Control Statements in Java

Announcements

- Assignment 1 (Karel) due at 3:15PM today.
  - You can use a late period and submit on Wednesday of next week by 3:15PM.
  - ***It's okay to use a late period on the Karel assignment – this is your first time programming!***
- Email assignment due Sunday.
  - Looking forward to meeting you!
- Assignment 2 (**Welcome to Java!**) goes out, is due on Monday, January 26 at 3:15PM.
  - Play around with graphics, control structures, and methods!
  - Some of these program require the use of methods. We'll cover methods today and at the start of Wednesday's lecture.

Announcements

- Continuing a longstanding tradition, Eric Roberts will be showing a video of the “I Have a Dream” speech on Monday in Gates B12 at 2:15PM.
- Highly recommended, especially if you haven't seen it before.

Outline for Today

- **The `if` Statement Revisited**
  - Now with variables!
- **The `for` Loop Revisited**
  - Now with graphical goodies!
- **Methods and Parameters**
  - Customizing the behavior of your methods.

Control Structures

Control Structures

- When using Karel, we used these three control structures:
  - `if` statements.
  - `for` loops.
  - `while` loops.
- These exist in standard Java as well!
- Let's see what they look like.

Control Structures

if
for
while

Control Structures

```
if
for
while
```

`if` statements

```
if (condition) {
… statements to run if condition holds …
}
```

Boolean Expressions

- A ***boolean expression*** is a test for a condition (it is either `true` or `false`).
- Value comparisons:

`==`	"equals"	*(note: not single =)*
`!=`	"not equals"	
`>`	"greater than"	
`<`	"less than"	
`>=`	"greater than or equal to"	
`<=`	"less than or equal to"	

Logical Operators

- We use ***logical operators*** combine or modify boolean values.
- Logical ***NOT***: `!p`

```
if (!isWeekday()) {
    relaxAndUnwind();
}
```

- Logical ***AND***: `p && q`

```
if (youreHappy() && youKnowIt()) {
    clapYourHands();
}
```

- Logical ***OR***: `p || q` *(inclusive OR)*

```
if (hasPuppy() || hasKitty()) {
    beHappy();
}
```

- Order of precedence given above.

Or `else`

```
if (condition) {
… statements to run if condition holds …
} else {
… statements to run if condition doesn't hold …
}
```

Cascading if

```
if (score >= 90) {
  println(" AWWWW YEAHHHHH ");
} else if (score >= 80) {
  println(" <(^_^)> ");
} else if (score >= 70) {
  println(" : - | ");
} else if (score >= 60) {
  println(" ಠ_ಠ ");
} else {
  println(" (ノ°□° )ノ ︵ ┻━┻ ");
}
```

Control Statements

if
for
while

Control Statements

```
if
for
while
```

The Syntax

- As with Karel, to repeat a set of commands ***N*** times, use the following code:

```
for (int i = 0; i < N; i++) {
    // … statements to execute …
}
```

- We'll talk about how exactly this works next time. For now, let's focus on what we can do with it!

Accessing the Counter

- Inside a for loop, the variable `i` keeps track of the index of the current loop, starting at 0.
  - First time through the loop: `i` = 0
  - Second time through the loop: `i` = 1
  - Third time through the loop: `i` = 2
- Let's see an example of this.

Accessing the Counter

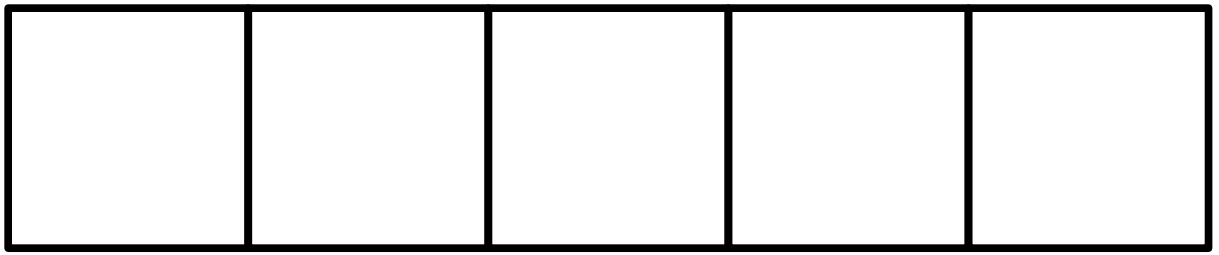
- Suppose we want to print out the first fifteen multiples of 50 (0, 50, 100, …).
- We can accomplish this using a `for` loop.

```
for (int i = 0; i < 15; i++) {
    println(i * 50);
}
```

- Do you see why?

Accessing the Counter

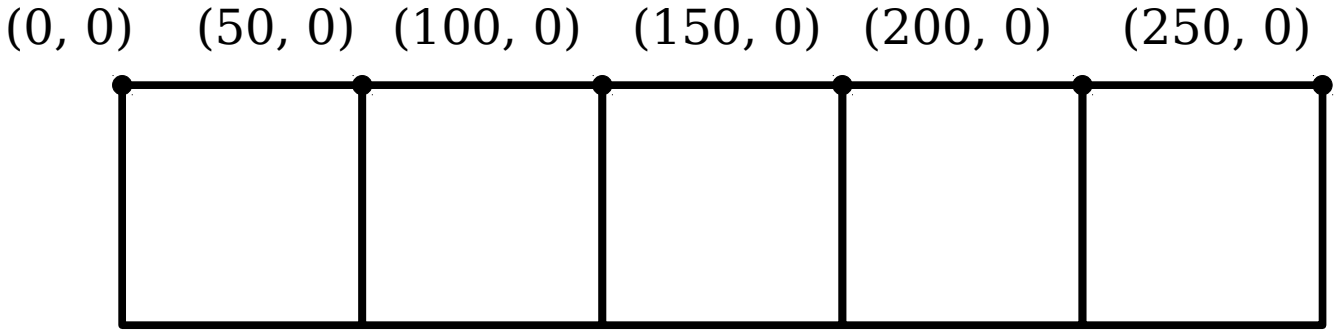
- Suppose we want to draw a row of boxes, like these:



- Suppose each box is 50 pixels wide and 50 pixels tall.
- Look where their corners are… seem familiar?

Accessing the Counter

- Suppose we want to draw a row of boxes, like these:



- Suppose each box is 50 pixels wide and 50 pixels tall.
- Look where their corners are… seem familiar?

Double For Loops

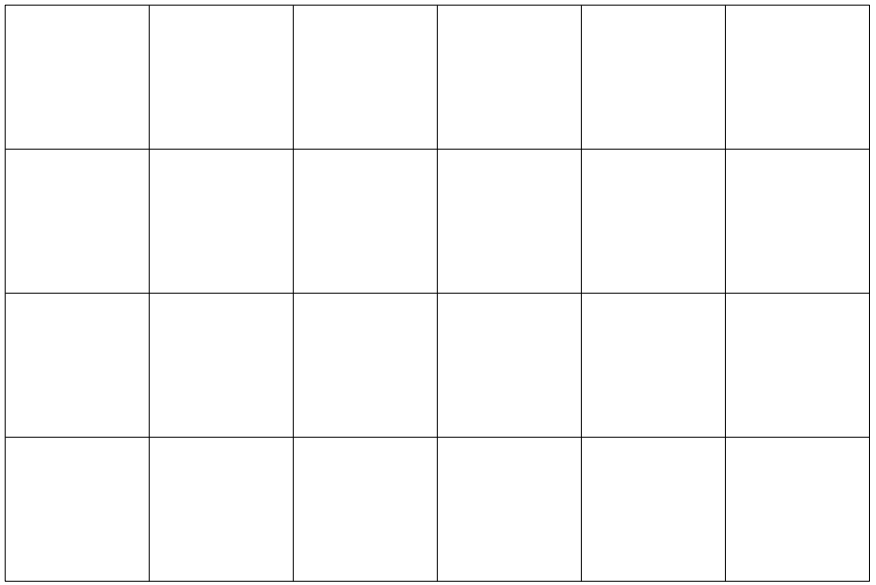
- You can put `for` loops inside of `for` loops! This is sometimes called a ***double for loop***.
- Syntax:

```
for (int i = 0; i < M; i++) {
  for (int j = 0; j < N; j++) {
     // … statements to execute …
  }
}
```

- This will run through all possible combinations of *i* and *j* where *i* is less than ***M*** and *j* is less than ***N***.

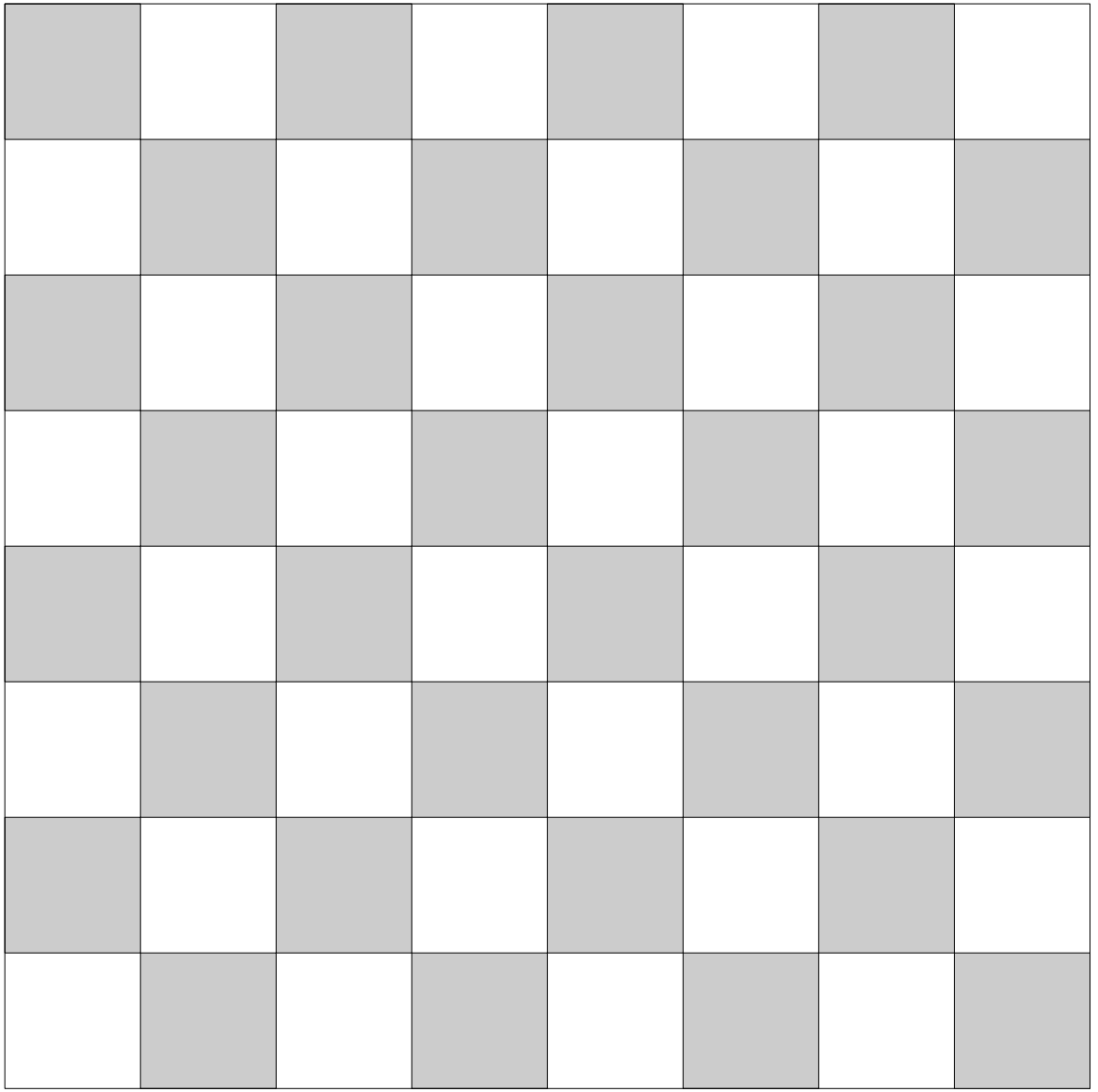
Double For Loops

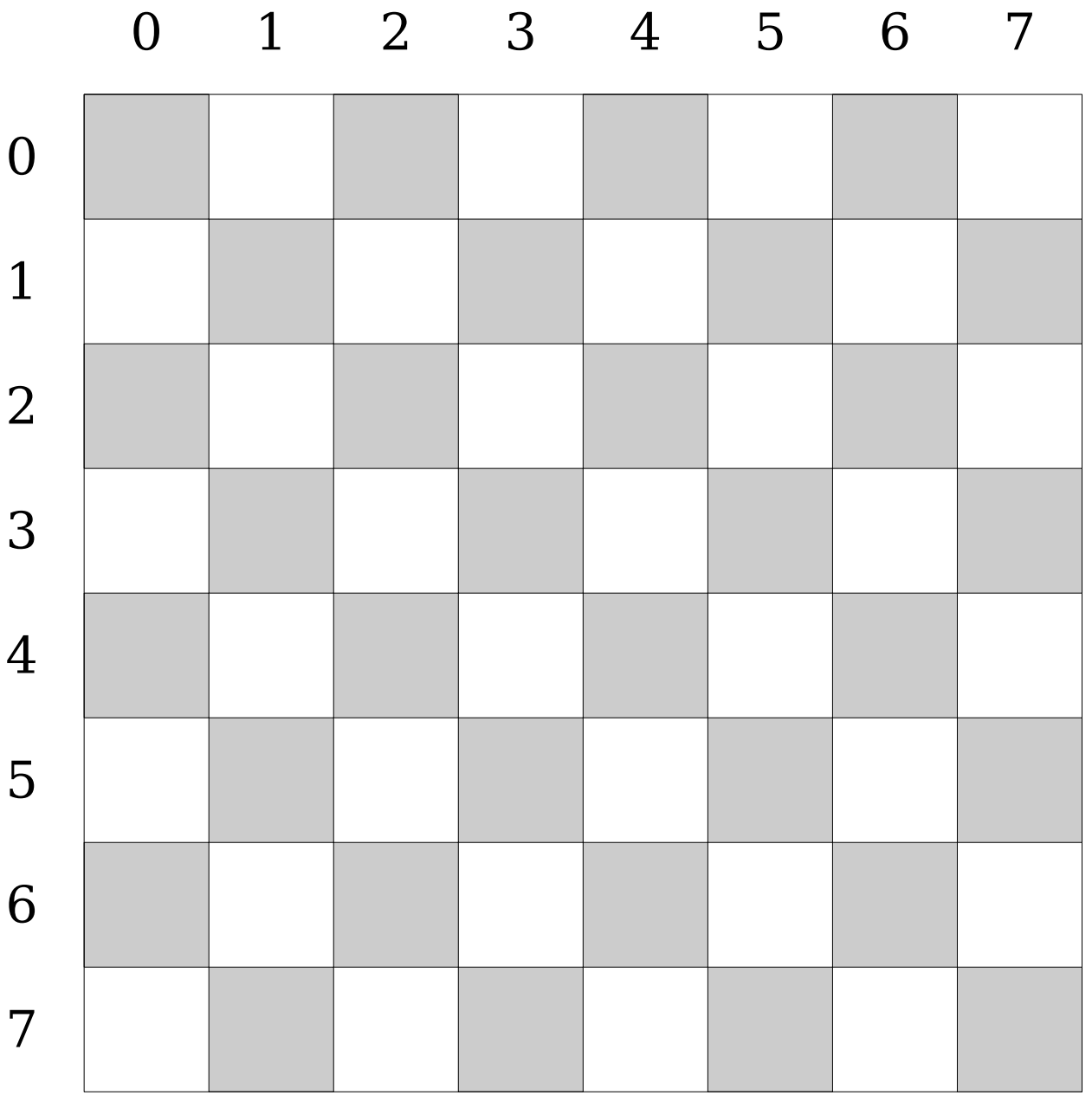
- Double for loops arise frequently when working with graphics.
- Suppose we want to draw this grid of boxes, each of which is 50 × 50:



- Notice anything about the corner positions?

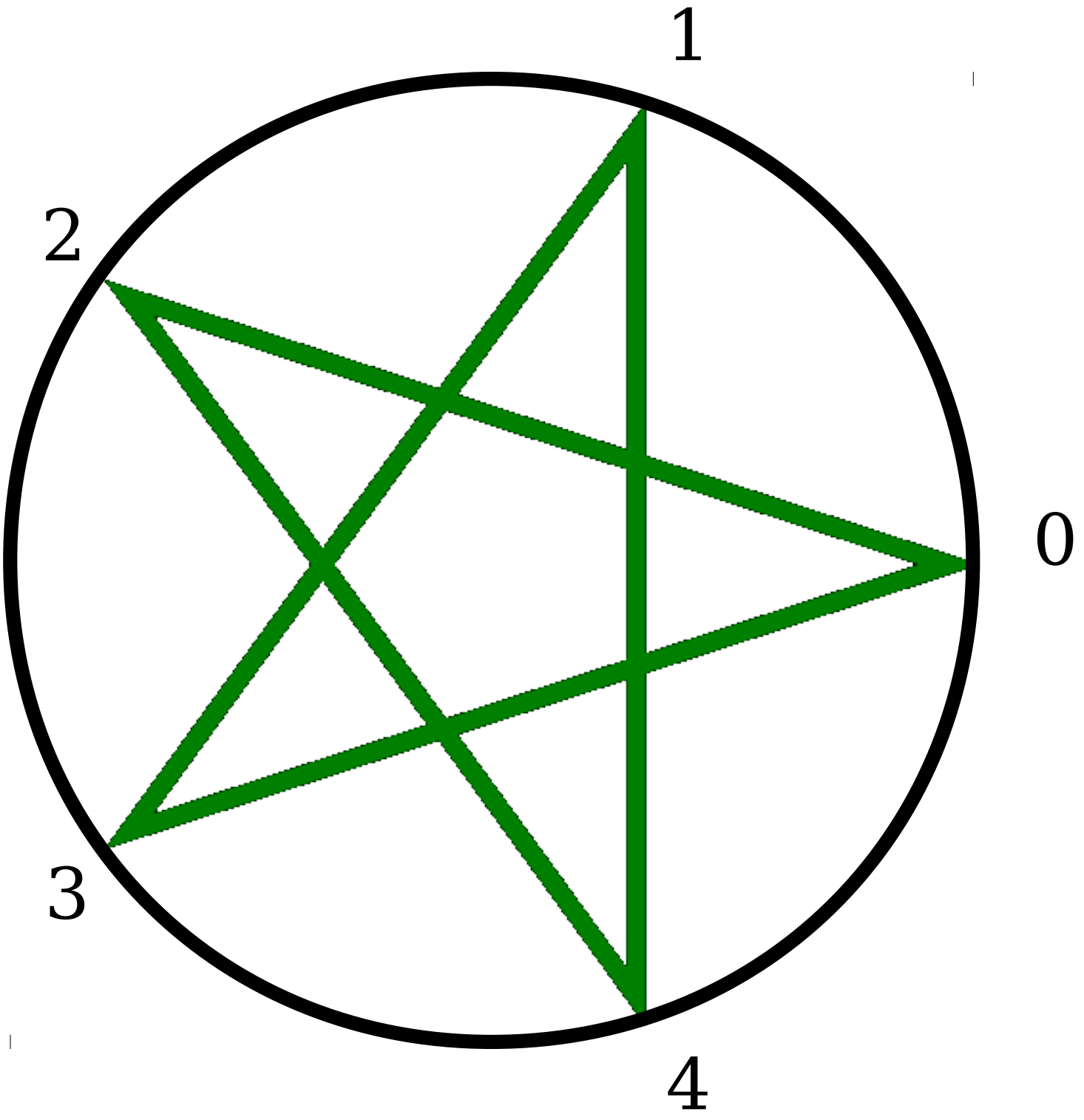
Drawing a Checkerboard, Java Style

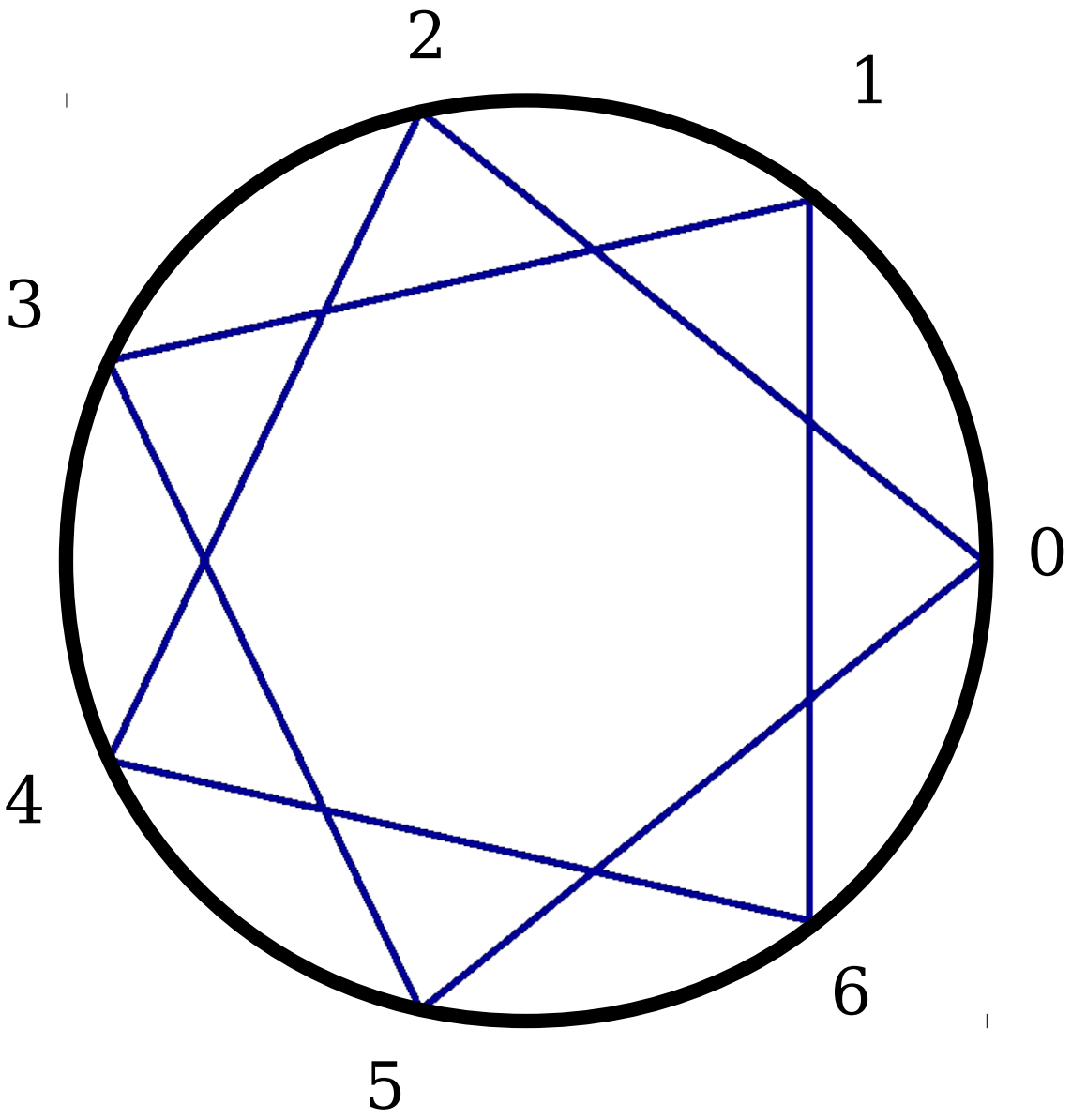



0 1 2 3 4 5 6 7
0
1
2
3
4
5
6
7

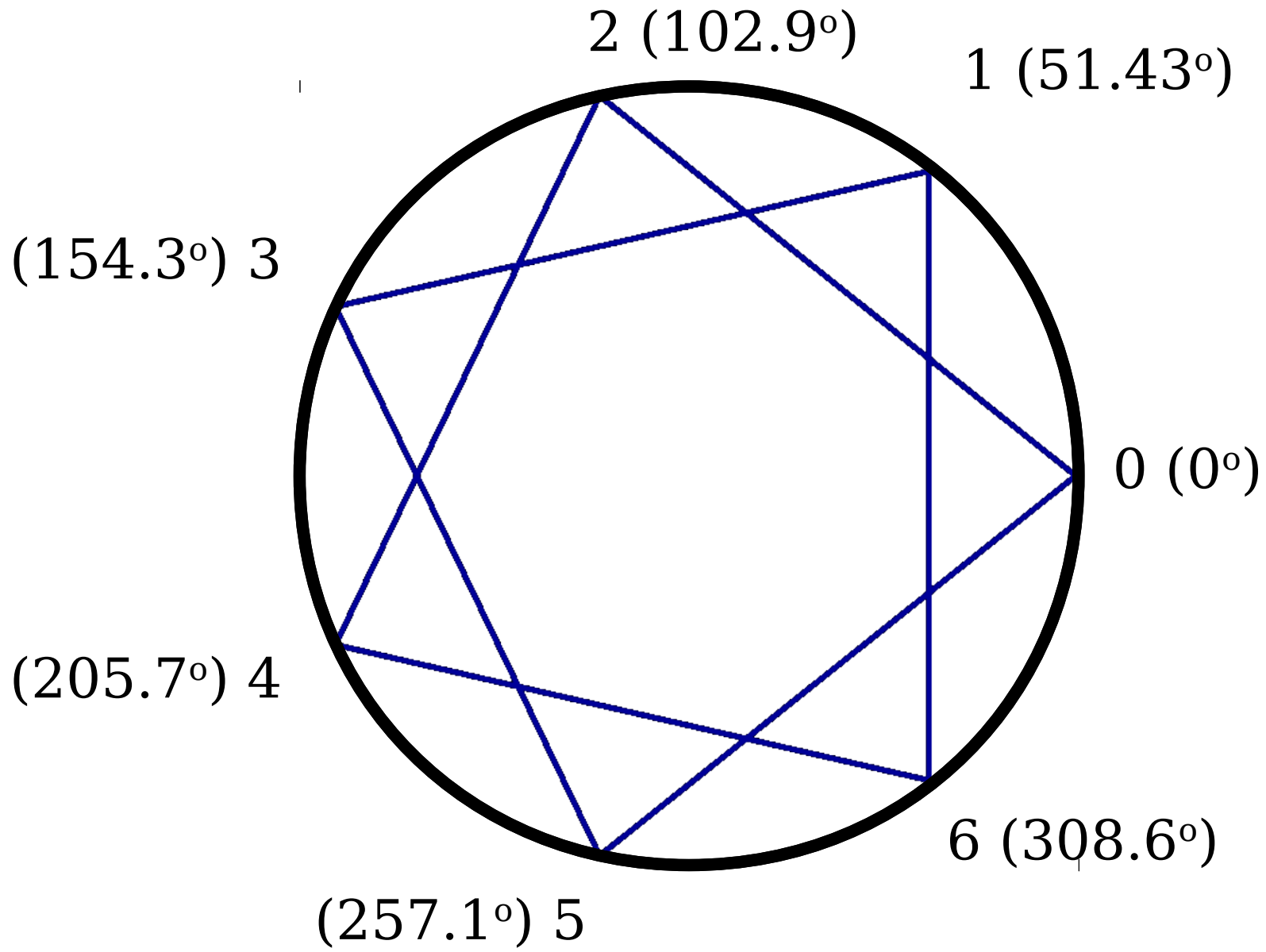
	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	8
2	2	3	4	5	6	7	8	9
3	3	4	5	6	7	8	9	10
4	4	5	6	7	8	9	10	11
5	5	6	7	8	9	10	11	12
6	6	7	8	9	10	11	12	13
7	7	8	9	10	11	12	13	14

Methods Revisited


1
2
0
3
4

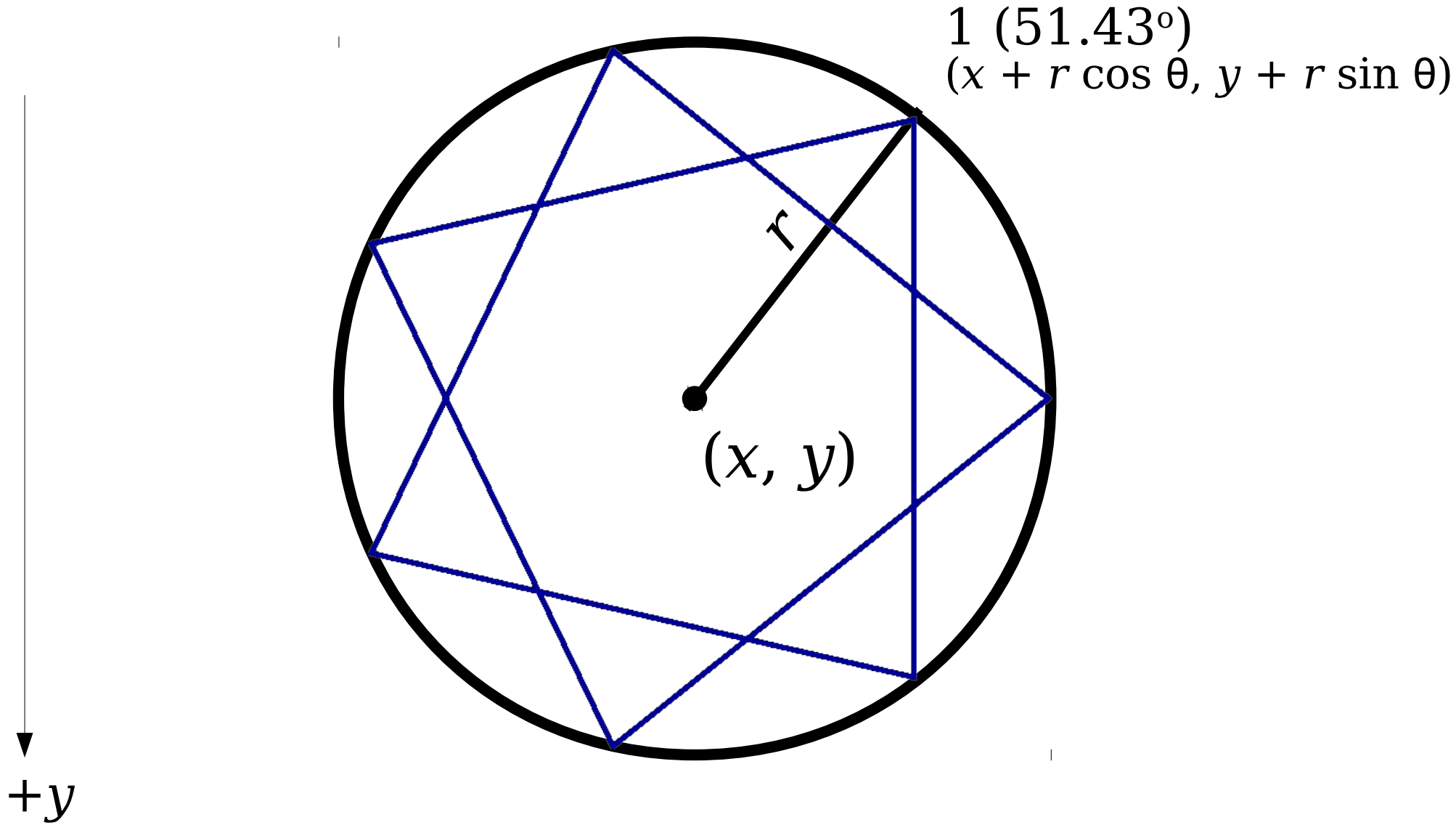


Each point k is connected to point $k + 2$, after wrapping around.



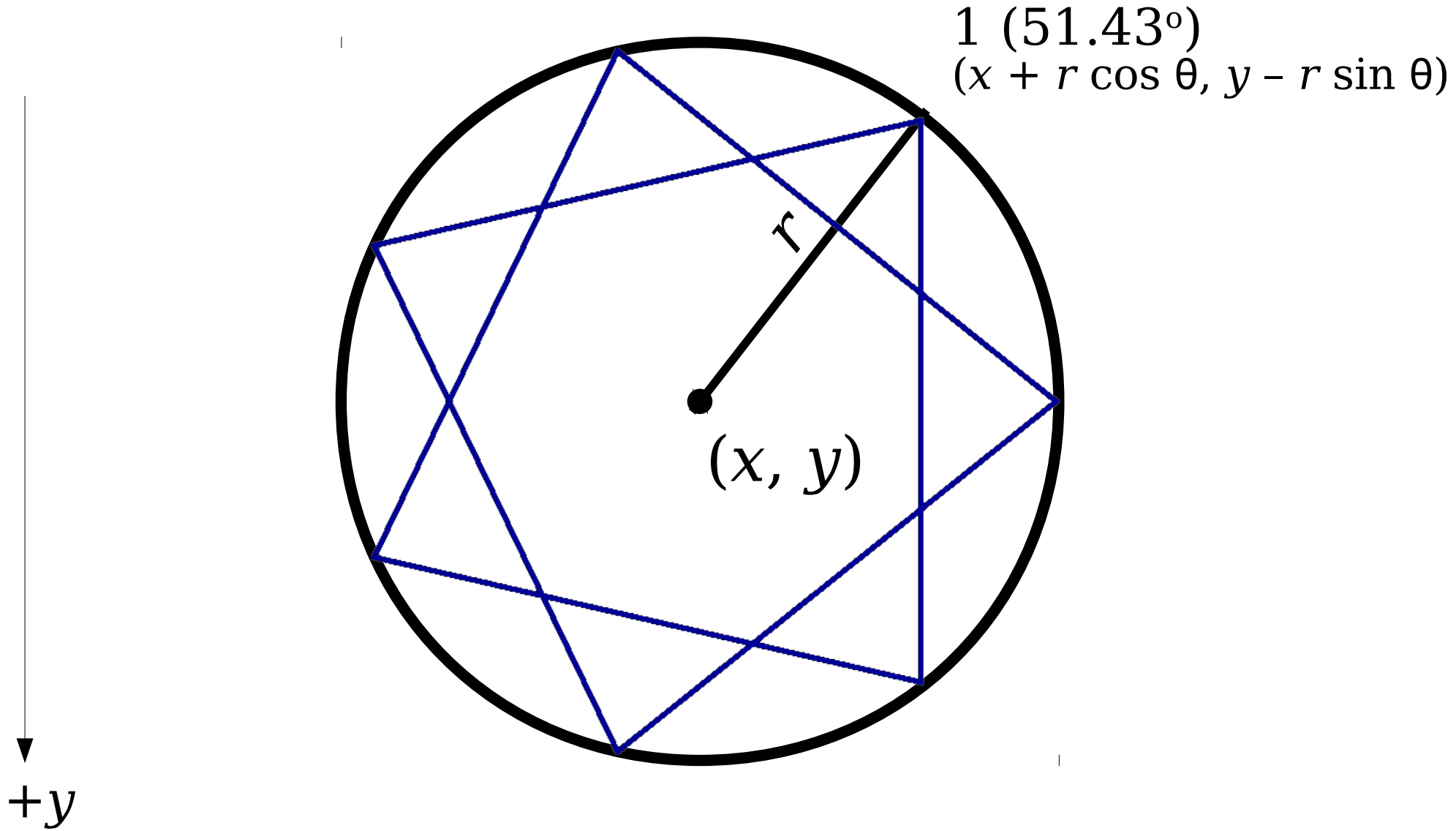
Each point k is connected to point $k + 2$, after wrapping around.

Point k is at $\frac{k}{numSides} \times 360^{\circ}$



Each point k is connected to point $k + 2$, after wrapping around.

Point k is at $\frac{k}{numSides} \times 360^{\circ}$



Each point k is connected to point $k + 2$, after wrapping around.

Point k is at $\frac{k}{numSides} \times 360^{\circ}$

Passing Parameters

- A method can accept ***parameters*** when it is called.
- Syntax:

```
private void name(parameters) {
    /* … method body … */
}
```

- The values of the parameters inside the method are set when the method is called.
- The values of the parameters can vary between calls.