MIT AITI

Lecture 4: Control Structures

Kenya 2005

Lecture Outline

- What control structures are
- Different types of control structures:
  - Block Statements
  - Decision Statements
  - Loops

What are Control Structures?

- Without control structures, a computer would evaluate the instructions in a program step-by-step

- Control structures allow you to change:
  - the order in which instructions are evaluated
  - which instructions are evaluated
  - and control the “flow” of the program

- Control structures include:
  - block statements (anything contained within curly brackets)
  - decision statements
  - loops

Decision Statements

If Statements

- The “if” decision statement causes a program to execute a statement conditionally

```
if (expression) {
  statement;
}
next_statement;
```

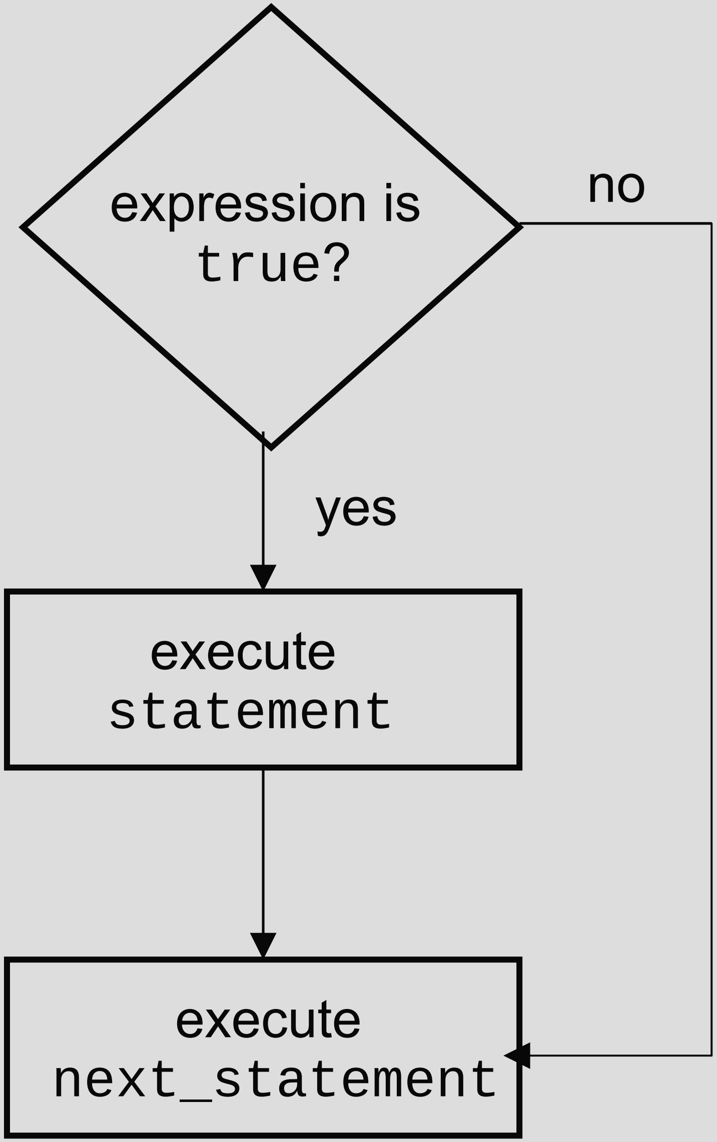
- The `expression` must produce either `true` or `false`, also known as a `boolean` value

- If `expression` returns `true`, `statement` is executed and then `next_statement`

- If `expression` returns `false`, `statement` is not executed and the program continues at `next_statement`

If Statement Flow Chart

```
if (expression) {
   statement;
}
next_statement;
```



If-Else Statements

- The basic “if” statement can be extended by adding the “else” clause in order to do something if expression is false

```
if (expression) {
  statement1;
}
else{
  statement2;
}
next_statement;
```

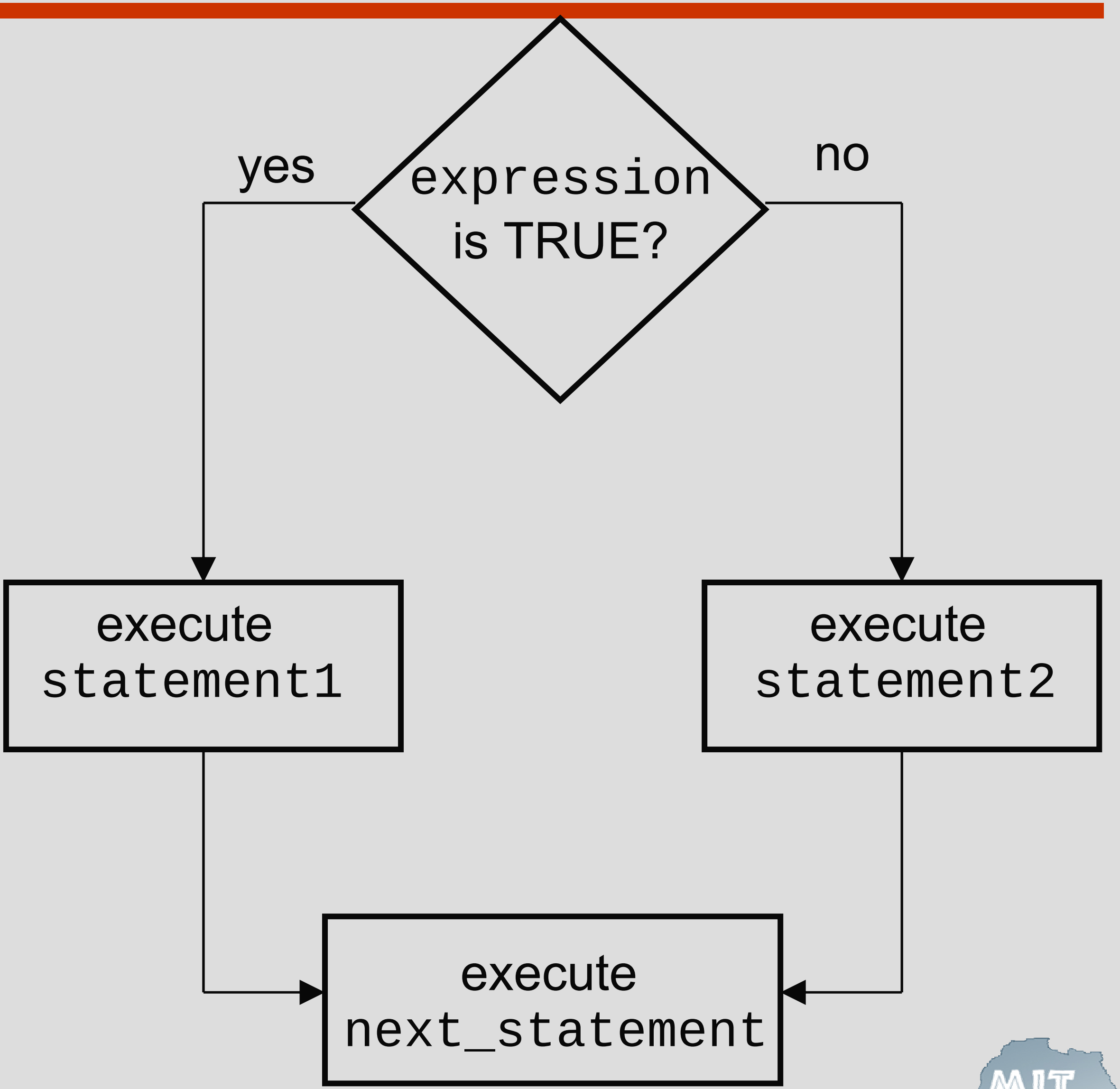
- Again, the `expression` must produce a `boolean` value

- If `expression` returns `true`, `statement1` is executed and then `next_statement` is executed.

- If `expression` returns `false`, `statement2` is executed and then `next_statement` is executed.

If-Else Statement Flow Chart

```
if (expression){
    statement1;
}
else {
    statement2;
}
next_statement;
```



Example of Chained If-Else Statements

- Note that you can combine if-else statements below to make a chain to deal with more than one case

```
if (grade == 'A')
    System.out.println("You got an A.");

else if (grade == 'B')
    System.out.println("You got a B.");

else if (grade == 'C')
    System.out.println("You got a C.");

else
    System.out.println("You got an F.");
```

Switch Statements

- The `switch` statement is another way to test ***several cases*** generated by a given expression.

- The expression must produce a result of type `char`, `byte`, `short` or `int`, but not `long`, `float`, or `double`.

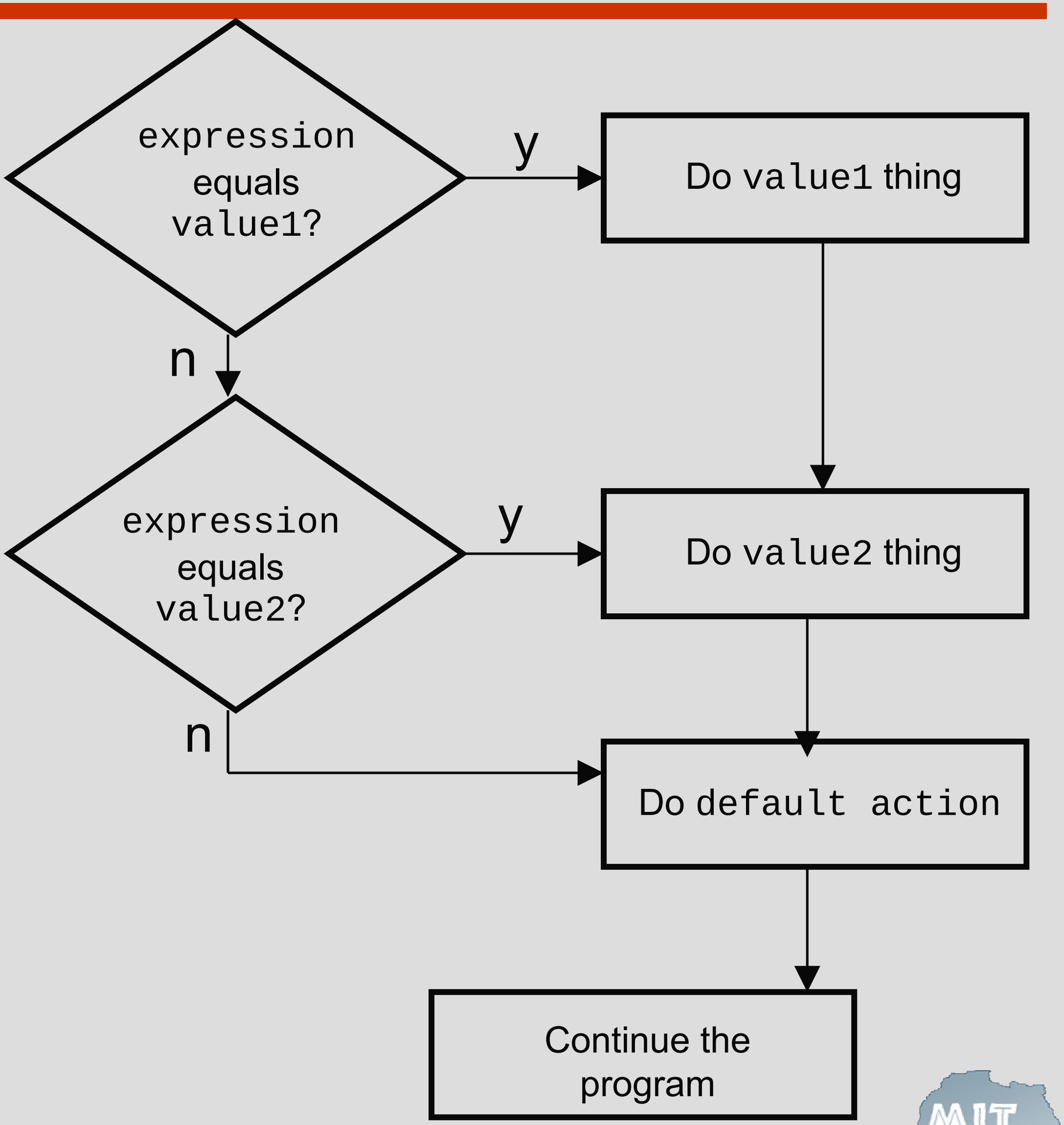
- For example:

```
switch (expression) {

       case value1:
              statement1;

       case value2:
              statement2;

       default:
              default_statement;
}
```

- NOTE: Every statement after the true case is executed

Switch Statement Flow Chart

```
switch (expression){
     case value1:
          // Do value1 thing

     case value2:
          // Do value2 thing

     ...
     default:
          // Do default action
}
// Continue the program
```



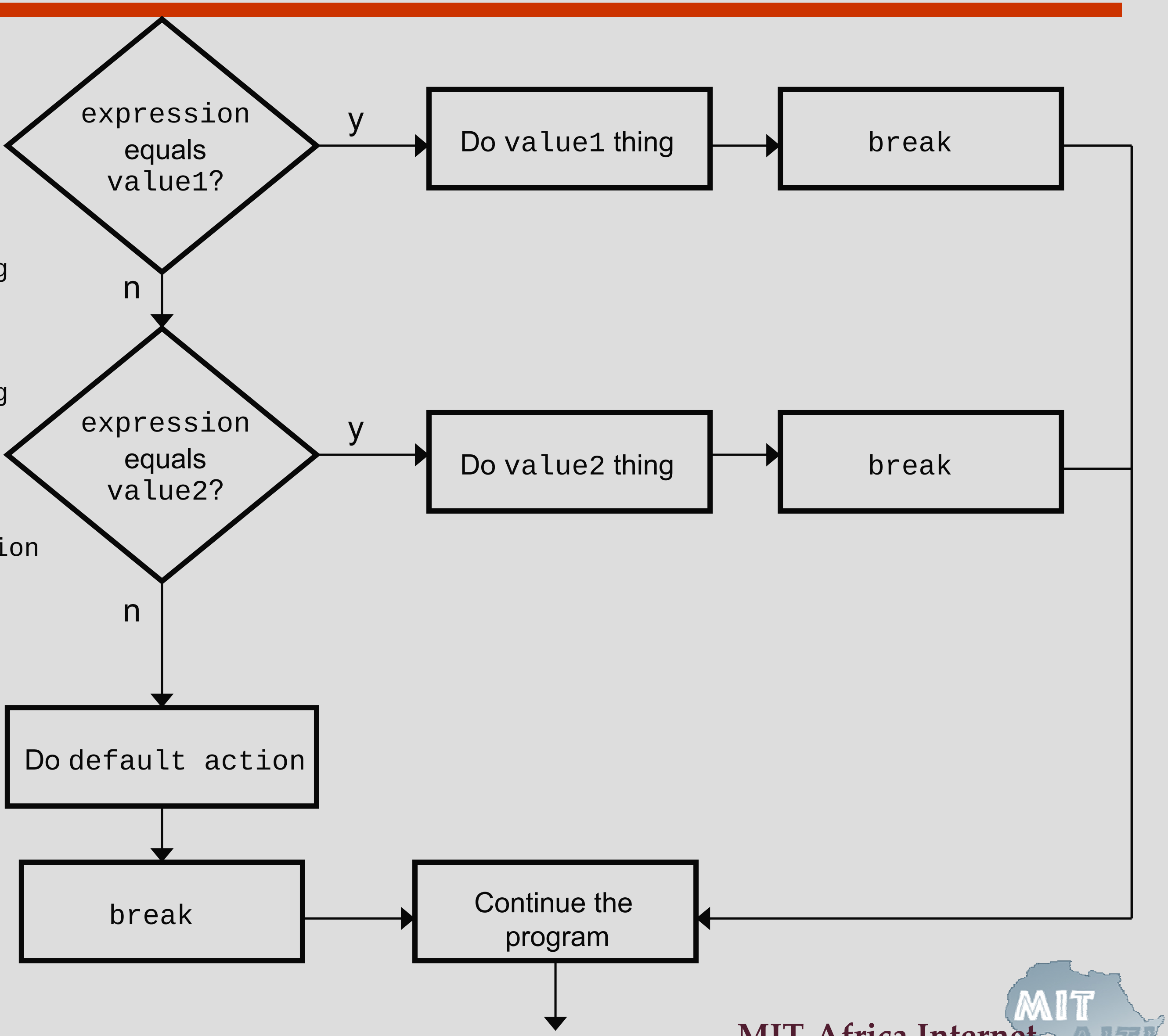
Break Statements in Switch Statements

- The `break;` statement tells the computer to exit the switch statement

- For example:

```
switch (expression) {

        case value1:
                statement1;
                break;

        case value2:
                statement2;
                break;

        default:
                default_statement;
                break;
}
```

```
switch (expression){
    case value1:
           // Do value1 thing
           break;

    case value2:
           // Do value2 thing
           break;

    ...
    default:
           // Do default action
           break;
}
// Continue the program
```



Remember the Example...

- Here is the example of chained if-else statements:

```
if (grade == 'A')
   System.out.println("You got an A.");

else if (grade == 'B')
   System.out.println("You got a B.");

else if (grade == 'C')
   System.out.println("You got a C.");

else
   System.out.println("You got an F.");
```

- Here is the way to convert the chained if-else statement to a switch statement

```
switch (grade) {
      case 'A':
            System.out.println("You got an A.");
            break;
      case 'B':
            System.out.println("You got a B.");
            break;
      case 'C':
            System.out.println("You got a C.");
            break;
      default:
            System.out.println("You got an F.");
}
```

Loops

- A loop allows you to execute a statement or block of statements repeatedly.

- There are three types of loops in Java:

1. `for` loops
2. `while` loops
3. `Do-while` loops (will not discuss in this course)

The For Loop

```
for (initialization_expression;
     loop_condition;
     increment_expression) {
     //statement
}
```

- The control of the `for` loop appear in parentheses and is made up of three parts.

1. The first part, the `initialization_expression`, sets the initial conditions for the loop and is executed before the loop starts.

2. Loop executes so long as the `loop_condition` is true and exits otherwise

3. The third part of the control information, the `increment_expression`, is used to increment the loop counter. This is executed at the end of each loop iteration.

Example

```
   int limit = 5;
   int sum = 0;
   for(int i = 1; i<=limit; i++){

/* initialization_expression
   loop_condition
   increment_expression  */

   // sum = sum + 2;
   sum += 2;
}
```

i = 1	sum = 2
i = 2	sum = 4
i = 3	sum = 6
i = 4	sum = 8
i = 5	sum = 10
i = 6	

- What is the value of `sum` ? **10**

Another Example

```
for(int div = 0; div<1000; div++) {
    if(div % 12 == 0){
       System.out.println(div+"is divisible by 12");
    }
}
```

- This loop will display every number from 0 to 999 that is evenly divisible by 12.

- If there is more than one variable to set up or increment they are separated by a comma.

```
for (i=0, j=0; i*j<1000; i++, j+=2) {

      System.out.println(i+"*"+j+"="+i*j);

}
```

- You do not have to fill every part of the control of the `for` loop but you must still have two semi-colons.

```
for (int i=0; i<100; ) {

      sum+=i;
      i++;
}
```

The `while` Loop

```
while (expression){
      statement
}
```

- This while loop executes as long as the given logical expression between parentheses is `true`. When expression is `false`, execution continues with the statement following the loop block.

- The expression is tested at the beginning of the loop, so if it is initially `false`, the loop will not be executed at all.

Example

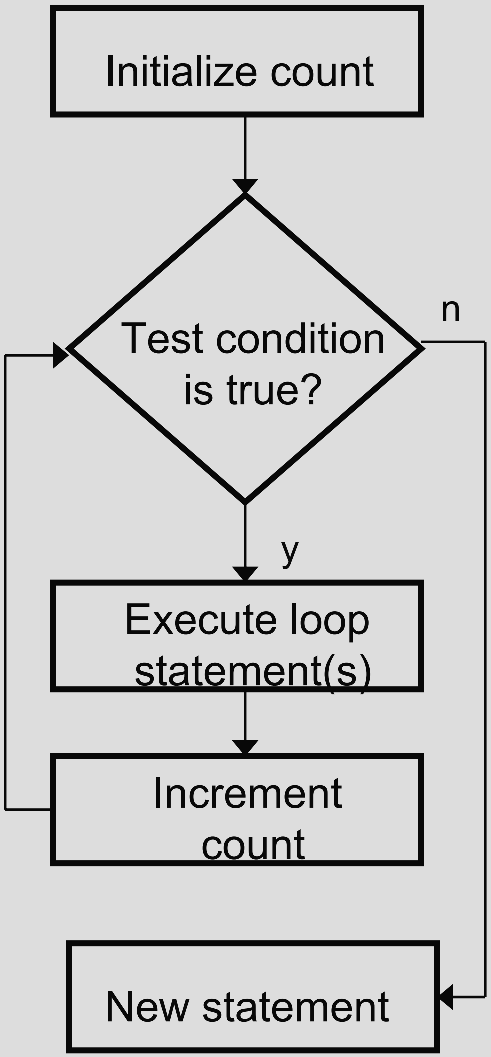
```
int limit = 4;
int sum = 0;
int i = 1;

while (i < limit){
     sum += i;
     i++;
}
```

i = 1	sum = 1
i = 2	sum = 3
i = 3	sum = 6
i = 4	

- What is the value of `sum` ?

6

The `for` loop



Initialize count

Test condition is true?

n

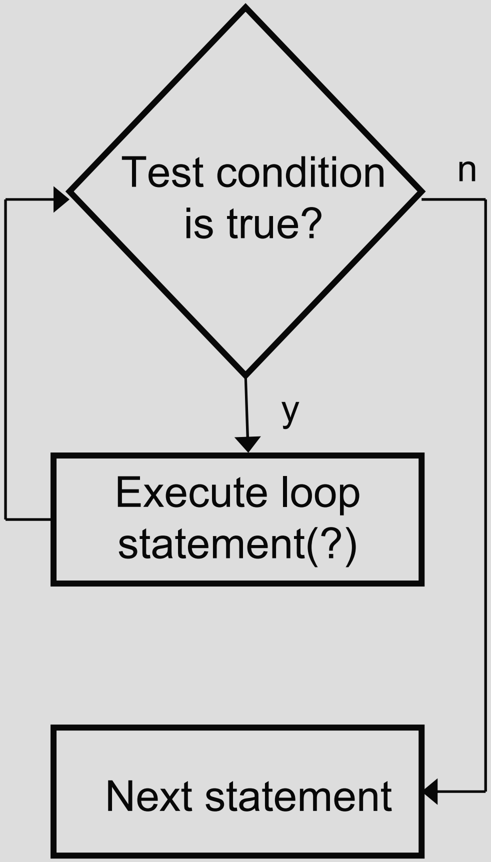
y

Execute loop statement(s)

Increment count

New statement

The `while` loop



Test condition is true?

n

y

Execute loop statement(?)

Next statement

Using the break Statement in Loops

- We have seen the use of the break statement in the switch statement.
- In loops, you can use the break statement to exit the current loop you are in. Here is an example:

```
 int index = 0;
 while (index <= 10){
   index++;
   if (index==3)
       break;
   System.out.println("The index is “
      + index);
}
```

```
index = 1       The index is 1

index = 2       The index is 2

index = 3
```

Nested Loops

- You can *nest* loops of any kind inside another to any depth. Here is a example:

```
int totalCount = 0;
while(totalCount<3){
  for(int i = 0; i < 2; i++){
      totalCount += 1;
  }
}
System.out.println(totalCount);
```

i = 0	totalCount = 1
i = 1	totalCount = 2
i = 0	totalCount = 3
i = 1	totalCount = 4

4

Control Structures Pop Quiz

Question 1 of 3

1. You are withdrawing money from a savings account. How do you use an If Statement to make sure you do not withdraw more than you have.

Use an if statement to check whether the amount you've tried to withdraw is greater than the balance.

Control Structures Pop Quiz

Question 2 of 3

2. How can you implement AbsoluteValue, a function which always returns the positive value of whatever integer it gets as input

If-Else Statement

Switch Statement

Control Structures Pop Quiz

Question 3 of 3

3. What does the following loop do?

```
for (int i=100; i>=0; i--)  {
  System.out.println(i);
}
```

Outputs 100→0 in reverse sequence.

Lecture Summary

- Decision Statements
  - If Statements
  - If-Else Statements
    - Chained If-Else Statements
  - Switch Statements
    - Breaks

- Loops
  - For loops
  - While Loops
  - Nested Loops

EC.S01 Internet Technology in Local and Global Communities
Spring 2005-Summer 2005