CIS 110: Introduction to Computer Programming

Lecture 5

The Loop-the-Loop

(§ 2.3-2.4)

Outline

1. For-loops!
2. Algorithm Design and Pseudocode

Announcements

- Date of the final is tentatively:

MONDAY, DECEMBER 19th, 6-8 PM

  - If you have a conflict, please let me know ASAP.
- Need more practice? Try Practice-it!
  - Web-based tool where you can work on practice problems that are automatically checked online.
  - Linked off of the course webpage.

For Loops

Redundancy in Patterns

- Problem: write a program that prints out successive cubes.

Output:

```
0^3 = 0
1^3 = 1
2^3 = 8
3^3 = 27
4^3 = 64
```

A Solution with Our Current Tools

```
public class Cubes {
  public static void main(String[] args) {
    System.out.println(“0^3 = “ + 0 * 0 * 0);
    System.out.println(“1^3 = “ + 1 * 1 * 1);
    System.out.println(“2^3 = “ + 2 * 2 * 2);
    System.out.println(“3^3 = “ + 3 * 3 * 3);
    System.out.println(“4^3 = “ + 4 * 4 * 4);
  }
}
```

- Pretty obvious repetition, but we have no way of dealing with it…

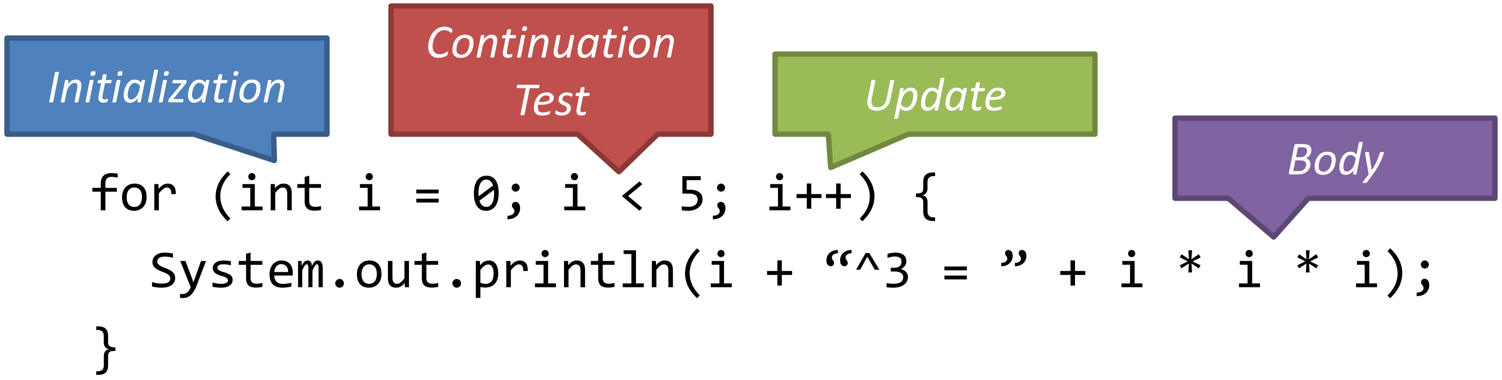
Our First For-loop

```
public class Cubes {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      System.out.println(i + “^3 = ” + i * i * i);
    }
  }
}
```

- The *for loop* construct allows us to express repetitive patterns in a structured way.

The For-loop

- The For-loop is a *control statement*.
  - Doesn’t do anything on its own, but instead *controls* the execution of other statements.



Initialization

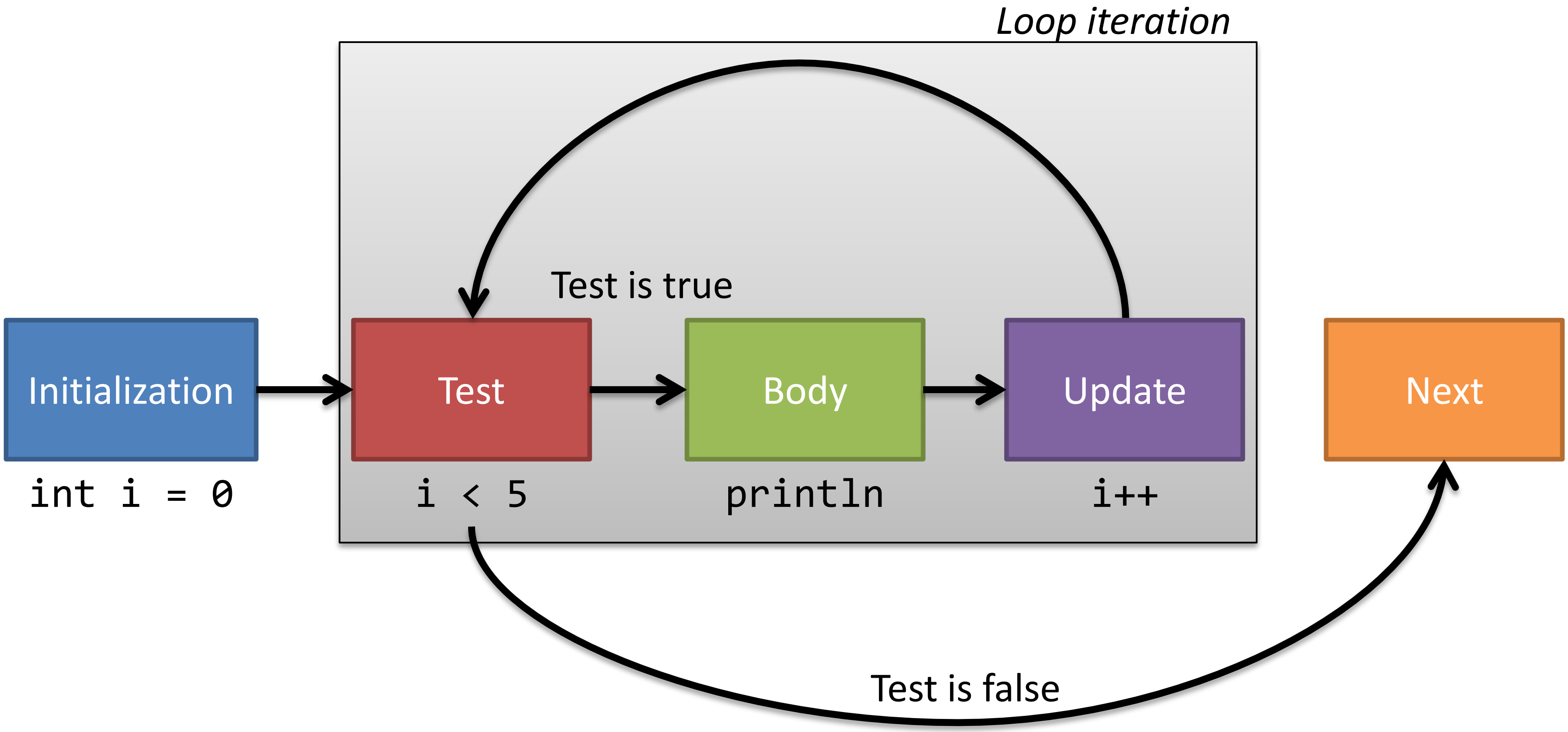
Continuation Test

Update

Body

```
for (int i = 0; i < 5; i++) {
  System.out.println(i + “^3 = ” + i * i * i);
}
```

For-loop Semantics



Our Example, Step-by-step (1)

```
public class Cubes {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      System.out.println(i + “^3 = ” + i * i * i);
    }
  }
}
```

Iteration #	Value of *i*	Test	Output

Our Example, Step-by-step (2)

```
public class Cubes {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      System.out.println(i + “^3 = ” + i * i * i);
    }
  }
}
```

Iteration #	Value of *i*	Test	Output
1	0	0 < 5 is *true*	0^3 = 0

Our Example, Step-by-step (3)

```
public class Cubes {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      System.out.println(i + “^3 = ” + i * i * i);
    }
  }
}
```

Iteration #	Value of *i*	Test	Output
1	0	0 < 5 is *true*	0^3 = 0
2	1	1 < 5 is *true*	1^3 = 1

Our Example, Step-by-step (4)

```
public class Cubes {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      System.out.println(i + “^3 = ” + i * i * i);
    }
  }
}
```

Iteration #	Value of *i*	Test	Output
1	0	0 < 5 is *true*	0^3 = 0
2	1	1 < 5 is *true*	1^3 = 1
3	2	2 < 5 is *true*	2^3 = 8

Our Example, Step-by-step (5)

```
public class Cubes {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      System.out.println(i + “^3 = ” + i * i * i);
    }
  }
}
```

Iteration #	Value of *i*	Test	Output
1	0	0 < 5 is *true*	0^3 = 0
2	1	1 < 5 is *true*	1^3 = 1
3	2	2 < 5 is *true*	2^3 = 8
4	3	3 < 5 is *true*	3^3 = 27

Our Example, Step-by-step (6)

```
public class Cubes {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      System.out.println(i + “^3 = ” + i * i * i);
    }
  }
}
```

Iteration #	Value of *i*	Test	Output
1	0	0 < 5 is *true*	0^3 = 0
2	1	1 < 5 is *true*	1^3 = 1
3	2	2 < 5 is *true*	2^3 = 8
4	3	3 < 5 is *true*	3^3 = 27
5	4	4 < 5 is *true*	4^3 = 64

Our Example, Step-by-step (7)

```
public class Cubes {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      System.out.println(i + “^3 = ” + i * i * i);
    }
  }
}
```

Iteration #	Value of *i*	Test	Output
1	0	0 < 5 is *true*	0^3 = 0
2	1	1 < 5 is *true*	1^3 = 1
3	2	2 < 5 is *true*	2^3 = 8
4	3	3 < 5 is *true*	3^3 = 27
5	4	4 < 5 is *true*	4^3 = 64
6	5	5 < 5 is *false*	

For-loop Bounds

- Different sorts of bounds are possible!

```
for (int i = 1; i <= 5; i++) {
  System.out.print(i + “ “);
}
```

```
Output: 1 2 3 4 5
```

```
for (int i = 5; i > 0; i--) {
  System.out.print(i + “ “);
}
```

```
Output: 5 4 3 2 1
```

```
for (int i = 256; i > 0; i /= 2) {
  System.out.print(i + “ “);
}
```

```
Output: 256 128 64 32 16 8 4 2 1
```

```
for (int i = -3; i < 10; i += 2) {
  System.out.print(i + “ “);
}
```

```
Output: -3 -1 1 3 5 7 9
```

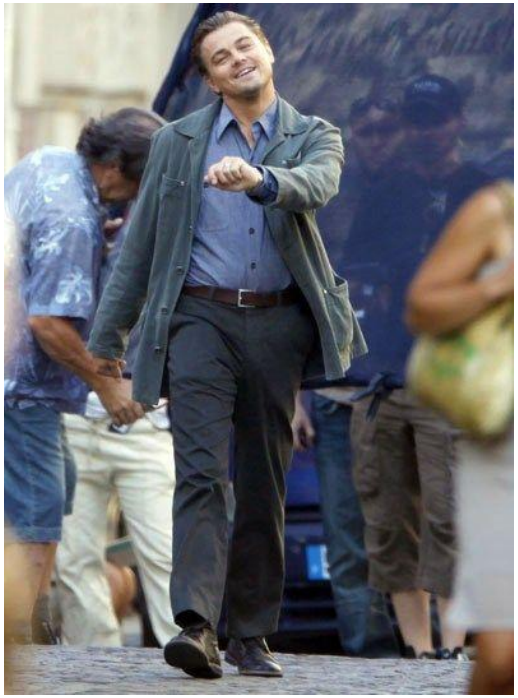
"<" versus "<=" Bounds

- "i = 0; i < 5" versus "i = 1; i <= 5":
  - Both give 5 loop iterations.
  - "i = 0, 1, 2, 3, 4" versus "i = 1, 2, 3, 4, 5".
- "i = 0; i < 5" is *less intuitive, more canonical*
  - Most computations are naturally *zero-based*.
  - For homework 2, either is fine.
    - Try to get used to the zero-based style.

Yo Dawg, I Heard You Liked Loops

- Problem: draw a rectangle of stars.

```
*****
*****
*****
*****
*****
```



Solution: Loopception

```
public class Rectangle {
  public static void main(String[] args) {
    // The outer for-loop controls the #/lines
    for (int i = 0; i < 5; i++) {
      // The inner for-loop controls the
      // contents of a single line.
      for (int j = 0; j < 5; j++) {
        System.out.print(“*”);
      }
      System.out.println();
    }
  }
}
```

Careful with Your Nested Loop Bounds!

```
for (int i = 0; i < 5; i++) {
  for (int j = 0; j < 5; j++) {
    System.out.print(“*”);
  }
  System.out.println();
}
```

```
*****
*****
*****
*****
*****
```

```
for (int i = 0; i < 5; i++) {
  for (int j = 0; j < i; j++) {
    System.out.print(“*”);
  }
  System.out.println();
}
```

```

*
**
***
****
```

```
for (int i = 0; i < 5; i++) {
  for (int j = 0; j < 5; i++) {
    System.out.print(“*”);
  }
  System.out.println();
}
```

```
Infinite loop!
```

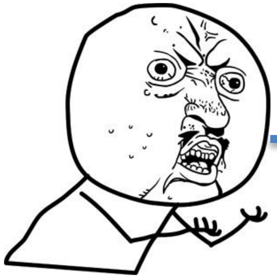
Algorithm Design and Pseudocode

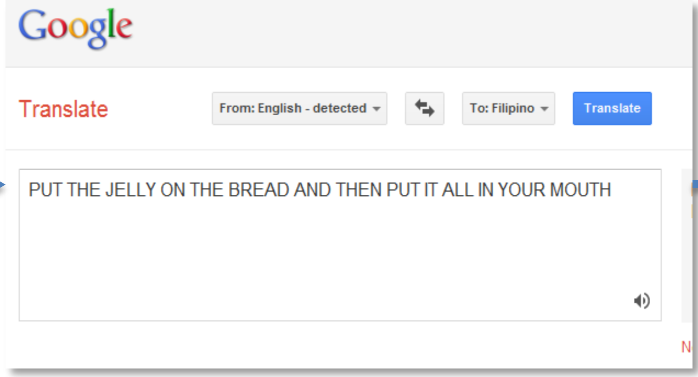
Remember Compilation?

You (the programmer)

The compiler

The computer
(your new best friend)





PUT THE JELLY ON THE BREAD AND THEN PUT IT ALL IN YOUR MOUTH

PUT THE JELLY ON THE
BREAD AND THEN PUT
IT ALL IN YOUR MOUTH

Ilagay ang
halaya SA tinapay AT pagkatapos a
y ilagay ito LAHAT SA
INYONG bibig

<u>The computer program</u>

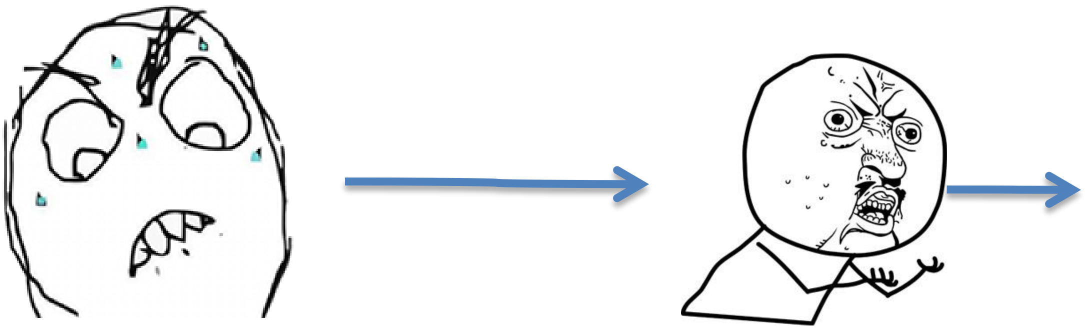
Machine code

English is Still Useful!

- Sometimes it is difficult to write an algorithm/computer program directly.

Problem:

```
*****
*****
*****
*****
*****
```



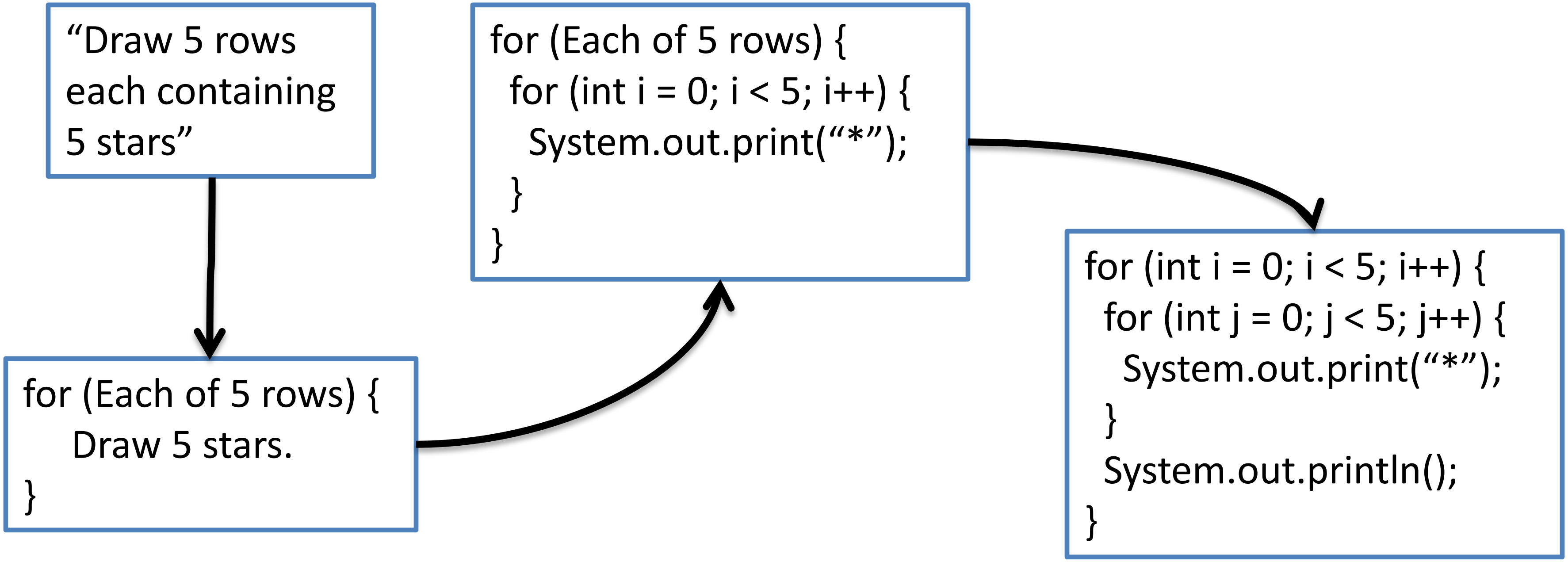
"Draw 5 rows each containing 5 stars"

/* ??? */

Pseudocode!

Pseudocode Helps Organize Your Thoughts

- Stuck and don’t know how to make progress?
  - Write an *English* description of your solution.
  - Transform that English description into code.



“Draw 5 rows each containing 5 stars”

```
for (Each of 5 rows) {
    Draw 5 stars.
}
```

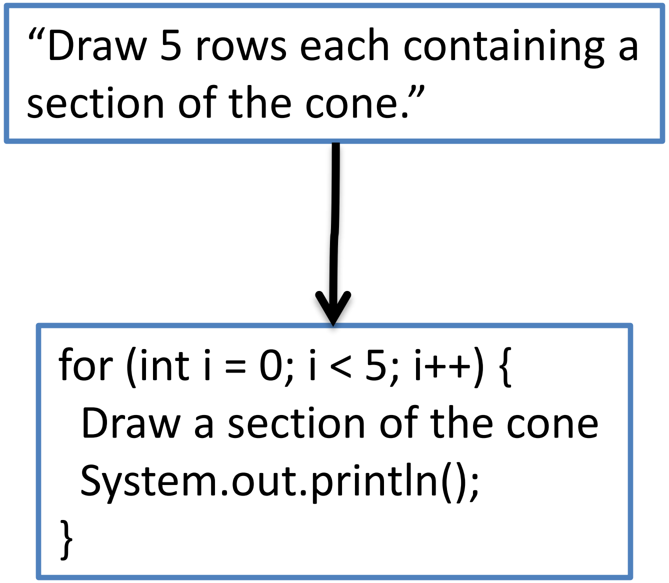
```
for (Each of 5 rows) {
  for (int i = 0; i < 5; i++) {
    System.out.print(“*”);
  }
}
```

```
for (int i = 0; i < 5; i++) {
  for (int j = 0; j < 5; j++) {
    System.out.print(“*”);
  }
  System.out.println();
}
```

By Example: A Cone

- Problem: draw the following cone shape.

```
    /\
   /--\
  /----\
 /------\
/--------\
```



"Draw 5 rows each containing a section of the cone."

```
for (int i = 0; i < 5; i++) {
  Draw a section of the cone
  System.out.println();
}
```

Cone Sections

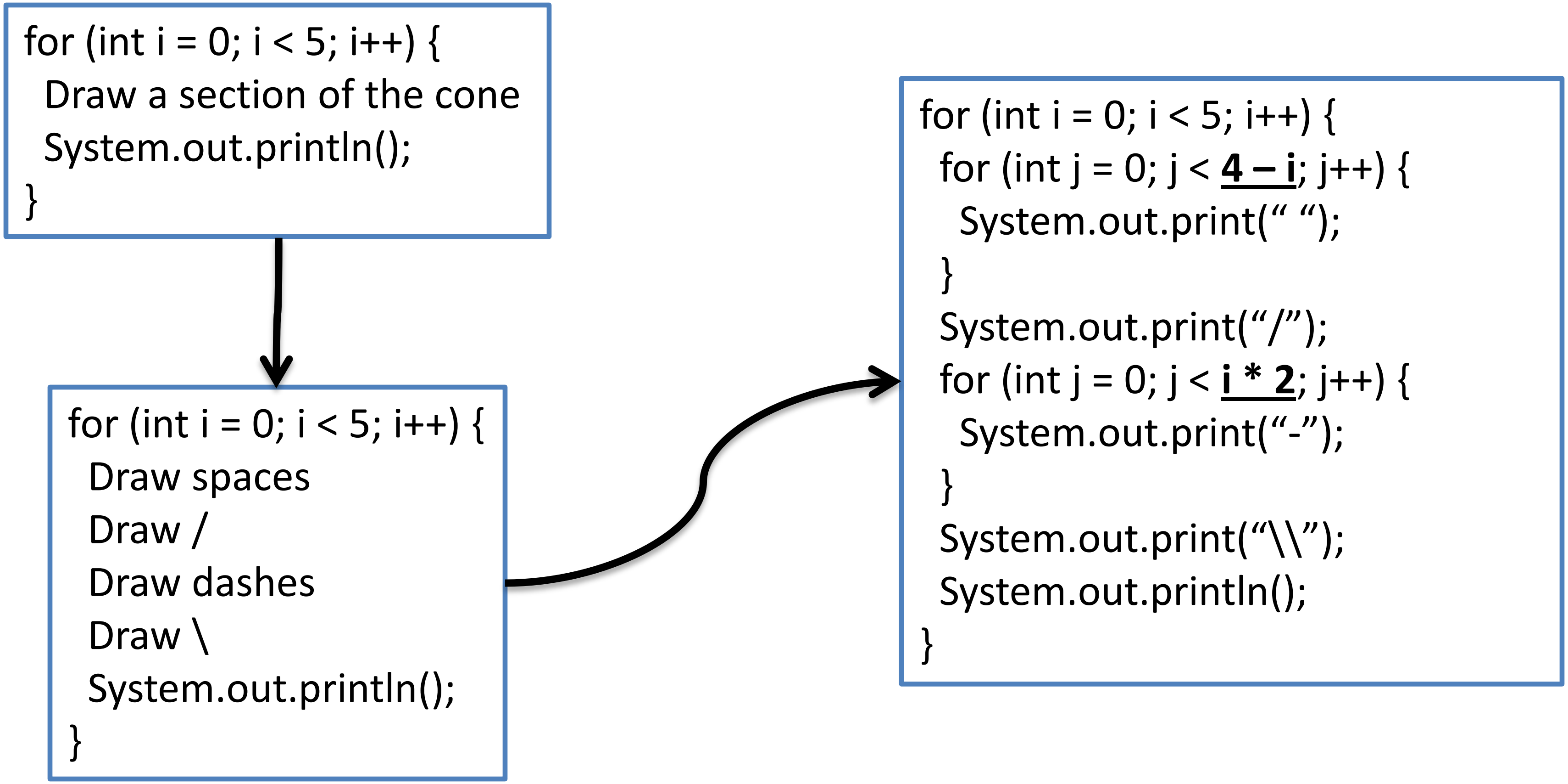
- Each line has the following form:
  - `<spaces> / <dashes> \`
  - Let’s find the pattern for each part!

```
    /\
   /--\
  /----\
 /------\
/--------\
```

Iteration/row (i)	Spaces	/	Dashes	\
0	4	1	0	1
1	3	1	2	1
2	2	1	4	1
3	1	1	6	1
4	0	1	8	1

- Formula for spaces: `4 - i`
- Formula for dashes: `i * 2`

From Tables to Code



```
for (int i = 0; i < 5; i++) {
  Draw a section of the cone
  System.out.println();
}
```

```
for (int i = 0; i < 5; i++) {
  Draw spaces
  Draw /
  Draw dashes
  Draw \
  System.out.println();
}
```

```
for (int i = 0; i < 5; i++) {
  for (int j = 0; j < 4 – i; j++) {
    System.out.print(" ");
  }
  System.out.print("/");
  for (int j = 0; j < i * 2; j++) {
    System.out.print("-");
  }
  System.out.print("\\");
  System.out.println();
}
```

Stop Being So Constant

- What value controls the height of the cone?

```
for (int i = 0; i < 5; i++) {
  for (int j = 0; j < 4 – i; j++) {
    System.out.print(“ “);
  }
  System.out.print(“/”);
  for (int j = 0; j < i * 2; j++) {
    System.out.print(“-”);
  }
  System.out.print(“\\”);
  System.out.println();
}
```

The height of the cone but not immediately obvious!
An example of a *magic number*.

Surprise! An *indirect use* of the height of the cone. If we change the height, then this number needs to change as well!

(Bad) Example: Quake III Arena

```
float Q_rsqrt( float number )
{
  long i;
  float x2, y;
  const float threehalfs = 1.5F;

  x2 = number * 0.5F;
  y  = number;
  i  = * ( long * ) &y;                       // evil floating point bit level hacking
  i  = 0x5f3759df - ( i >> 1 );               // what the fuck?
  y  = * ( float * ) &i;
  y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//    y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed

  return y;
}
```

- *Note:* code is written in C but should be (somewhat) understandable!
- This is the exact source from the game!
- The constant was a source of much debate! See Fast Inverse Square Root @ Wikipedia for more details.

Solution to Magic Numbers: Class Constants

- Class constants let us “document” magic numbers by naming them and giving us a central place to change their values if necessary.

```
public class Cone {
 public static final int HEIGHT = 5;
  public static void main(String[] args) {
  for (int i = 0; i < HEIGHT; i++) {
   for (int j = 0; j < (HEIGHT – 1) – i; j++) {
    System.out.print(“ “);
   }
   System.out.print(“/”);
   for (int j = 0; j < i * 2; j++) {
    System.out.print(“-”);
   }
   System.out.print(“\\”);
   System.out.println();
  }
 }
}
```

Syntax of Class Constants

```
public class Cone {
  public static final int HEIGHT = 5;
    public static void main(String[] args) {
    for (int i = 0; i < HEIGHT; i++) {
/* Snip! */
```

- Constants are declared *outside of methods* but *inside the class*.
  - `public static final <type> <name> = <value>;`
- Constants are variables that cannot be reassigned.
- Convention is to NAME_THEM_LIKE_THIS.