un tipo primitivo y se maneja como un objeto, las operaciones aritméticas con esta clase son significativamente más lentas en comparación con los tipos float y double.
- **Complejidad:** El uso de BigDecimal requiere una mayor cantidad de código, ya que las operaciones aritméticas (como la suma o multiplicación) no se pueden realizar directamente con operadores como +, sino que se deben invocar métodos específicos como add(), subtract(), multiply(), etc.

CONSULTA: ¿Cuál es el error en este código?

```
package holamundo;
public class HolaMundo {
    public static void main(String[] args) {
        float dato1, dato2, resultado;
        dato1 = 20.5;
        dato2 = 10.3;
        resultado = dato1 + dato2;
        System.out.println(dato1 + "+" + dato2 + "=" + resultado);
        // Resta
        resultado = dato1 - dato2;
        System.out.println(dato1 + "-" + dato2 + "=" + resultado);
        // Multiplicación
        resultado = dato1 * dato2;
        System.out.println(dato1 + "*" + dato2 + "=" + resultado);
        // División
        resultado = dato1 / dato2;
        System.out.println(dato1 + "/" + dato2 + "=" + resultado);
    }
}
```

El problema en el código está relacionado con el manejo de los literales de punto flotante. Por defecto, los literales de punto flotante en Java se interpretan como valores de tipo double, no como float. Cuando se intenta asignar un literal de tipo double a una variable float sin realizar una conversión explícita, el compilador generará un error.

Para corregir este problema, se debe utilizar el sufijo f para indicar que los literales son de tipo float. A continuación, se muestra la versión corregida del código:

```java
package holamundo;
public class HolaMundo {

    public static void main(String[] args) {
        float dato1, dato2, resultado;
        dato1 = 20.5f; // Use 'f' suffix for float literals
        dato2 = 10.3f; // Use 'f' suffix for float literals

        // Suma
        resultado = dato1 + dato2;
        System.out.println(dato1 + "+" + dato2 + "=" + resultado);

        // Resta
        resultado = dato1 - dato2;
        System.out.println(dato1 + "-" + dato2 + "=" + resultado);

        // Multiplicación
        resultado = dato1 * dato2;
        System.out.println(dato1 + "*" + dato2 + "=" + resultado);

        // División
        resultado = dato1 / dato2;
        System.out.println(dato1 + "/" + dato2 + "=" + resultado);
    }
}
```

Webgrafía

1. *Programandolo o intentandolo*. *Limitar el número de decimales de un double o un float en Java*. Recuperado Septiembre 9, 2024, desde https://programandoointentandolo-com.
   Descripción: Artículo que explica como trabajar con decimales en Java.
2. *Medium*. *Java desde 0: Números Decimales*. Recuperado Septiembre 9, 2024, desde https://classy-bear.medium.com.
   Descripción: Explicación más a fondo sobre los decimales en Java.