Control Statements (Chapter 5 of Schilit)

Object Oriented Programming BS (CS/SE) II

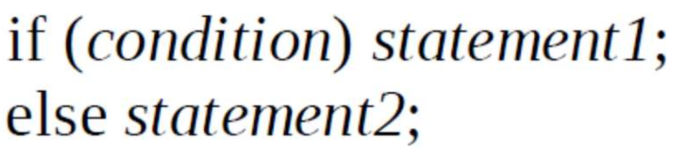
Compiled By:

Abdul Haseeb

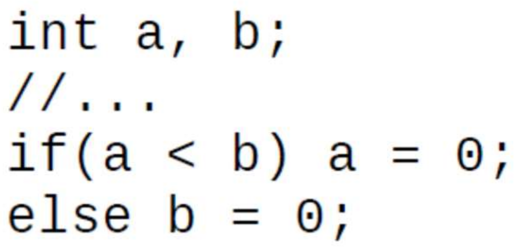
Control Statements

- Change the normal flow of execution
  - Selection Statement:
    - Flow changes based on outcome of an expression
  - Iteration Statement:
    - Repeat one or more statements
  - Jump Statements:
    - Allow you to jump from one section to other

if statement



if (*condition*) *statement1*;
else *statement2*;



```
int a, b;
//...
if(a < b) a = 0;
else b = 0;
```

Write a java program to find maximum between two Numbers

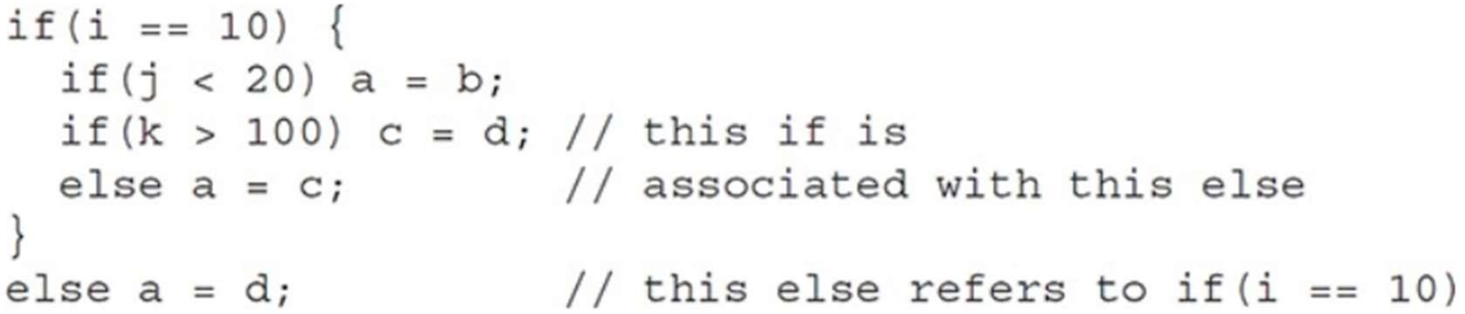
- Sample:
  - Enter first number: 23
  - Enter second number:56
  - Maximum is:56

Now Modify it using if-else, to show Minimum value too

Write a java program to find out whether a number is divisible by 5 or not

- Sample:

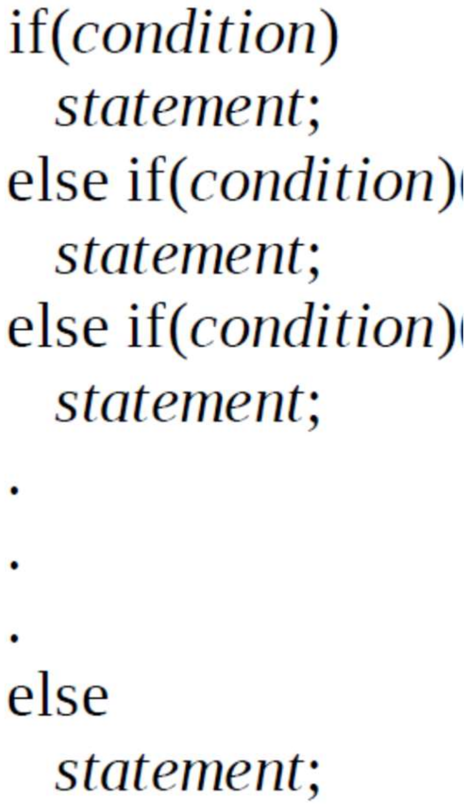
Nested if



```
if(i == 10) {
  if(j < 20) a = b;
  if(k > 100) c = d; // this if is
  else a = c;        // associated with this else
}
else a = d;          // this else refers to if(i == 10)
```

Write a C program using nested if to develop a login program, it should take user name and password as input

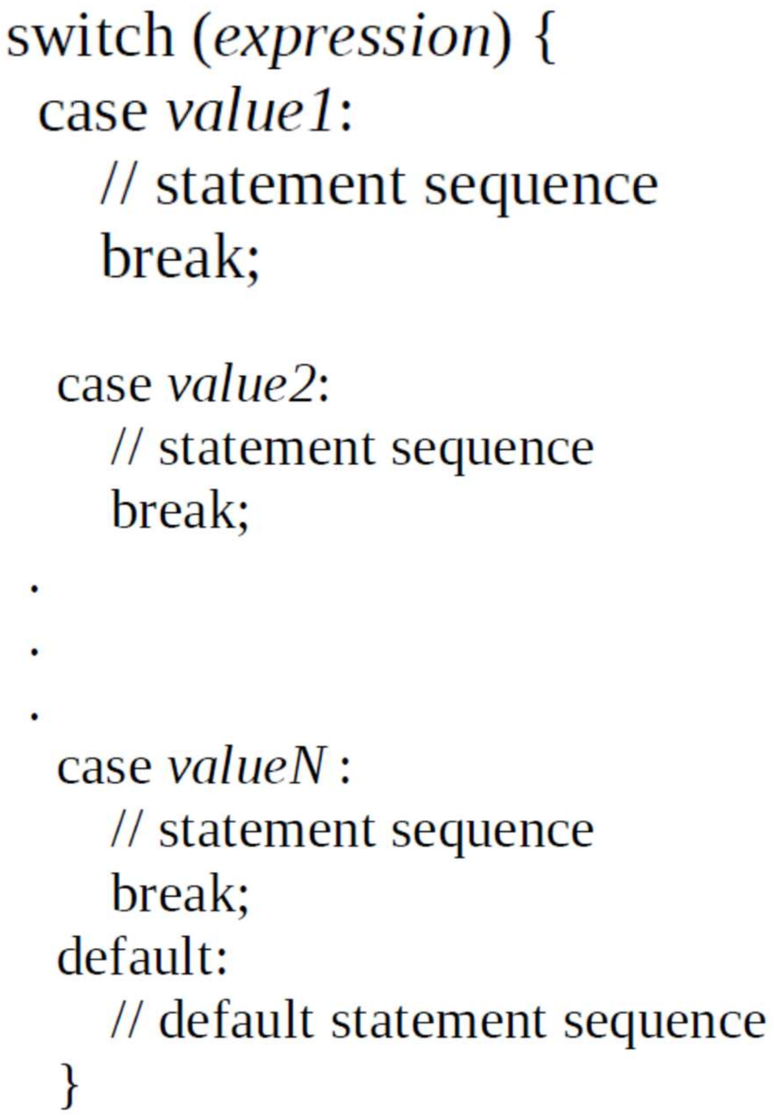
if-else-if



```
if(condition)
   statement;
else if(condition)
   statement;
else if(condition)
   statement;
.
.
.
else
   statement;
```

Demo season according to month

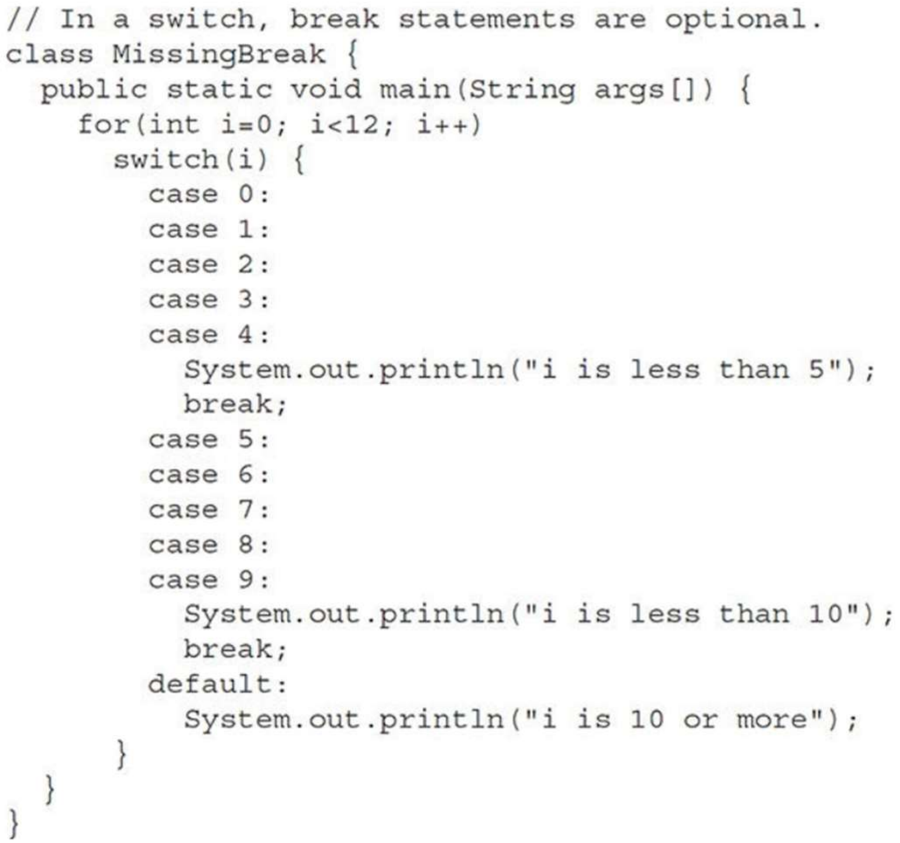
switch statement



```
switch (expression) {
 case value1:
    // statement sequence
    break;

  case value2:
    // statement sequence
    break;
 .
 .
 .
  case valueN :
    // statement sequence
    break;
  default:
    // default statement sequence
  }
```

```java
// A simple example of the switch.
class SampleSwitch {
  public static void main(String args[]) {
    for(int i=0; i<6; i++)
      switch(i) {
        case 0:
          System.out.println("i is zero.");
          break;
        case 1:
          System.out.println("i is one.");
          break;
        case 2:
          System.out.println("i is two.");
          break;
        case 3:
          System.out.println("i is three.");
          break;
        default:
          System.out.println("i is greater than 3.");
      }
  }
}
```

Omitting break from some cases



```
// In a switch, break statements are optional.
class MissingBreak {
  public static void main(String args[]) {
    for(int i=0; i<12; i++)
      switch(i) {
        case 0:
        case 1:
        case 2:
        case 3:
        case 4:
          System.out.println("i is less than 5");
          break;
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
          System.out.println("i is less than 10");
          break;
        default:
          System.out.println("i is 10 or more");
      }
  }
}
```

Modify the seasons program using switch statement

Question

- Which types of data a switch can accept?

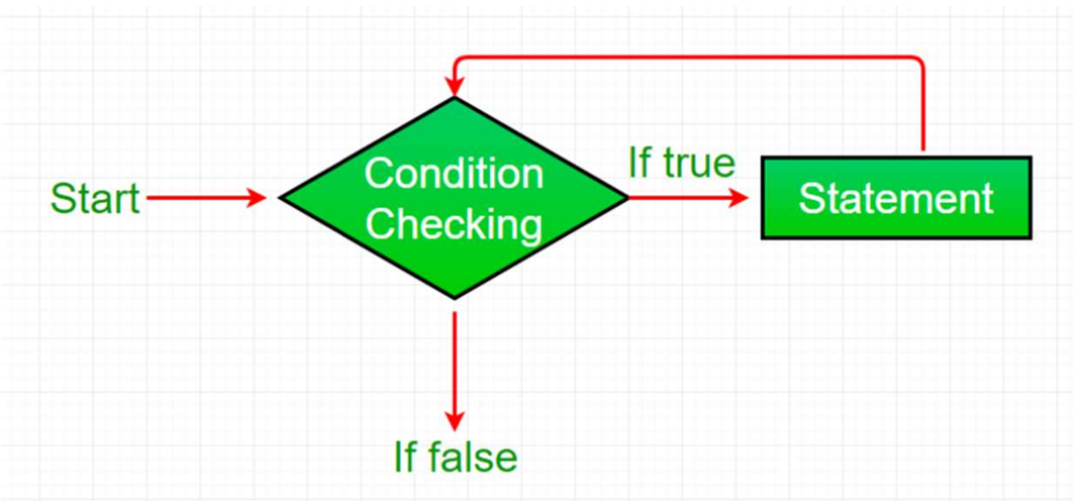
Iteration Statements

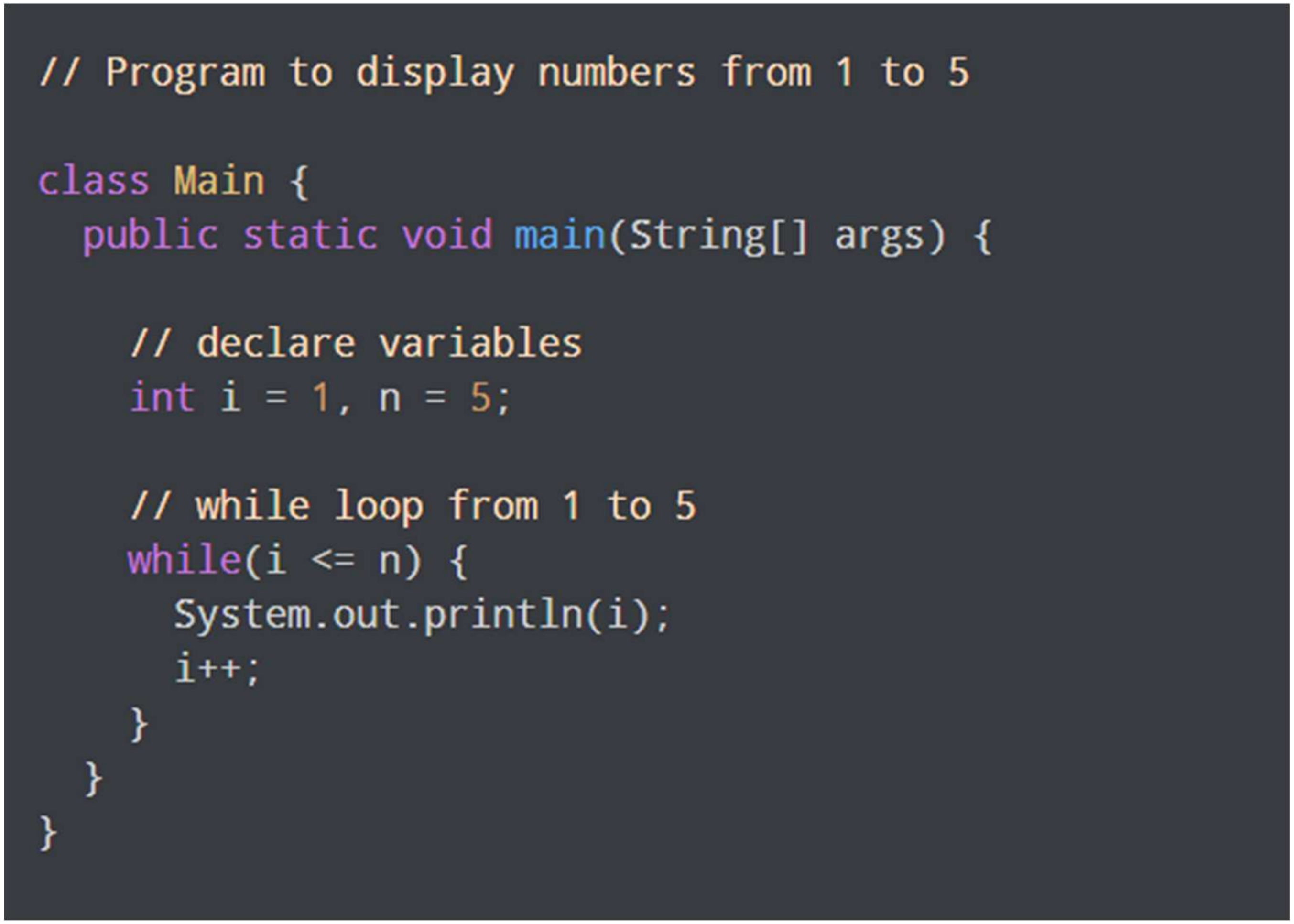
- while, do-while, for

while

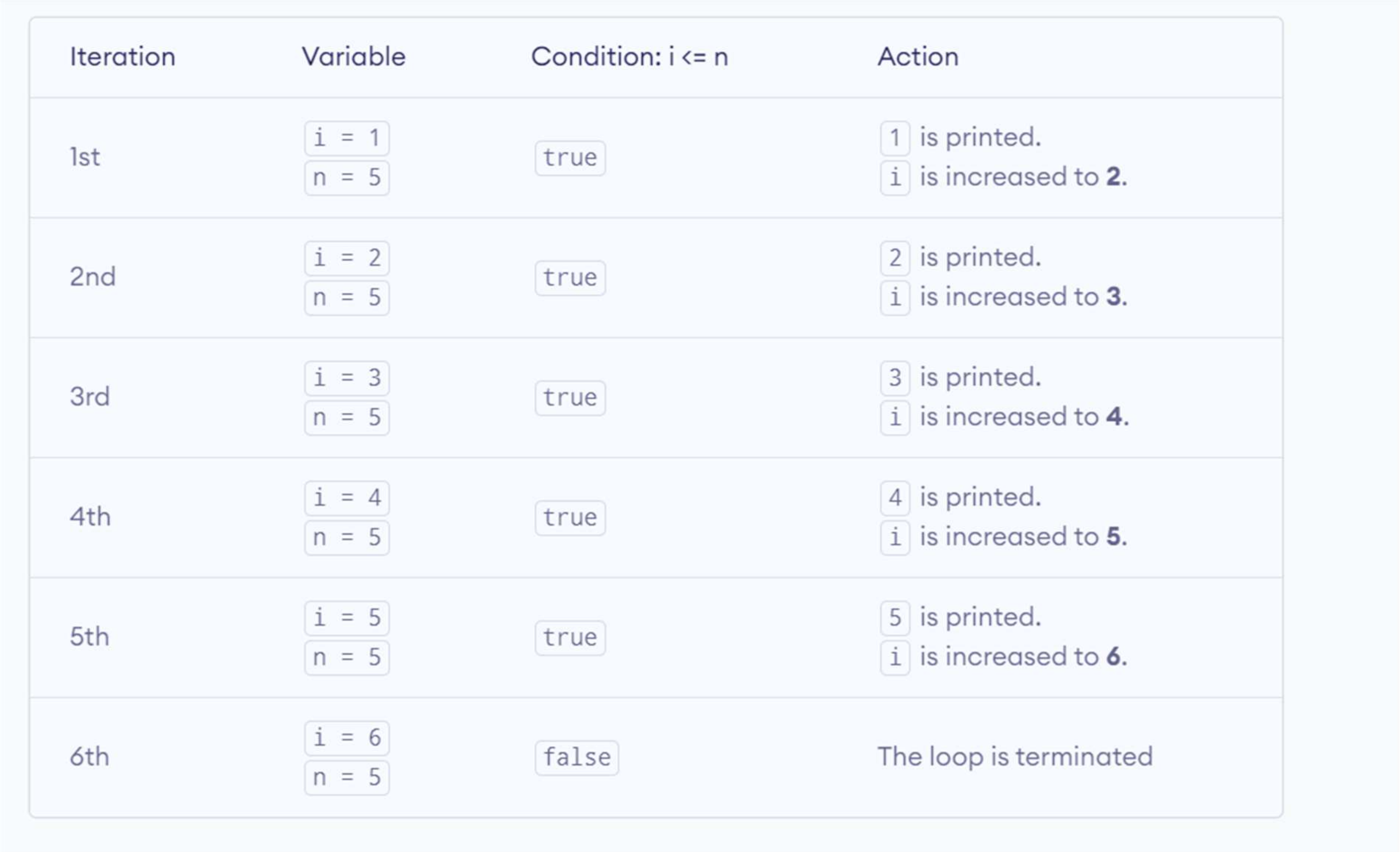
- Used when number of repetitions is unknown
- Demo

Flowchart For while loop (Control Flow):





```
// Program to display numbers from 1 to 5

class Main {
  public static void main(String[] args) {

    // declare variables
    int i = 1, n = 5;

    // while loop from 1 to 5
    while(i <= n) {
      System.out.println(i);
      i++;
    }
  }
}
```



Iteration	Variable	Condition: i <= n	Action

1st	`i = 1`
	`n = 5`
		`true`	`1` is printed.
			`i` is increased to **2.**

2nd	`i = 2`
	`n = 5`
		`true`	`2` is printed.
			`i` is increased to **3.**

3rd	`i = 3`
	`n = 5`
		`true`	`3` is printed.
			`i` is increased to **4.**

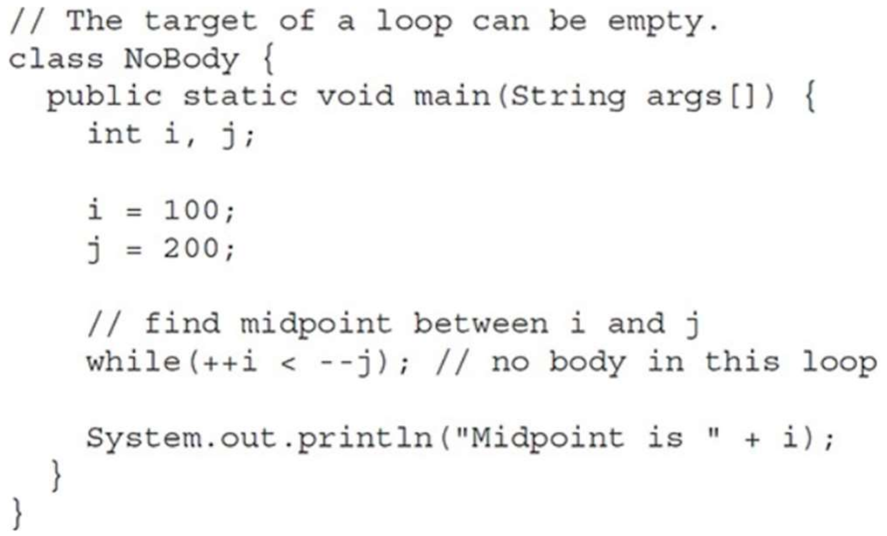
4th	`i = 4`
	`n = 5`
		`true`	`4` is printed.
			`i` is increased to **5.**

5th	`i = 5`
	`n = 5`
		`true`	`5` is printed.
			`i` is increased to **6.**

6th	`i = 6`
	`n = 5`
		`false`	The loop is terminated

while

- while with no body

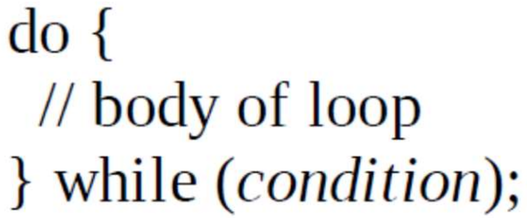


```
// The target of a loop can be empty.
class NoBody {
  public static void main(String args[]) {
    int i, j;

    i = 100;
    j = 200;

    // find midpoint between i and j
    while(++i < --j); // no body in this loop

    System.out.println("Midpoint is " + i);
  }
}
```

Using a while loop take continuous input from user until that input becomes one

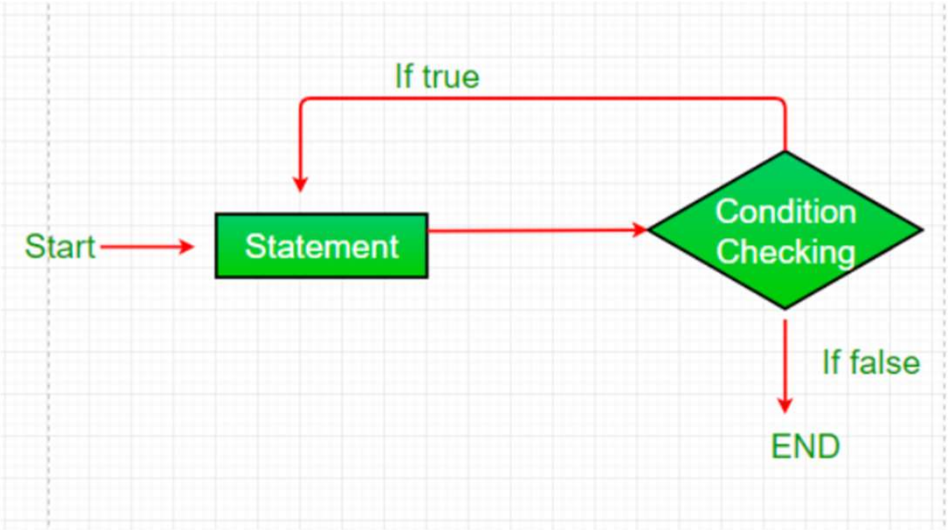
Using a while loop print natural numbers from 1 to 100

do-while



```
do {
  // body of loop
} while (condition);
```

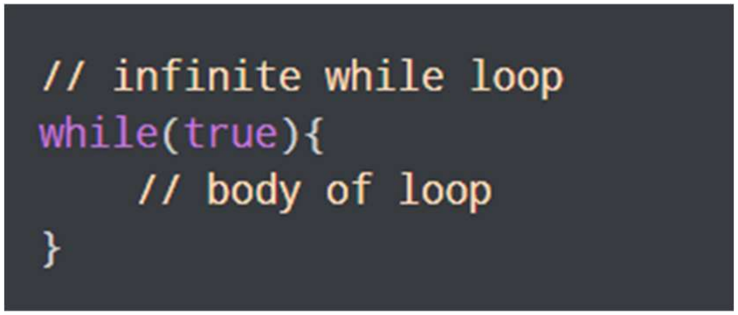
Flowchart do-while loop:



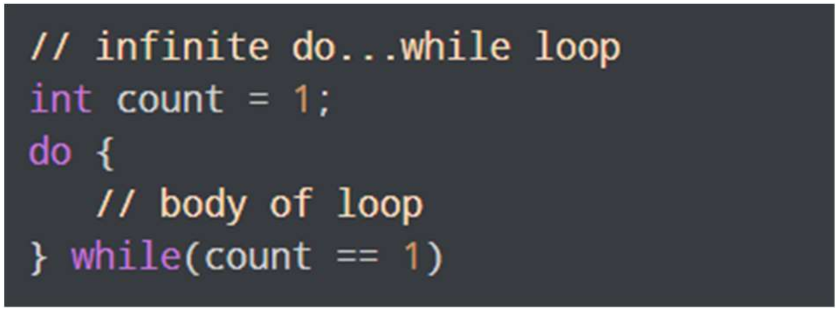
Do sum of first 10 natural numbers using do while Loop

Task

- Take an integer number as input from user, it should be greater than 100000
- Divide that number with 10, until it becomes lesser than 100



```
// infinite while loop
while(true){
    // body of loop
}
```

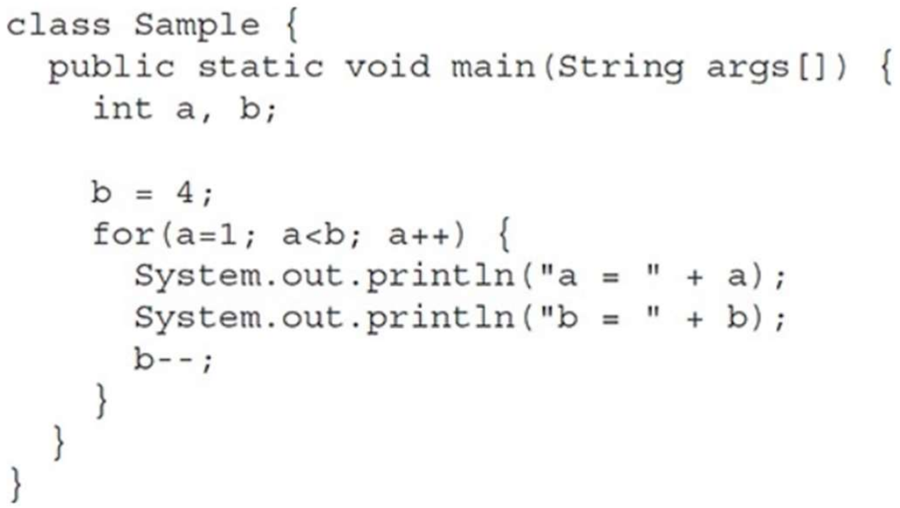


```
// infinite do...while loop
int count = 1;
do {
   // body of loop
} while(count == 1)
```

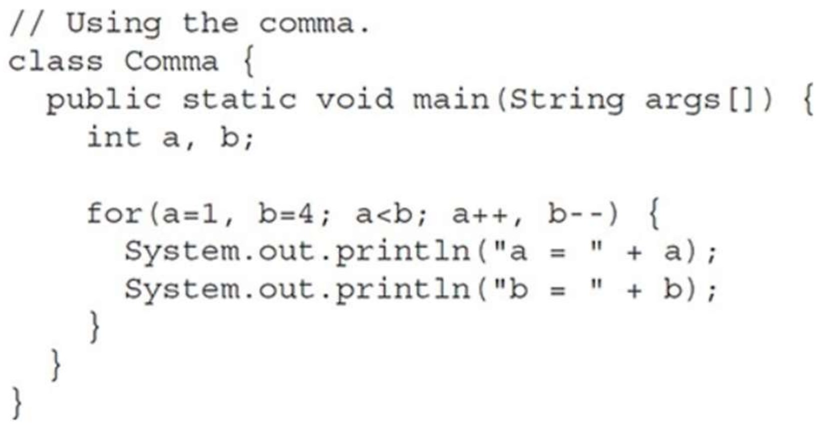
for

- Used when Number of iterations are already known
- Loop variable declared inside/outside loop

For



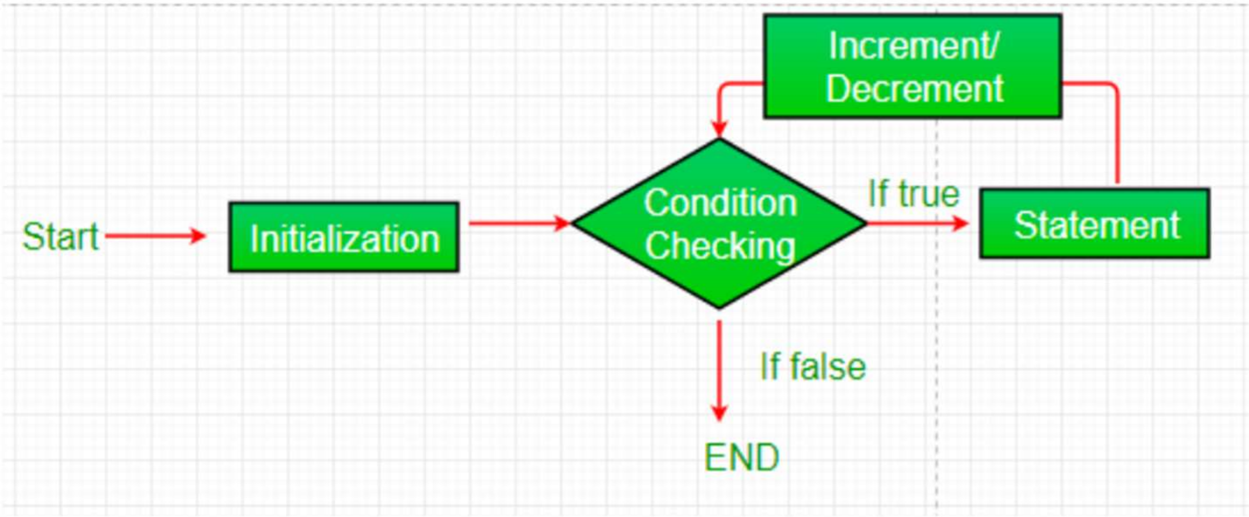
```
class Sample {
  public static void main(String args[]) {
    int a, b;

    b = 4;
    for(a=1; a<b; a++) {
      System.out.println("a = " + a);
      System.out.println("b = " + b);
      b--;
    }
  }
}
```



```
// Using the comma.
class Comma {
  public static void main(String args[]) {
    int a, b;

    for(a=1, b=4; a<b; a++, b--) {
      System.out.println("a = " + a);
      System.out.println("b = " + b);
    }
  }
}
```

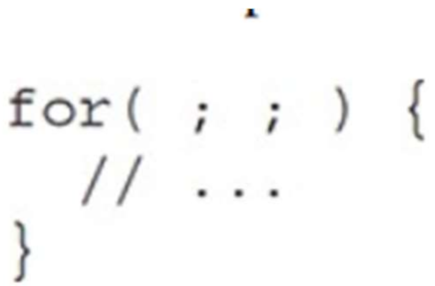
For

Flow chart for loop (For Control Flow):



For

- All three parts of for loop are optional
- An infinite loop

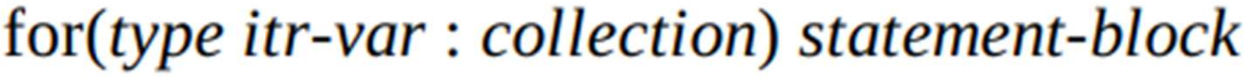


```
for( ; ; ) {
  // ...
}
```

For loop Tasks

- Write a for loop to print first n natural numbers in reverse order
- Write a for loop to print output like this:
    - Line 1
    - Line 2
    - Line 3
    - .
    - .
    - .
    - Line 10

For-each version of for loop



for(*type itr-var* : *collection*) *statement-block*

Early Termination

- Break statement
- Terminate a loop as soon as even number is found in an array

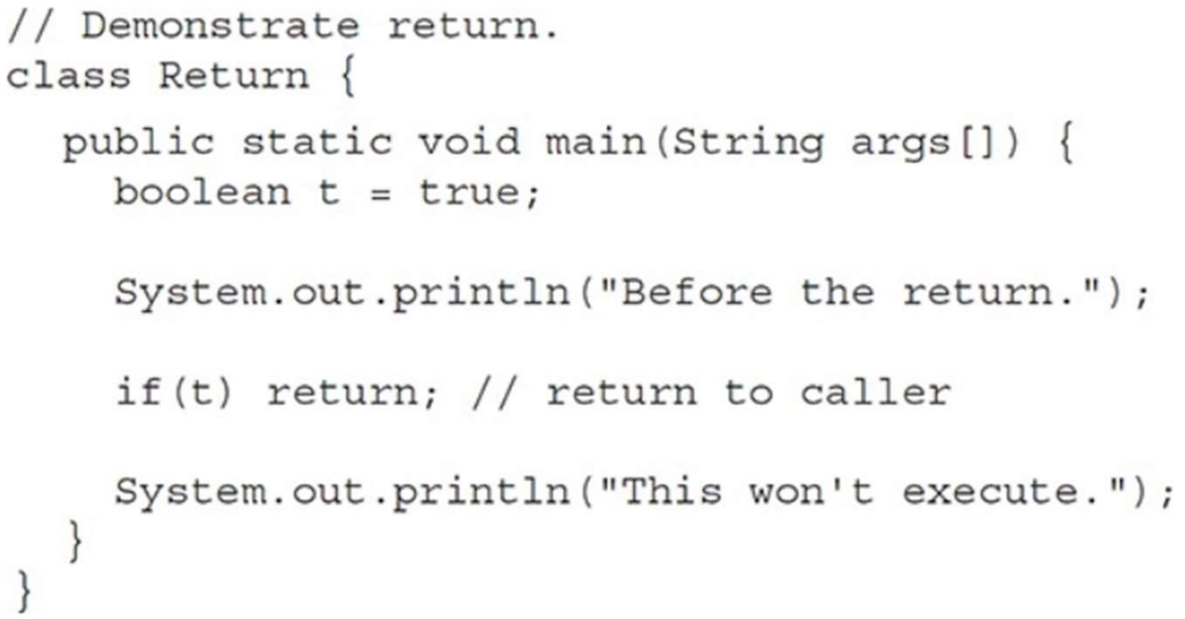
Multidimensional Array Iteration through nested loop

Nested Loops

Jump Statements

- Break:
  - Used to terminate immediate loop
  - Used to terminate a case statement in switch
- Continue:
  - Skips a particular iteration of loop
- Return:
  - Return the control to immediate caller

Return passes control to java run-time system, as Run time system calls the main



```
// Demonstrate return.
class Return {
  public static void main(String args[]) {
    boolean t = true;

    System.out.println("Before the return.");

    if(t) return; // return to caller

    System.out.println("This won't execute.");
  }
}
```

Use of var