Instituto Tecnológico Superior del Oriente del Estado De Hidalgo

“ITESA”

Programa Educativo:

Ingeniería en Sistemas Computacionales

Asignatura y Tema:

Métodos Numéricos, Tema 1

Nombre del Docente:

M. T. I. Efrén Rolando Romero León

Nombre y Matricula del Alumno:

Velazquez Solis Mario David – 23030006

2 de febrero de 2025

Ejercicio

Cuarto Semestre 4F21 Grupo I Enero - Junio 2025

Parte 1:

```
int resultado = 1;

for (int i = 1; i<=5 ; i++) {
       resultado = resultado * 10;
       System.out.println(resultado);
}

System.out.println("No formateada: " + 2.55*resultado);
System.out.printf("Formateada: "+"%.0f%n", 2.55 * resultado);
```

Objetivo:

El objetivo de este código es demostrar el uso de un bucle for para multiplicar un valor inicial por 10 varias veces y luego aplicar operaciones matemáticas con formato de salida.

Inicialización de Variables:

- Se declara una variable resultado de tipo int e inicializa con el valor 1.
- El bucle for se ejecuta 5 veces (de i = 1 a i = 5).
- En cada iteración, resultado se multiplica por 10, lo que genera una progresión de multiplicación por potencias de 10.
- El valor de resultado se imprime en cada iteración con System.out.println().

Cálculos y Formateo de Salida:

- Después del bucle, se calcula el valor de 2.55 * resultado, primero mostrando el resultado sin formato.
- Luego, se usa System.out.printf() para imprimir el mismo cálculo, pero con formato de número entero, redondeando el valor a cero decimales usando "%.0f".

Resultados Esperados:

Impresión de los valores de resultado en el bucle:

```
Iteración 1: 10
Iteración 2: 100
Iteración 3: 1000
Iteración 4: 10000
Iteración 5: 100000
```

Cálculos:

Después del bucle, se realiza la multiplicación 2.55 * resultado. Dado que resultado es 100,000 después del bucle, el cálculo será 2.55 * 100000 = 255000.

En la salida formateada con printf, el valor será mostrado como 255000, sin decimales.

Conclusión:

Este código muestra cómo se pueden realizar multiplicaciones sucesivas dentro de un bucle y cómo manejar la salida de datos, tanto sin formato como con formato específico (en este caso, redondeando a números enteros). La principal enseñanza aquí es el uso de operaciones matemáticas dentro de un bucle y la aplicación de formato a la salida con System.out.printf().

Parte 2:

```
float sumaFloat = 0.0f;
double sumaDouble = 0.0;

for (int i = 0; i < 10; i++) {
       sumaFloat += 0.1f;
       sumaDouble += 0.1;
}

System.out.println("Resultado del florat: "+ sumaFloat);
System.out.println("Resultado con Double: "+ sumaDouble);
System.out.println("Error absoluto con float: "+ Math.abs(1.0 - sumaFloat));
System.out.println("Error absoluto con double: "+ Math.abs(1.0 - sumaDouble));
```

Objetivo:

El objetivo de este código es comparar las operaciones de suma con tipos de datos float y double, y calcular el error absoluto entre el resultado esperado y el resultado obtenido.

Se declaran dos variables:

sumaFloat, de tipo float, inicializada con 0.0f.

sumaDouble, de tipo double, inicializada con 0.0.

El bucle for se ejecuta 10 veces, incrementando las variables sumaFloat y sumaDouble en 0.1 en cada iteración:

sumaFloat += 0.1f (se usa el tipo float para la suma).

sumaDouble += 0.1 (se usa el tipo double para la suma).

Impresión de Resultados:

Después del bucle, se imprimen los resultados de las sumas con System.out.println().

También se calcula el error absoluto entre 1.0 y los resultados de las sumas usando Math.abs(), para observar la diferencia acumulada debido a la precisión de los tipos float y double.

Resultados Esperados:

- Con float: Después de 10 iteraciones de sumar 0.1, el valor esperado sería 1.0. Sin embargo, debido a las limitaciones de precisión de los números float, el valor calculado podría no ser exactamente 1.0.
- Con double: Después de 10 iteraciones de sumar 0.1, el valor debería ser exactamente 1.0, ya que los double tienen mayor precisión que los float.

Cálculo del Error Absoluto:

- Se utiliza Math.abs(1.0 - sumaFloat) y Math.abs(1.0 - sumaDouble) para calcular el error absoluto, que muestra la diferencia entre el valor esperado (1.0) y el resultado obtenido con cada tipo de dato.
- Se espera que el error con float sea mayor que con double, debido a la menor precisión de float.

Resultados Esperados en Consola:

- Resultado con float: Probablemente algo cercano a 1.0, pero no exactamente.
- Resultado con double: Exactamente 1.0.
- Error absoluto con float: Un valor pequeño, pero mayor que el error con double.
- Error absoluto con double: Muy cercano a 0.0.

Conclusión:

Este código muestra la diferencia de precisión entre los tipos de datos float y double al realizar operaciones aritméticas repetidas. Mientras que float tiene una menor precisión y puede generar errores acumulados, double mantiene una mayor exactitud, lo que se refleja en los cálculos y en el menor error absoluto al comparar con el valor esperado.