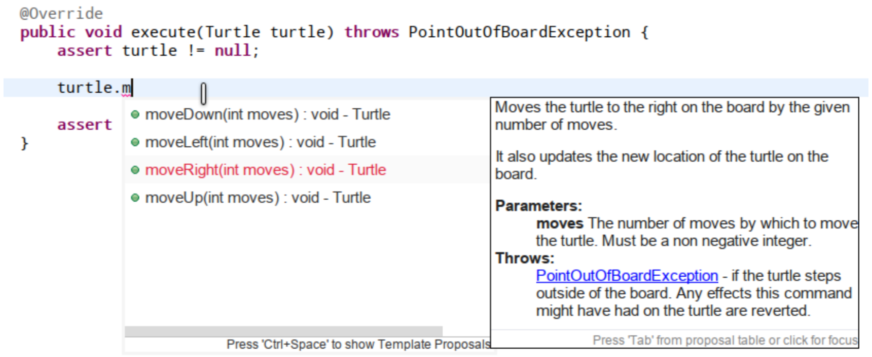
Documentation (Javadoc)

Aliaksei Syrel
(slides by Aaron Karper)

February 28, 2014

Contents

Motivation example



```
@Override
public void execute(Turtle turtle) throws PointOutOfBoardException {
    assert turtle != null;

    turtle.m

    assert
}
```

● moveDown(int moves) : void - Turtle
● moveLeft(int moves) : void - Turtle
● moveRight(int moves) : void - Turtle
● moveUp(int moves) : void - Turtle

Press 'Ctrl+Space' to show Template Proposals

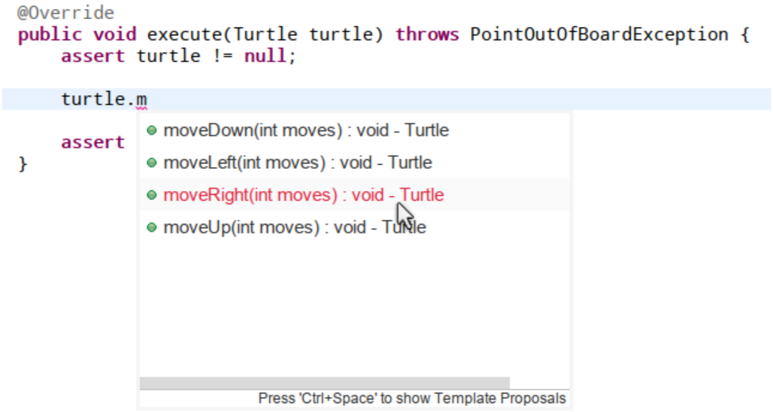
Moves the turtle to the right on the board by the given number of moves.

It also updates the new location of the turtle on the board.

Parameters:
  moves The number of moves by which to move the turtle. Must be a non negative integer.
Throws:
  PointOutOfBoardException - if the turtle steps outside of the board. Any effects this command might have had on the turtle are reverted.

Press 'Tab' from proposal table or click for focus

Motivation example



```
@Override
public void execute(Turtle turtle) throws PointOutOfBoardException {
    assert turtle != null;

    turtle.m

    assert
}
```

moveDown(int moves) : void - Turtle
moveLeft(int moves) : void - Turtle
moveRight(int moves) : void - Turtle
moveUp(int moves) : void - Turtle

Less talk, more code?

Do not write *how* it works, write *what* it does

- Inform other coders how to use your code without having to read it.
- Understanding your code
  - Know what the intention is.
  - Know what the code does and does not need to handle.
- Specification
  - Reminder to yourself what you need to do.
  - Makes you think about your responsibilities.
- Only necessary for public interface (?)

What is Good Documentation?

- Don't be a poem writer
    - No fillers – *This method/function/class...* is not necessary.
    - Make the first sentence count – JavaDoc assumes it to be the summary.

```
/**
 * This is a nice method to assert equality
 * of chars at a given index
 */
```

What is Good Documentation?

- Remember to describe
  - Responsibilities (pre- & post-conditions)
  - Corner cases. e.g. `null`? negative `int`s?
  - Exceptions (@throws)
  - Link to other documentation – with `@see` or `@link`

Class Comments

- What is the class responsible for? What information does it hold, what things can it do?
- Who uses this class? How should the class be used?
- Does this class need special treatment, for example a lifetime?

Method Comments

- Use `@param` to
  - Define constraints
  - What are your preconditions?
- Use `@return` to
  - Offer more specific information.
  - What are your postconditions?
- Use `@throws` to
  - Describe exceptional conditions
  - Name possible exception types

What NOT to do

```
public class ServerProxy implements IServer{
```

What NOT to do

```
/**
 * Constructor.
 */
public ServerProxy(String url, int port) throws
    NetworkConnectionException {
        // ...
}
```

What NOT to do

```
/**
 * Ends the connection.
 */
public void disconnect() throws DeadConnectionException {
        // ...
}

/**
 * Returns the number of jobs.
 */
public int getJobCount() throws DeadConnectionException {
        // ...
}
```

What NOT to do

```
/**
 * Returns the url of the server.
 */
public String getUrl() {
        return url;
}
```

How to do class comments better (before)

```
public class ServerProxy implements IServer{
```

How to do class comments better (after)

```
/**
 * Relays method calls to a remote {@see Server}.
 * <p>
 * The proxy is responsible for establishing and
 * keeping a connection to the server. The caller
 * must ensure that a connection is destroyed with
 * the {@see #disconnect} method.
 */
public class ServerProxy implements IServer {
```

How to do constructor comments better (before)

```
/**
 * Constructor.
 */
public ServerProxy(String url, int port) throws
    NetworkConnectionException {
        // ...
}
```

How to do constructor comments better (after)

```
/**
 * Established a connection to a remote server.
 * Throws if it fails to do so.
 *
 * @param url address that can either be resolved
 *            via hosts.conf or DNS or is an IP
 *            address.
 * @param port port to connect to on the server. A
 *             positive integer, typically above 1024.
 *             Must be the same as the {@see Server}
 *             uses with its {@see Server#listenOn} method.
 * @throws NetworkConnectionException if it was
 *            not able to initiate a connection.
 */
public ServerProxy(String url, int port)
        throws NetworkConnectionException {
        // ...
}
```

How to do method comments better (before)

```
/**
 * Ends the connection.
 */
public void disconnect() throws DeadConnectionException {
        // ...
}

/**
 * Returns the number of jobs.
 */
public int getJobCount() throws DeadConnectionException {
        // ...
}
```

How to do method comments better (after)

```
/**
 * Ends the connection. After this call, no other
 * method call is valid, including this one. The
 * server is not affected by this.
 */
public void disconnect() throws DeadConnectionException {
        // ...
}

/**
 * Returns the number of jobs running on the server.
 *
 * @return a non-negative integer that is the
 *         number of jobs that are alive.
 */
public int getJobCount() throws DeadConnectionException {
        // ...
}
```

How to do method comments better (before)

```
/**
 * Returns the url of the server.
 */
public String getUrl() {
        return url;
}
```

Some times no comments are best comments

```
public String getUrl() {
        return url;
}
```