

Lecture 46

Java – While Loop

Mostafa Elhosseini

Professor at College of Computer Science and
Engineering

Taibah University

<https://youtube.com/drmelhosseini>

| Intro...

- What to do to process an arbitrary number of grades?
- How can the program determine when to stop the input of grades?

| Intro...

- One way to solve this problem is to **use a special value** called a **sentinel value** (also called a **signal value**, a **dummy value**, or a **flag value**) to indicate “**end of data entry.**”
 - The user types in grades until all legitimate grades have been entered.
 - The user then types the sentinel value to indicate “the last grade has been entered.”
 - Sentinel-controlled repetition is often called indefinite repetition because the number of repetitions isn’t known before the loop begins executing

| while loop

- Allow programs to **repeat without necessarily knowing ahead of time** how many times the loop will run
- **while** repeats the execution **as long as the condition evaluates to True**
- if the condition is False as early as when it is tested for the first time, the body is not executed even once

| while loop

- the body should be able to change the condition's value
- An infinite loop, also called an endless loop, is a sequence of instructions in a program which repeat indefinitely (loop endlessly.)
 - To terminate your program, just press Ctrl-C (or Ctrl-Break on some computers). This will cause the so-called keyboard interrupt

| while loop

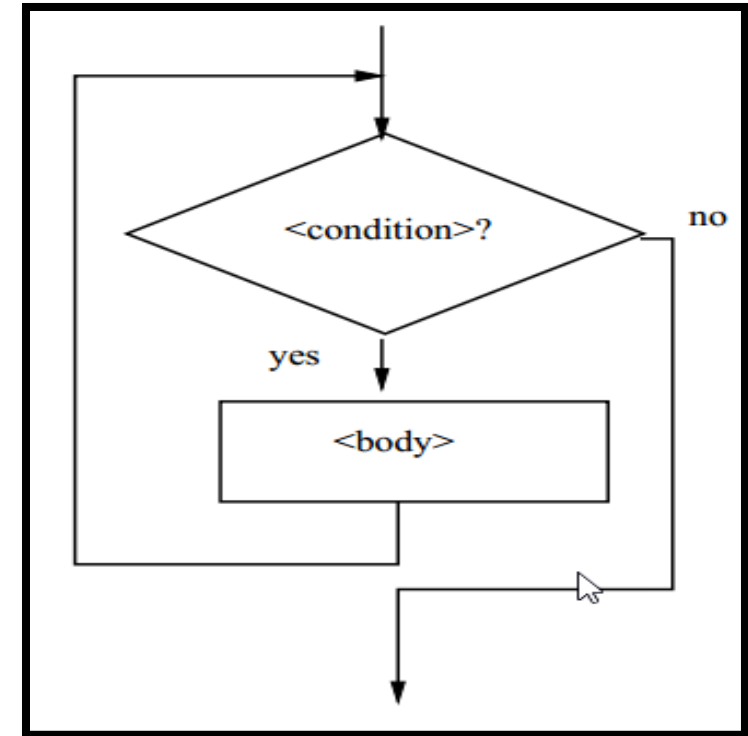
Input the first grade

While the user has not as yet entered the sentinel

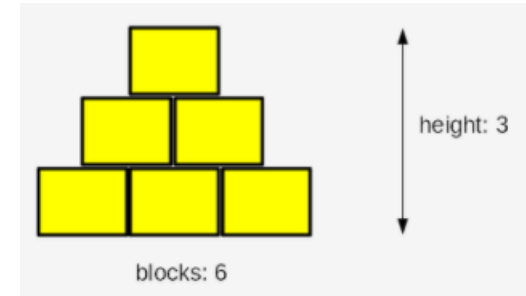
Add this grade into the running total

Add one to the grade counter

Input the next grade (possibly the sentinel)



| Listen to this story:



- A boy and his father, a computer programmer, are playing with wooden blocks. They are building a pyramid.
- Their pyramid is a bit weird, as it is actually a pyramid-shaped wall - it's flat. The pyramid is stacked according to one simple principle: each lower layer contains one block more than the layer above.
- Your task is to write a program which reads the number of blocks the builders have, and outputs the height of the pyramid that can be built using these blocks.
- Note: the height is measured by the number of fully completed layers - if the builders don't have a sufficient number of blocks and cannot complete the next layer, they finish their work immediately.

| Listen to this story:

Scenario

In 1937, a German mathematician named Lothar Collatz formulated an intriguing hypothesis (it still remains unproven) which can be described in the following way:

1. take any non-negative and non-zero integer number and name it `c0` ;
2. if it's even, evaluate a new `c0` as `c0 ÷ 2` ;
3. otherwise, if it's odd, evaluate a new `c0` as `3 × c0 + 1` ;
4. if `c0 ≠ 1` , skip to point 2.

The hypothesis says that regardless of the initial value of `c0` , it will always go to 1.

Of course, it's an extremely complex task to use a computer in order to prove the hypothesis for any natural number (it may even require artificial intelligence), but you can use Python to check some individual numbers. Maybe you'll even find the one which would disprove the hypothesis.

Write a program which reads one natural number and executes the above steps as long as `c0` remains different from 1. We also want you to count the steps needed to achieve the goal. Your code should output all the intermediate values of `c0` , too.

Hint: the most important part of the problem is how to transform Collatz's idea into a `while` loop - this is the key to success.

Test your code using the data we've provided.

| do – while loop

- A characteristic of the while loop is that if the **test condition is initially false, the loop never executes**. For example, the while loop in the following code fragment will not execute the statement body because the variable x evaluates to 0 before the first iteration

```
int x = 0;
```

```
...
```

```
while (x != 0)
```

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

| do – while loop

- In the do-while loop, the **test expression is evaluated after the loop executes**. This ensures that the **loop executes at least once**, as in the following example:

```
int x = 0;
```

```
...
```

```
do
```

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

```
while (x != 0);
```

| do – while loop

```
loopVar = 0; ← External initialization
do ← Start of loop
{
    // Processing statements
    loopVar++; ← Loop variable update
}
while(loopVar != 10); ← Loop continuation test
```

| Break - Continue

- The **break** statement **gets you out of loop as soon as it's executed**. It is often used when an **unexpected condition occurs** — one that the loop test condition is not looking for
 - exits the loop immediately, and unconditionally ends the loop's operation; the program begins to execute the nearest instruction after the loop's body
- The **continue** statement is inserted in the body of the loop, and **when executed, takes you back to the beginning of the loop — bypassing** any statements not yet executed
 - behaves as if the program has suddenly reached the end of the body; the next turn is started and the condition expression is tested immediately

| Break

- The **break** statement, when executed in a while, for, do/while or switch structure, **causes immediate exit from that structure.**
- Execution **continues with the first statement after the structure.** Common uses of the break statement are to escape early from a loop or skip the remainder of a switch structure.

| Continue

- The **continue** statement, when executed in a while, for or do/while structure, **skips the remaining statements in the loop** body and **proceeds with the next iteration** of the loop.

| References

- Java: How to Program, 9th Edition (Deitel) 9th Edition by Paul Deitel , Harvey M. Deitel
- Head First Java, 2nd Edition 2nd Edition by Kathy Sierra, Bert Bates
- Think Java: How to Think Like a Computer Scientist 1st Edition by Allen B. Downey, Chris Mayfield