Programación orientada a objetos

Flujos de control

Dr. (c) Noé Alejandro Castro Sánchez

Ámbito de variables

- Área del programa donde la variable existe y puede ser utilizada.
- El ámbito se determina por la ubicación de las llaves **{ }**.

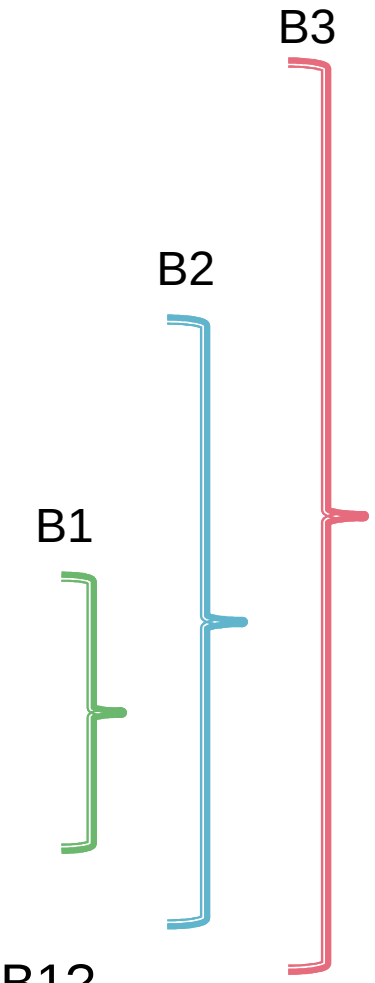
```
{
  int x;  // inicia ámbito de x
  {
    int q; // inicia ámbito de q.
           // x continua siendo visible
  } // finaliza ámbito de q
}   // finaliza ámbito de x
```

Ámbito de variables (II)

```
class MiClase
{
   Variables miembro (atributos)

   public void metodo1(lista de parámetros)
   {
     Variables locales (al método)

     Sentencias compuestas
     {
        Variables locales
     }
   }
}
```

Ámbito de variables (III)

```
class MiClase
{
   int x = 1; // atributo

   public void metodo1(int a)
   {
     int i = 0; // local

     for(i = 0; i < 10; i++)
     {
       int j = 5;
       System.out.println(i*j + a);
     }
   }
}
```



// ¿ “j” puede referenciarse fuera del bloque B1?
// ¿ “x” en B1, en B2? ¿ “i” en B1, B2, B3?

Condicional – if… else

- Permite tomar una decisión para ejecutar una u otra acción, considerando el valor V o F de una expresión

```
if(condición)
  sentencia1;
[else
  sentencia2;]
```

[…] opcional

Condicional – if… else (II)

- Para definir un bloque de instrucciones se usan las llaves { …}

```
if (condición)
{
  // bloque de instrucciones
  // para la condición true
}
[else
{
  // bloque de instrucciones
  // para la condición false
}]
```

[…]* else opcional

Condicional – if… else (III)

- Pueden incluirse sentencias *if* dentro de otras sentencias *if*

```
if (condición)
{
   if (condicion2)
   {
      // instrucciones para condición2
   }
}
[else
{
   // bloque de instrucciones para condición else
}]
```

[…]* else opcional

Condicional – else if

- Se utiliza para evaluar múltiples condiciones y ejecutar aquella que sea verdadera

```
if (condición1)
   sentencia1;
else if (condición2)
   sentencia2;
else if (condición3)
   sentencia3;
…
else
   sentenceN;
```

Condicional – else if (II)

```
int i = 10;


if (i > 100)
  System.out.println(“i más grande que 100”);
else if (i > 50)
  System.out.println(“i más grande que 50”);
else if (i > 25)
   System.out.println(“i más grande que 25”);
else
   System.out.println(“i menor que 25”);
```

switch… case

- Instrucción de decisión múltiple, útil para resolver condiciones muy grandes.
- Se busca el valor de una variable en una lista de constantes (char, byte, short o int). Si se encuentra, se ejecuta el grupo de instrucciones asociadas a la constante.

switch… case (II)

```
switch (expresión)
{
  case valor_1:
          sentencia1;
          sentencia2;
          break;
  case valor_2:
          sentencia1;
          sentencia2;
          break;
  default: // opcional
          sentencia1;
          sentencia2;
}
```

Ejercicio I

- Evaluar una variable entera, cuyo valor representa un **número de mes**.
- Imprimir en pantalla cuántos días tiene el mes indicado por la variable.

Estructuras de repetición (while)

- while
  - Evalúa una expresión y en caso de que sea verdadera repite una serie de instrucciones hasta que la instrucción se evalúe a falsa.

```
while (condición_booleana) {
 instrucción(es);
}
```

Estructuras de repetición

```
public class CicloWhile
{
  public static void main(String[] args)
  {
    int i = 0;

    while (i++ < 5)
    {
      System.out.println(i);
    }
  }
}
```

Estructuras de repetición (for)

- for
  - Se conforma por tres elementos: inicialización, condición e iterador.
  - La variable de control de ciclo puede ser declarada en el inicializador.

```
for (inicializador; condición; iterador) {
 instrucción(es);
}
```

Estructuras de repetición

```
public class CicloFor
{
  public static void main(String[] args)
  {
    int suma = 0;

    for (int i = 0; i <= 10; i++)
    {
       suma += i;
    }
    System.out.println("Suma total: " + suma);
  }
}
```

Ejercicio

- Imprima las tablas de multiplicar del 1 al 5

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
```

Estructuras de repetición (while)

- do… while
  - Ejecuta (al menos una vez) una serie de instrucciones mientras la expresión se evalúe a verdadera.

```
do {
 instrucción(es);
} while (condición)
```

Estructuras de repetición

```
public class CuentaRegresiva
{
  public static void main(String[] args)
  {
    int i = 9;

    do
    {
      System.out.println("Cuenta regresiva:" + i);

      i--;
    } while (i > 0);

  }
}
```

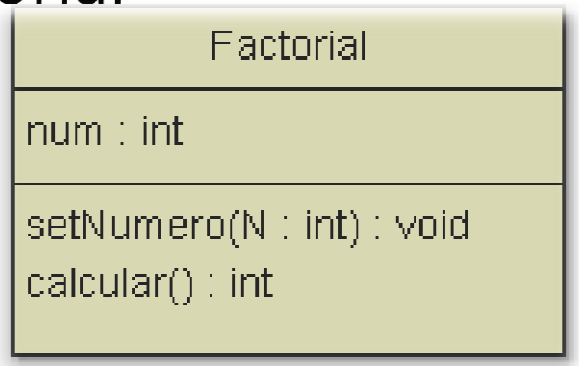
Estructuras de repetición

- Ciclos infinitos

```
while (true)
{
   System.out.println(“ciclo infinito”);
}
```

Ejemplo

- Implementar la clase Factorial que calcule el factorial de un número indicado
  - Desarrollar la clase Factorial
  - Desarrollar la clase PruebaFactorial que demuestre el uso de la clase Factorial



setNumero(int x) // recibe un entero, lo agrega a *num*, y no devuelve nada

calcular() // No recibe nada, devuelve el factorial del número

Ejercicio

- Clase Intervalo. Suma el intervalo cerrado que se proporciona.

sumar() // devuelve el resultado de la suma

setLimites(int x, int y) // recibe dos enteros, el primero corresponde al límite inferior, y el segundo al mayor. No retorna nada