Testing and Debugging

The scientific process of debugging, the information problem, and the value in automated testing.

 Kevin Lin, with thanks to many others.

Feedback from the Reading Quiz

Relatable list of incorrect assumptions. (Are assumptions bad?)

What does it mean to reproduce a bug?

How to write code so that it's easier to debug.

> "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you [won't be able] to debug it."

How to develop tests that cover the edge cases.

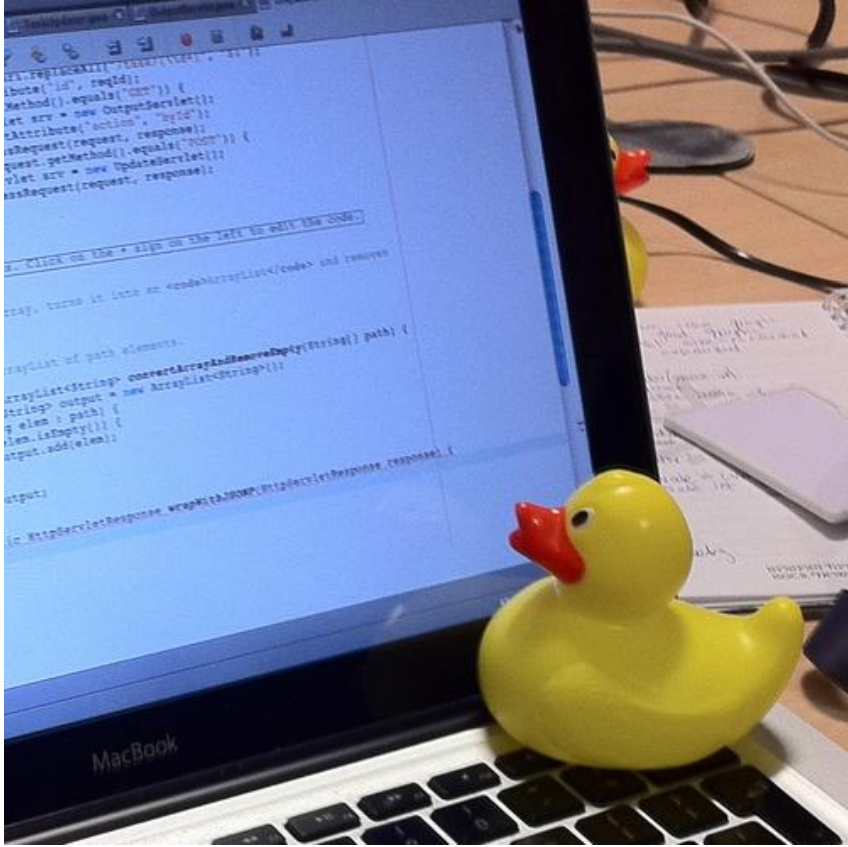
The Elements of Programming Style (Kernighan/McGraw-Hill)

Real world analogues

[Minimal working example]

[Rubber duck debugging]

[Scientific method]



Rubber duck assisting with debugging (Tom Morris/Wikimedia)

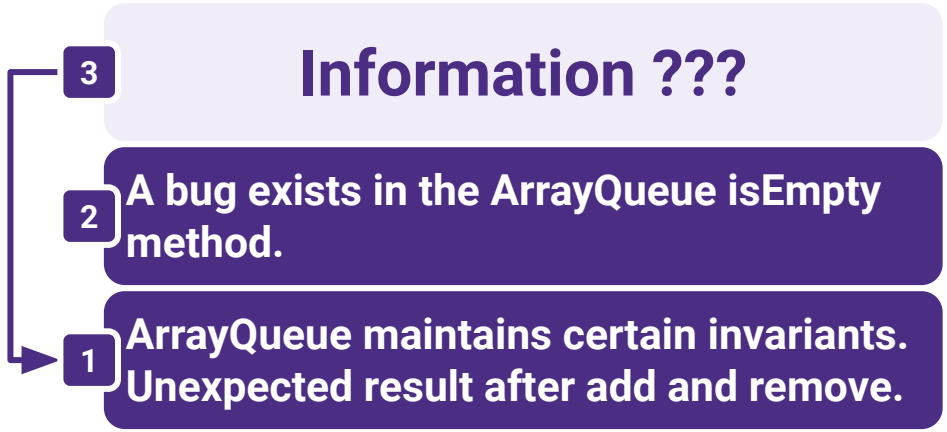
ArrayQueue Demo

The Role of Information

How are bugs fixed? Here's one proposal.

- Productive changes fix bugs.
- Information gathered about the system informs productive changes.
- A hypothesis guides information gathering and testing.
- Things we know about the problem inform how we choose hypotheses.

The point here is that information is the most important thing and you need to do whatever's necessary to get information.



What does debugging a program look like? (Julia Evans); The Debugging Mindset (Devon H. O'Dell/ACM Queue)

Generating Hypotheses

A good hypothesis identifies the cause of failure separately from where and when the program actually fails. The **state** of the ArrayQueue determines the behavior of isEmpty.

State

```
Item[] data
int size
int front
int back
```

3 **Information ???**

2 **A bug exists in the ArrayQueue isEmpty method.**

Descriptive Hypotheses

A good hypothesis identifies the cause of failure separate from where and when the program actually fails. The **state** of the ArrayQueue determines the behavior of isEmpty.

The hypothesis on the left suggests more about the problem than the one on the right.

The size variable is not set correctly, causing isEmpty to return false.

A bug exists in the ArrayQueue isEmpty method.

```
ArrayQueue1<Integer> queue = new ArrayQueue1<>();
queue.add(1);
queue.remove();
queue.add(3);
queue.remove();
queue.remove();
queue.add(6);
queue.remove();
System.out.println(
    "isEmpty() expected true, got " + queue.isEmpty());
```

Tests as a Source of Information

Propose a new hypothesis that, upon answering, could yield more information.

Join by Web

1. Go to **PollEv.com**
2. Enter **KEVINL**
3. Respond to activity

Hypotheses

Reference may cause the error in the program.

The method is unable to correctly remove elements from the data structure.

Two remove methods were used when there was only one value in the queue.

The remove is not used correctly.

Gathering Information

Ad-Hoc Testing

ArrayQueue1.main

```
queue.add(1);
queue.remove();
queue.add(3);
queue.remove();
queue.remove();
queue.add(6);
queue.remove();
System.out.println(
    "..." + queue.isEmpty());
```

ArrayQueue2.main

```
queue.add(0);
queue.remove();
queue.add(2);
queue.add(3);
queue.add(5);
queue.add(6);


System.out.println(
    "..." + queue.remove());
```

JUnit Testing

Simple JUnit Testing

```
org.junit.Assert.assertEquals(expected, actual);
```

ArrayQueueTest method

Call org.junit.Assert.assertEquals(...) to check that the expected equals the actual. If not, program terminates with a verbose message.

We can use this in place of writing out long print messages just to compare two arguments.

JUnit supports many more methods. (Check the online JUnit documentation for more.)

- assertEquals
- assertFalse
- assertNotNull

Better JUnit Testing

```
@org.junit.Test
```

ArrayQueueTest

The messages output by JUnit are kind of ugly, and invoking each test manually is annoying.

IntelliJ has built-in support for JUnit.

1. Annotate each test (Java method) with @org.junit.Test.
2. Change all test methods to non-static.
3. Use IntelliJ's built-in JUnit runner to run all tests and tabulate results.

This is called **boilerplate code**. IntelliJ can generate this code for you!

Even Better JUnit Testing

ArrayQueueTest

```
import org.junit.Test;
import static org.junit.Assert.*;
```

Don’t want to type out the name of the library (**org.junit.**Test, **org.junit.Assert.**assertEquals)?

To workaround this annoyance, start every file with two import statements.

The Role of Information

How are bugs fixed? Here's one proposal.

- Productive changes fix bugs.
- Information gathered about the system informs productive changes.
- A hypothesis guides information gathering and testing.
- Things we know about the problem inform how we choose hypotheses.

Two new regression tests

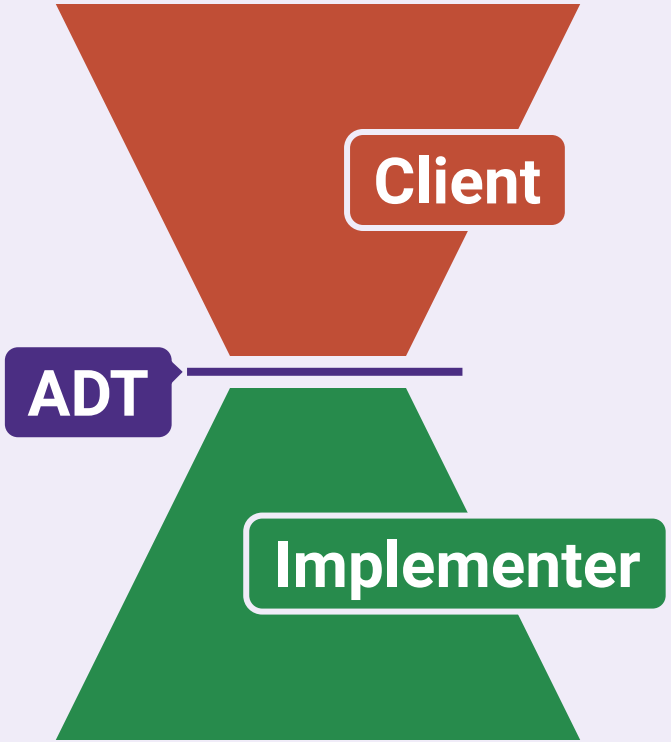
The point here is that information is the most important thing and you need to do whatever's necessary to get information.

What does debugging a program look like? (Julia Evans); The Debugging Mindset (Devon H. O'Dell/ACM Queue)

Q Testing as Planning

Suppose we're implementing ArrayQueue.

1. Describe a unit test we might want to write for ArrayQueue.
2. What behaviors does this test check? Describe in terms of the methods it checks as well as concepts like contracts, invariants, etc. that we've discussed in class.



Describe a unit test we might want to write for ArrayDeque.

Join by Web

1. Go to **PollEv.com**
2. Enter **KEVINL**
3. Respond to activity

Testing as Planning

The point here is that information is the most important thing and you need to do whatever's necessary to get information.

Not only does running a test improve our understanding of a problem, **but so does writing a test**!

> "I'm almost done, I just need to make sure it works."
> – Famous last words

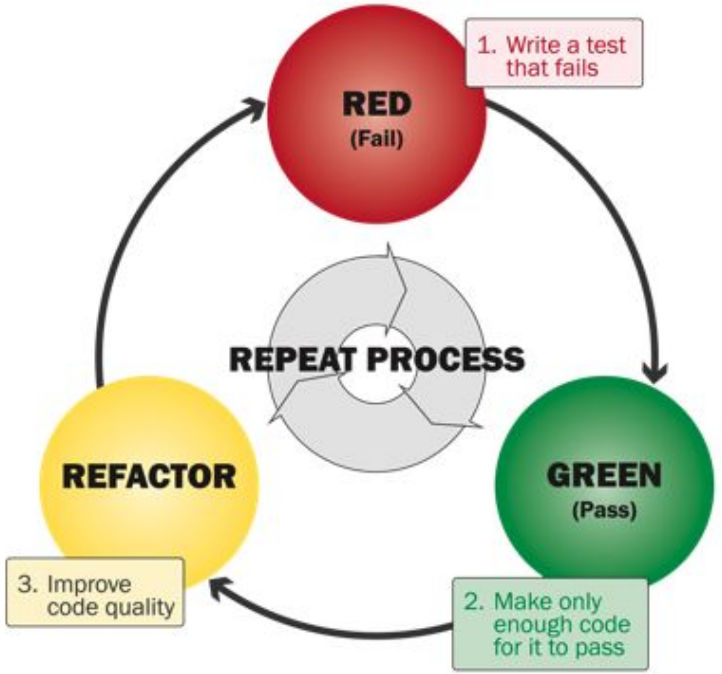
Tests are hard to write, but easy to run.

Maximize the benefit of testing by writing tests first (or early) and code afterwards.

Test-Driven Development

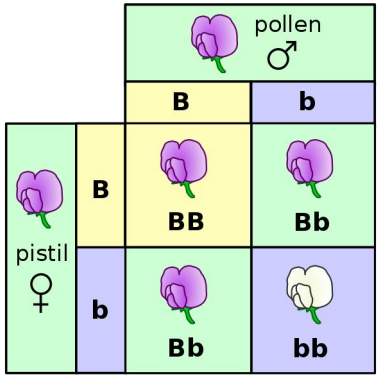
The point here is that information is the most important thing and you need to do whatever's necessary to get information.

1. Identify a new feature.
2. Write a unit test for that feature.
3. **RED**: Run the test. It should fail.
4. **GREEN**: Write code that passes test.
5. **REFACTOR**: Improve code quality.

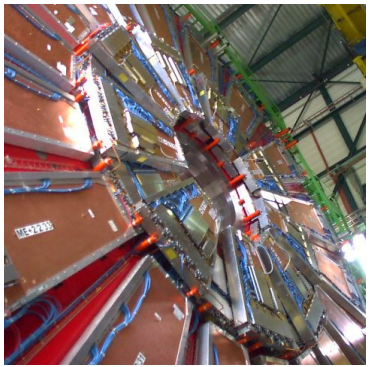


What does debugging a program look like? (Julia Evans); Red-Shirt, Red, Green, Refactor - A TDD Fairytale (Ryan Tablada)

Chemistry
(~1 test/day)



Biology
(~1 test/month)



Physics
(~1 test/month+)

CS
(1+ test/sec)

Running tests is virtually free compared to other sciences

The Role of Information

How are bugs fixed? Here's one proposal.

- Productive changes fix bugs.
- Information gathered about the system informs productive changes.
- A hypothesis guides information gathering and testing.
- Things we know about the problem inform how we choose hypotheses.

The point here is that information is the most important thing and you need to do whatever's necessary to get information.

3 **Modify the remove method to handle the special case of removing if empty.**

2 **The remove method decrements the size variable even when the queue is empty.**

1 **ArrayQueue maintains certain invariants. Unexpected result after add and remove.**

What does debugging a program look like? (Julia Evans); The Debugging Mindset (Devon H. O'Dell/ACM Queue)