Repetition Structures: for, while, do…while

CCS1110
Programming
Principles and Algorithms

Dr Ioanna Stamatopoulou

Lecture Outline

- Repetition Structures (loops)
- for loop
- while loop
  - Counter-controlled repetition
  - Sentinel-controlled repetition
- do...while loop
- More operators: Increment & decrement
- Top-down stepwise refinement

Repetition Structures (loops)

- Repetition Structures (loops) allow the execution of a block of code multiple times
- There are 3 repetition structures in Java:
  - **for**
    - Repeat **for** a particular number of times
  - **while**
    - Repeat **while** a condition is true
    - (condition is checked at the beginning)
  - **do…while**
    - Execute the commands once **(do …)** and then repeat **while** a condition is true
    - (condition is checked at the end)
- All repetition structures are equivalent:
  - A while loop can be written as a for loop, etc.

`for` Repetition Structure

- Used when the number of repetitions is known
- General syntax:

```
for (counter_initialisation; condition; counter_update){
   <body_of_statements>
}
```

- where:
  - **counter_initialisation** initialises a variable that will act as a counter
  - **condition** is a boolean expression that represents the check that is performed before each repetition to determine whether the body of statements will be executed once more
  - **counter_update** is a statement that modifies the counter at the end of each repetition

`for` Example 1:

- Print the numbers 0, 1, …, 5

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

Output on the Screen:

```
0
1
2
3
4
5
```


`for` Example 2:

- Find one power of a number

```
int number = 3;     //the number…
int power = 4;      // …to the power of
int result = 1;     // initial value of the result

for (int i = 0; i < power; i++)
{
        result = result * number;
}
String output = number + " to the power of " + power + " is " + result;
System.out.println(output);
```

Output on the Screen:

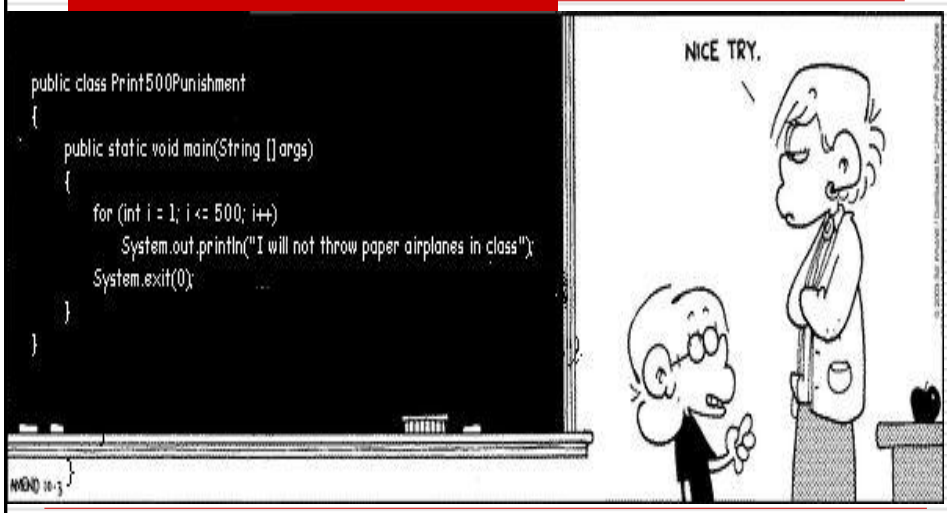
```
3 to the power of 4 is 81;
```

`for` Example 3

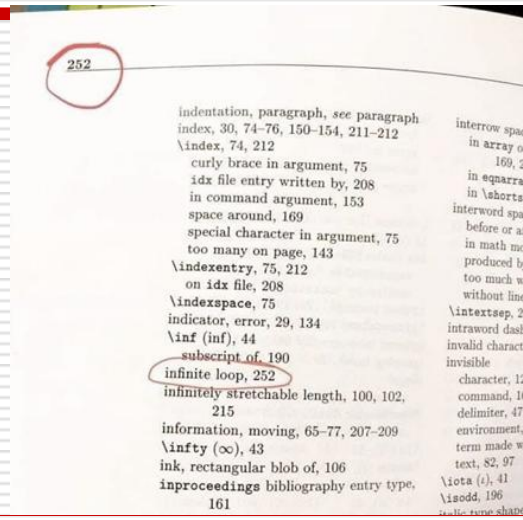
- Extend the previous code so that it finds the first 10 powers of the numbers 1-10

```
int power = 10;      // …to the power of
int result;

for (int number = 1; number <= 10; number++)
{
    result = 1; // initial value of the result for each number

    for (int i = 0; i < power; i++)
    {
        result = result * number;
    }

    String output = number + " to the power of " + power + " is " + result;
    System.out.println(output);
}
```


Repetitions are useful…

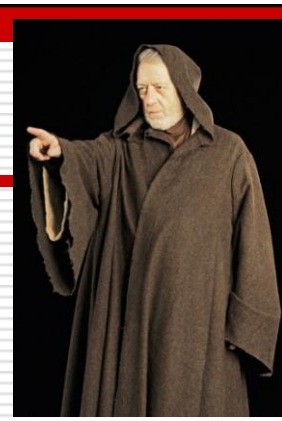


…as long as they terminate at some point….



252

indentation, paragraph, *see* paragraph
index, 30, 74–76, 150–154, 211–212
\index, 74, 212
curly brace in argument, 75
idx file entry written by, 208
in command argument, 153
space around, 169
special character in argument, 75
too many on page, 143
\indexentry, 75, 212
on idx file, 208
\indexspace, 75
indicator, error, 29, 134
\inf (inf), 44
subscript of, 190
infinite loop, 252
infinitely stretchable length, 100, 102, 215
information, moving, 65–77, 207–209
\infty (∞), 43
ink, rectangular blob of, 106
inproceedings bibliography entry type, 161



May the FORs be with you

`while` Repetition Structure

- Used when the number of repetitions is unknown and depends on a condition
- General Syntax:

```
while(condition){
    <body_of_statements>
}

// it will repeat (execute) the loop (body of statements)
// while (as long as) the condition is true
```


`while` Example 1

- Find the first power of 2 that is greater than 1000

```
// LET'S THINK ABOUT THE SOLUTION

// The result is a product that will be calculated incrementally:
// We initialize to 1(neutral element of multiplication)

//while this product is less or equal to 1000

        // we multiply it by 2 (to get the next power of 2)


// at the end we print the result (product)
```

`while` Example 1 (cont'd)

```
// LET'S JAVA !

// The result is a product that will be calculated incrementally:
// We initialize to 1(neutral element of multiplication)
int product = 1 ;

//while this product is less or equal to 1000
while ( product <= 1000 )
{
        // we multiply it by 2 (to get the next power of 2)
        product = 2 * product ;
}

// at the end we print the result (product)
System.out.println("The 1st power of 2 that is greater than 1000
                                                    is: " + product);
```


`while`: sentinel-controlled repetition

- In cases where repetition termination depends on the user, a special value, called the **sentinel** value, must be used
- **Be careful** which value you choose as a sentinel; it must not be a valid value!

`while` Example 2 (sentinel-controlled repetition)

- Find the sum of all the numbers that the user enters
  - Zero is selected as the sentinel value

```
//LET'S THINK ABOUT THE SOLUTION

// we need one variable for the user number and one for the sum (result)

//while the user number is not 0 (sentinel value is 0)
   // add the number to sum
   // prompt again for a number

// at the end print the sum
```


`while` Example 2 (cont'd)

```
//LET'S JAVA

// we need one variable for the user number and one for the sum (result)
int number;
int sum = 0 ;
Scanner scanner = new Scanner(System.in);

// prompt for the 1st number before the loop
// (because it is used in the condition of the while loop that follows)
System.out.println("Enter an integer number (0 to stop)");
number = scanner.nextInt();

//while the user number is not 0 (sentinel value is 0)
while ( number != 0 ) {
   // add the number to sum
   sum += number ;
   // prompt again for a number
   System.out.println("Enter a positive number (0 to stop)");
   number = scanner.nextInt();
}
// at the end print the sum
System.out.println("The sum of all entered numbers is: " + sum);
```

`while` Example 3

Expand the previous code so that it also displays at the end:

- how many numbers have been entered, and
- their average

```
int number;
int sum = 0;
// we need one more variable for the average, and a counter
// (we have to count the numbers to calculate the average)
double average;
int count;

Scanner scanner = new Scanner(System.in);
System.out.print("Enter an integer number (0 to stop)");
number = scanner.nextInt();

while (number != 0) {
   sum += number;
   count++; // increase the counter when a number is added
   System.out.print("Enter an integer number (0 to stop)");
   number = scanner.nextInt();
}
// make sure the user entered at least one number
if (count != 0 ) {
   average = (double) sum / count; // find the average
   System.out.println(count + " numbers entered");
   System.out.println("Sum : " + sum) ;
   System.out.println("Average : " + average) ;
}
else {
   System.out.println("No numbers entered");
}
```


More operators

- Increment operator:
  `sum += number;`
  instead of:
  `sum = sum + number;`
- Also: `-=` , `*=`, `/=`, `%=`
- Increment operator:
  `count++;`
  instead of :
  `count += 1;`
  `count = count + 1;`
- Also: `--`

do…while Repetition Structure

- Used (same as the while loop) when the number of repetitions is not known but…
- Because the check of the condition takes place at the end, the do…while is always executed **at least once**
- General syntax:

```
do {
   <body_of_statements>
}
while (condition);
```

- What does the code on the right do?

```
int number = 1;
int doubleNumber;
do {
   doubleNumber = 2 * number;
   System.out.println(doubleNumber);
   number++;
}
while(number <= 10 );
```


do...while Example

- Find the first power of 2 that is greater than 1000
  - (same exercise as Example 1 of the while loop)

```
int product = 1;

do {
   product = product * 2;
}
while(product <= 1000);

System.out.println(product);
```

Loop Control: `break`

- Used inside a loop in cases when after checking a condition we are certain that there is no reason to execute the following/remaining iterations
- Program execution continues after the end of the loop

`break` Example

[what does the following program do?]

```
public class BreakDemo {
    public static void main(String args[]) {
        String searchMe = "123ioanna22900%^(*$236";
        char searchFor = '9';
        boolean foundIt = false; //boolean flag

        for ( int i = 0; i < searchMe.length(); i++ ) {
            if (searchMe.charAt(i) == searchFor)               {
                foundIt = true;
                //if you find it once, there is no need to continue
                //looking to the rest of the string: BREAK the loop
                break;
            }
        }
        if (foundIt)
            System.out.println(searchFor + " appears in the string");
        else
            System.out.println(searchFor + " does not appear in the string");
    }
}
```

Loop Control: `continue`

- Used inside a loop in cases when after checking a condition we are certain that there is no reason to continue with the execution of the current iteration but want to move to the next iteration

`continue` Example

[what does the following program do?]

```
public class ContinueDemo {
  public static void main(String args[]) {
    String searchMe = "peter piper picked a peck of pickled peppers";
    int numberOfPs = 0;

    for (int i = 0; i < searchMe.length(); i++) {
      //interested only in p's
      if (searchMe.charAt(i) != 'p') {
        //if the letter is not a 'p' there is nothing to do
        //CONTINUE with the next letter
        continue;
      }
      //this is not executed when the letter is not a 'p' because
      //of the continue command
      numberOfPs++;
    }
    System.out.println("Found " + numberOfPs + " p's in the string" + searchMe);
  }
}
```

The previous program without the use of a **continue**

```
public class NonContinueDemo {
  public static void main(String args[]) {
    String searchMe = "peter piper picked a peck of pickled peppers";
    int numberOfPs = 0;

    for (int i = 0; i < searchMe.length(); i++) {
      //interested only in p's
      if (searchMe.charAt(i) == 'p')
        numberOfPs++;
      }
    }
    System.out.println("Found " + numberOfPs + " p's in the string" + searchMe);
  }
}
```


Check list

- What is a sentinel-controlled loop?
- Can 0 always be used as a sentinel value?
- What problems can you identify in the pieces of code to the right?

```
int counter;
int sum = 0 ;

while ( counter <= 1000 ) {
   sum = sum + counter ;
   counter = counter + 1 ;
}
```

```
int counter = 1 ;
int sum = 0;

while ( counter <= 1000 ) {
   sum = sum + counter ;
   counter = counter - 1 ;
}
```

```
int sum;
int counter = 1 ;

while ( counter <= 1000 ) {
   sum = sum + counter ;
   counter = counter + 1 ;
}
```

Check list (cont'd)

- What is the difference between a while and a do…while loop?
- How can I stop the execution of a repetition structure?
- How can I skip part of an iteration and move on to the next one?