

Comments, Memory and Assertions

Instructors: Sam McCauley and Dan Barowy

February 25, 2022

Admin

- Questions or comments?

Javadoc Comments

Javadoc comments

- Unified way of commenting java code

Javadoc comments

- Unified way of commenting java code
- Allows others to more effectively understand what you're doing

Javadoc comments

- Unified way of commenting java code
- Allows others to more effectively understand what you're doing
- Can generate html automatically!

Javadoc comments

- Unified way of commenting java code
- Allows others to more effectively understand what you're doing
- Can generate html automatically!
- Some editors can use javadocs to help you code

Where to use javadoc comments

- Official Oracle rules: every class, method, and instance variable

Where to use javadoc comments

- Official Oracle rules: every class, method, and instance variable
- We'll be focusing on the *methods*: every method you write should have a javadoc comment

Where to use javadoc comments

- Official Oracle rules: every class, method, and instance variable
- We'll be focusing on the *methods*: every method you write should have a javadoc comment
- We'll be checking for this this Lab 3 and beyond

javadoc notation

- Starts with `/**`

javadoc notation

- Starts with `/**`
- Every new line starts with `*`. Finally ends with `*/`.

javadoc notation

- Starts with `/**`
- Every new line starts with `*`. Finally ends with `*/`.
- Tags denoted with `@` character (examples on next slides). New line for each tag

Required javadoc elements

- First line gives a short method description

Required javadoc elements

- First line gives a short method description
- For every parameter in the method, need an `@param` tag that names the parameter, and describes it
 - Required even if obvious!

Required javadoc elements

- First line gives a short method description
- For every parameter in the method, need an `@param` tag that names the parameter, and describes it
 - Required even if obvious!
- If the method has a non-void return type, need an `@return` tag that describes what is returned
 - Required even if obvious!

Required javadoc elements

- First line gives a short method description
- For every parameter in the method, need an `@param` tag that names the parameter, and describes it
 - Required even if obvious!
- If the method has a non-void return type, need an `@return` tag that describes what is returned
 - Required even if obvious!
 - Two more (CS 136 only): `@pre` and `@post` for pre and post conditions

Pre- and Post-Conditions

Definition

- **Pre conditions:** what you are assuming is true *before* the method begins

Definition

- **Pre conditions:** what you are assuming is true *before* the method begins
- **Post conditions:** what you are assuming is now true *after* the method ends

Definition

- **Pre conditions:** what you are assuming is true *before* the method begins
- **Post conditions:** what you are assuming is now true *after* the method ends
- These are instructions for other people using your methods. You're not responsible for what happens if they ignore them!

Definition

- **Pre conditions:** what you are assuming is true *before* the method begins
- **Post conditions:** what you are assuming is now true *after* the method ends
- These are instructions for other people using your methods. You're not responsible for what happens if they ignore them!
- Let's look at some examples

Pre condition example: get

```
/**
 * Fetch the element at a particular index.
 * The index of the first element is zero.
 *
 * @param index the index of the value sought.
 * @return A reference to the value found in the vector.
 * @pre 0 <= index && index < size()
 * @post Returns a reference to the value found in the vector.
 */
public E get(int index)
{
    return (E)elementData[index];
}
```

Anyone calling this method *must* make sure that `index` is between 0 and `size()`.

Post condition example: set

```
/**
 * Change the value stored at location index.
 *
 * @param index The index of the new value.
 * @param obj The new value to be stored.
 * @return The value previously stored at index
 * @pre 0 <= index && index < size()
 * @post element value is changed to obj
 * @post Returns the value previously stored at index
 */
public E set(int index, E obj)
{
    E previous = (E)elementData[index];
    elementData[index] = obj;
    return previous;
}
```

Return vs post

- You do need to include the return values as a post condition

Return vs post

- You do need to include the return values as a post condition
- Yes it's redundant (sorry!)

Return vs post

- You do need to include the return values as a post condition
- Yes it's redundant (sorry!)
- `@post` is also for a change in state of the data (i.e. something done by the method other than generating the return value). So you may need additional postconditions

Figuring out when to use pre and post conditions

- If someone calling this method can generate an error (with an empty string? negative integer? etc.), there should be a precondition addressing this

Figuring out when to use pre and post conditions

- If someone calling this method can generate an error (with an empty string? negative integer? etc.), there should be a precondition addressing this
- If the method accomplishes something other than returning a value, there should be a postcondition

Figuring out when to use pre and post conditions

- If someone calling this method can generate an error (with an empty string? negative integer? etc.), there should be a precondition addressing this
- If the method accomplishes something other than returning a value, there should be a postcondition
- Both (if they exist) should be listed in javadoc comments

Assert

Assert

- `structure5` package

Assert

- `structure5` package
- Java also has an `assert` keyword; we won't use it in this class. Use the `structure5 Assert`.

Assert

- `structure5` package
- Java also has an `assert` keyword; we won't use it in this class. Use the `structure5 Assert`.
- Basic idea of `Assert`: works like an `if` statement. But if condition is true, gives an error and exits the program.

Assert

- `structure5` package
- Java also has an `assert` keyword; we won't use it in this class. Use the `structure5 Assert`.
- Basic idea of `Assert`: works like an `if` statement. But if condition is true, gives an error and exits the program.
 - Why would we want this?

Assert

- `structure5` package
- Java also has an `assert` keyword; we won't use it in this class. Use the `structure5 Assert`.
- Basic idea of `Assert`: works like an `if` statement. But if condition is true, gives an error and exits the program.
 - Why would we want this?
 - Want to write easily debuggable code. If you need something to be true at a certain point in your code, check it!

Different methods in Assert

```
public static void pre(boolean test, String message);  
public static void post(boolean test, String message);  
public static void condition(boolean test, String message);  
public static void fail(String message);
```

- pre: checks precondition; outputs message if false
 - `Assert.pre(0 <= index && index < size(), "index is within bounds");`

Different methods in Assert

```
public static void pre(boolean test, String message);  
public static void post(boolean test, String message);  
public static void condition(boolean test, String message);  
public static void fail(String message);
```

- pre: checks precondition; outputs message if false
 - `Assert.pre(0 <= index && index < size(), "index is within bounds");`
- post, condition: same idea; slightly different output message

Different methods in Assert

```
public static void pre(boolean test, String message);
public static void post(boolean test, String message);
public static void condition(boolean test, String message);
public static void fail(String message);
```

- pre: checks precondition; outputs message if false
 - `Assert.pre(0 <= index && index < size(), "index is within bounds");`
- post, condition: same idea; slightly different output message
- fail: no condition; always exits

When to use Assert

- Good to use if even a semi-plausible case where something bad happens

When to use Assert

- Good to use if even a semi-plausible case where something bad happens
- If something would break your method, and isn't listed as pre-condition, should have an Assert

When to use Assert

- Good to use if even a semi-plausible case where something bad happens
- If something would break your method, and isn't listed as pre-condition, should have an Assert
- It's a good idea to use Asserts for pre-conditions to double-check things.

If we have time: Another asymptotic analysis example

Analyzing the Table class from wordgen

- Let's say we have a table containing n Associations

Analyzing the Table class from wordgen

- Let's say we have a table containing n Associations
- How long does add take?

Analyzing the Table class from wordgen

- Let's say we have a table containing n Associations
- How long does add take?
- How long does choose take?

Analyzing the `Table` class from `wordgen`

- Let's say we have a table containing n `Associations`
- How long does `add` take?
- How long does `choose` take?
- Let's say `WordGen` reads in a text of length n and generates n characters. Can we upper bound how long that takes?