Looping

CSE 142, Summer 2002

Computer Programming 1

http://www.cs.washington.edu/education/courses/142/02su/

Readings and References

- Reading
  - » Chapter 12, *An Introduction to Programming and Object Oriented Design using Java*, by Niño and Hosch
  - » Chapter 15, *Introduction to Programming in Java*, Dugan
- Other References

What is a loop?

- Loop - some definitions from dictionary.com
  - » Something having a shape, order, or path of motion that is circular or curved over on itself.
  - » A segment of film or magnetic tape whose ends are joined, making a strip that can be continuously replayed.
  - » Computer Science. A sequence of instructions that repeats either a specified number of times or until a particular condition is met.

Why do we want loops in our code?

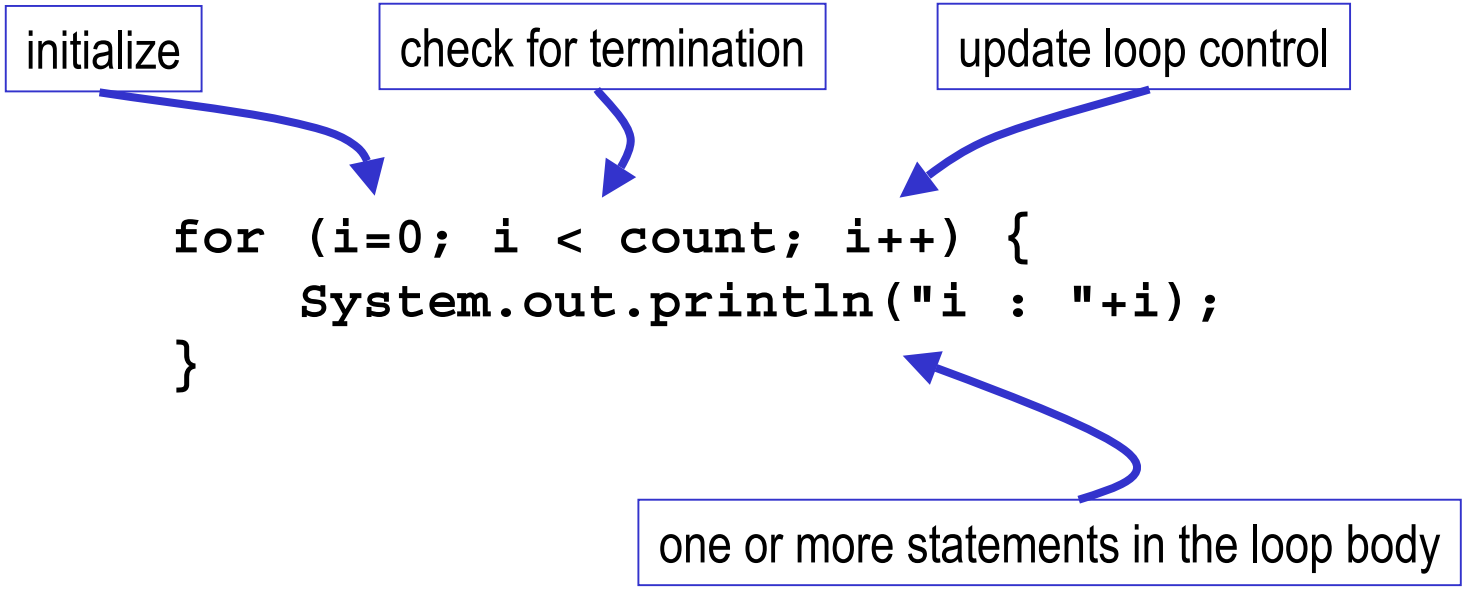
- Do something for a given number of times or for every object in a collection of objects
  - » for every animal in the list, provide it with a meal
  - » for every shape in the blob, move the shape
  - » find the classroom with the most seats
  - » calculate the average meal size for all the pets
  - » make a list of all movies that Kevin Bacon has appeared in with Harrison Ford
- Termination of some loops is based on a count

Why do we want loops in our code?

- Keep doing something until we arrive at a termination condition
  - » read until the end of an input command file
  - » search the disk until we find a requested file
  - » read packets from the network until all information for a web page has been read in
  - » remove items from a request queue and process them until the queue is empty
- Termination of some loops is based on a condition

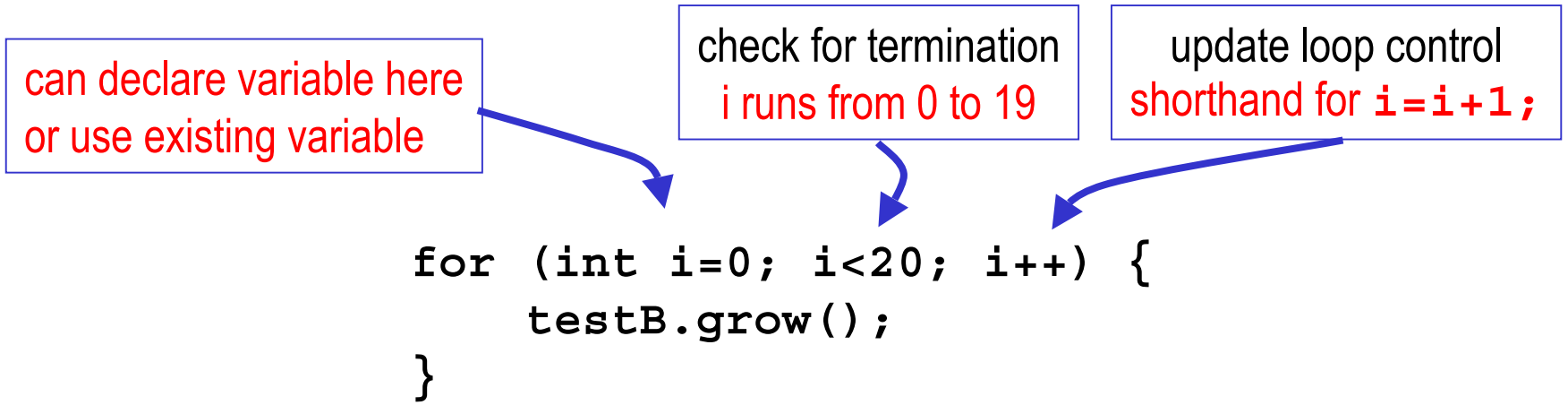
The `for` loop

- A counting loop is usually implemented with `for`
  - » The `for` statement is defined in section 14.13 of the Java Language Specification



`for` example

- a counting loop implemented with **`for`**



Looper.java

limited life of a loop control variable

- The scope of a local variable declared in the ForInit part of a for statement includes all of the following:
  - » Its own initializer
  - » Any further declarators to the right in the ForInit part of the for statement
  - » The Expression and ForUpdate parts of the for statement
  - » The contained Statement

from Java Language Specification, section 6.3

some shortcuts

- **`i++`**
  - » **`theAnimal = pets.get(i++);`**
  - » get the value of i for use in the call to get(int), then increment i and store the incremented value
  - » This is known as post-increment
- **`++i`**
  - » **`theAnimal = pets.get(++i);`**
  - » get the value of i, increment it, set a copy aside for the call to get(int) and store incremented value in i
  - » This is known as pre-increment

compound assignment operators

- can shorten statements like this
  - » from this: `a = a + b;`
  - » to this: `a += b;`

- Any time the left hand side is repeated on the right hand side as a simple operand you can use a compound assignment operator

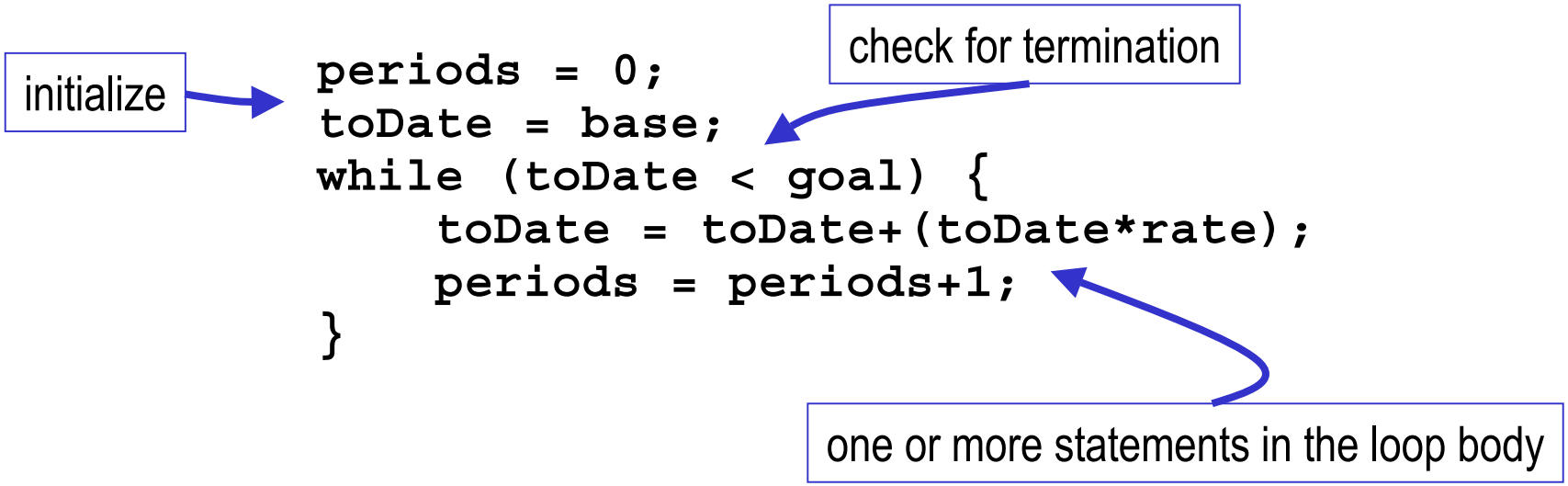
```
step = step / 2;      ⇔     step /= 2;
area = area * factor  ⇔     area *= factor;
```

Why do we want loops in our code?

- Keep doing something until we arrive at a termination condition
  - » read until the end of an input command file
  - » search the disk until we find a requested file
  - » read packets from the network until all information for a web page has been read in
  - » remove items from a request queue and process them until the queue is empty
- Termination of some loops is based on a condition

The `while` loop

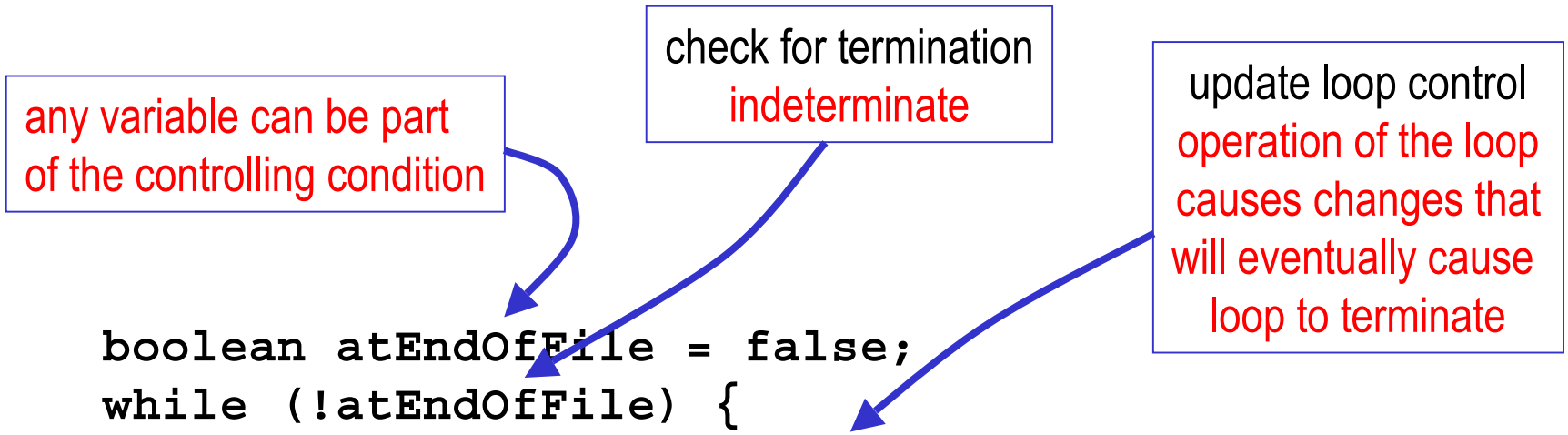
- condition loop is usually implemented with **`while`**
  - » The **`while`** statement is defined in section 14.11 of the Java Language Specification



Note: reaching a limit by counting is satisfying a condition.
`for` loops can be rewritten as **`while`** loops, and vice versa

`while` example

- a condition loop implemented with **`while`**




```
boolean atEndOfFile = false;
while (!atEndOfFile) {
    read another line and set atEndOfFile if appropriate
    process the new line if needed
}
```

`Looper.java`

body of loop may not execute at all

- Notice that depending on the values of the control variables, it is quite possible that the body of the loop will not execute at all in both **`for`** and **`while`**

```
goal = 75;
...
periods = 0;
toDate = 100;
while (toDate < goal) {
    toDate += toDate*rate;
    periods++;
}
```

check for termination
`toDate` is already greater than **`goal`**,
and so the entire loop is skipped