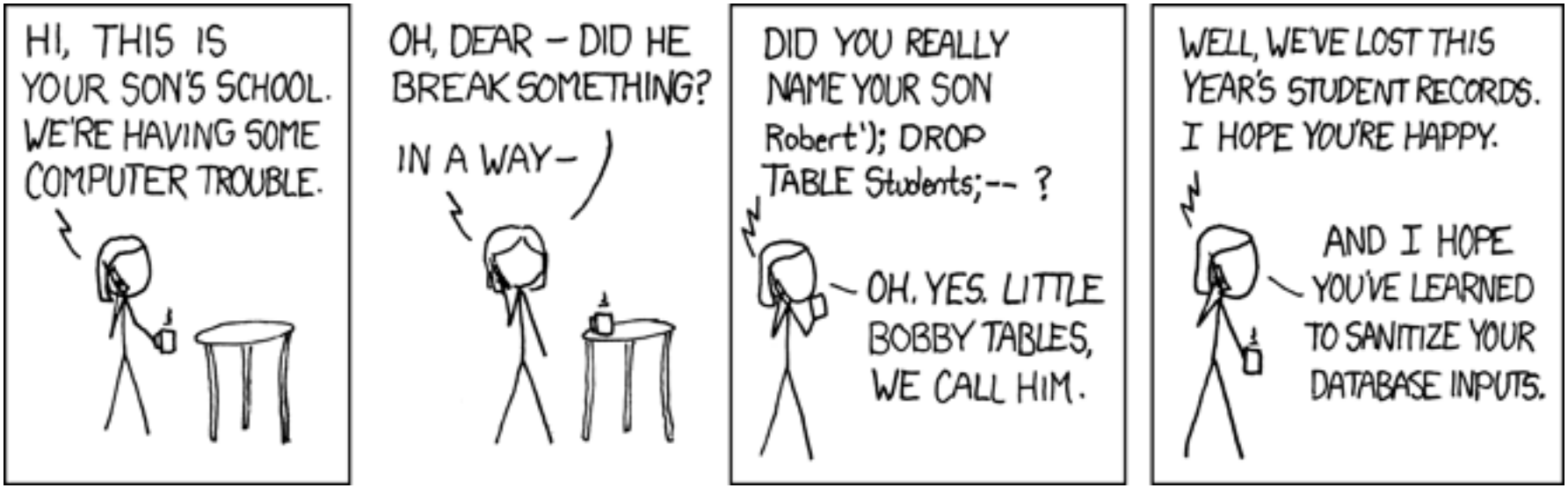

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

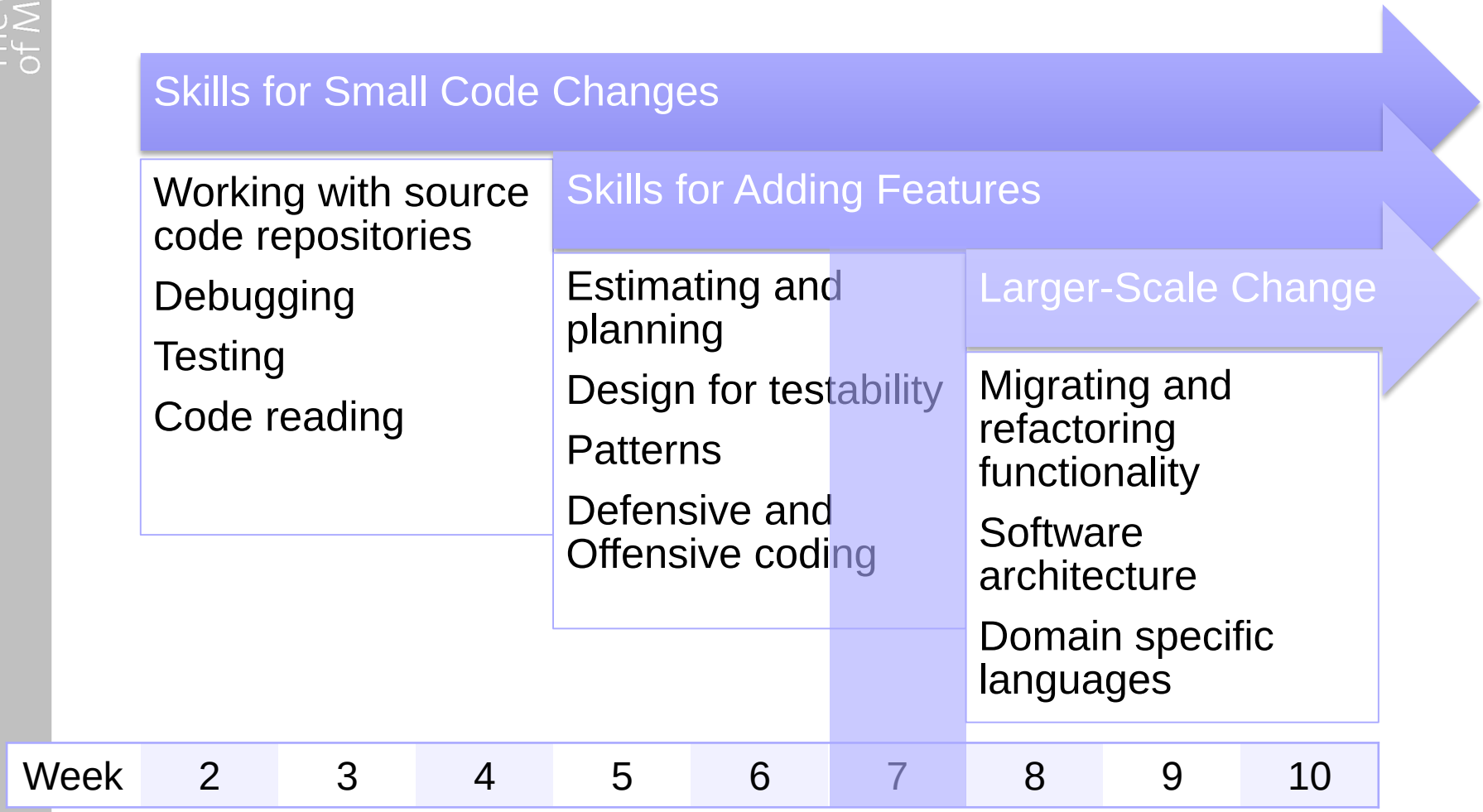
Software Patterns and Offensive and Defensive Coding

COMP23420: Software Engineering

Week 7

Caroline Jay and Robert Haines

Course Unit Roadmap (Weeks 2-10)



Link to the Coursework/Exam

- We will introduce design patterns and a couple of typical coding styles – offensive and defensive coding.
- Coursework: As you work on your projects and add new features it is important to be aware of standard methods of coding.
- Exam: There will be questions on patterns and offensive and defensive coding.

Patterns

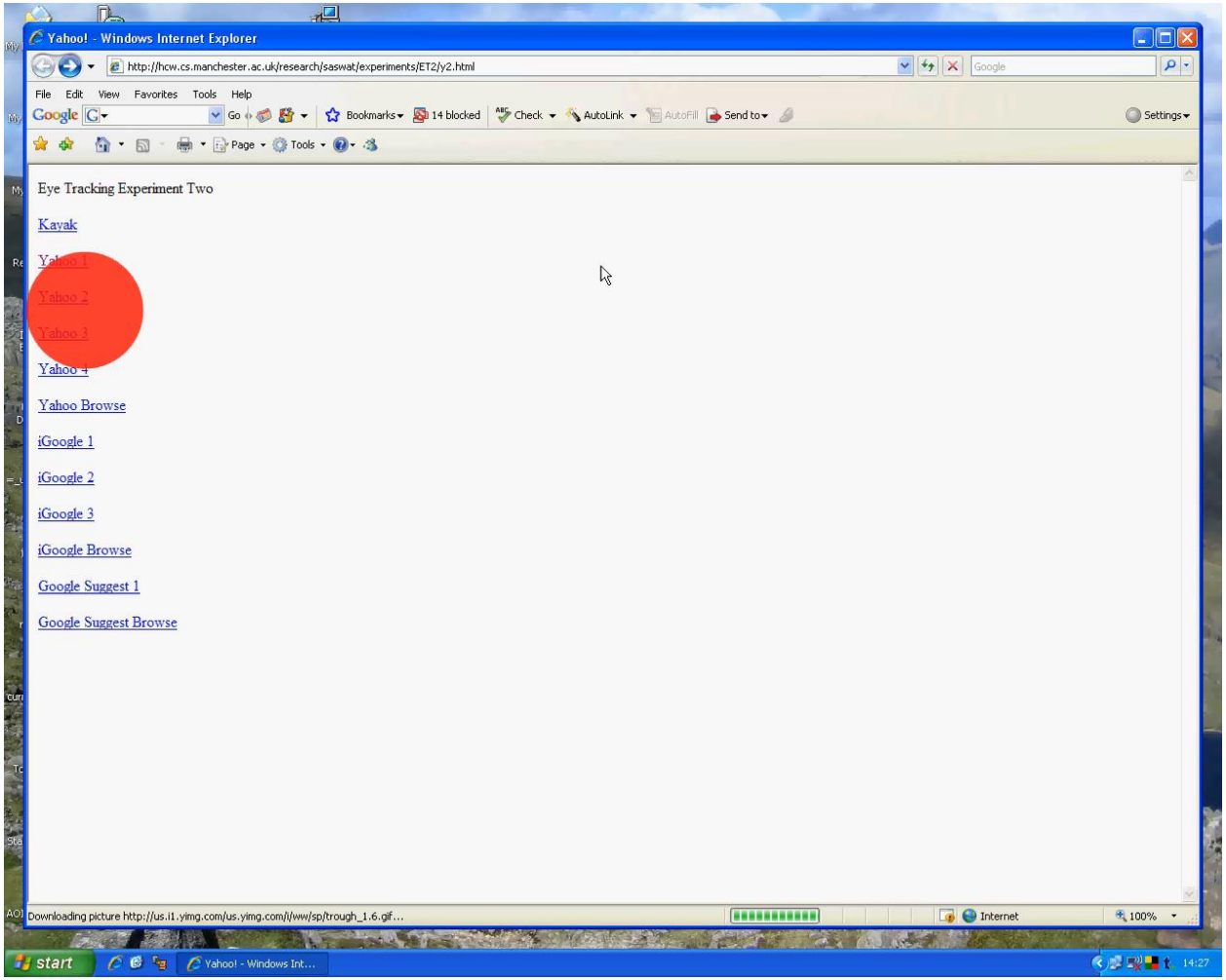
- Reusable solutions to recurring problems that occur during software development
  - Not necessarily the same code
  - Same structure
- Can be used in discussions among programmers who know them
  - Know that we're talking about the same thing
  - More effectively collaborate
- Mark Grand, "Patterns in Java"

Common patterns: Singleton and Factory

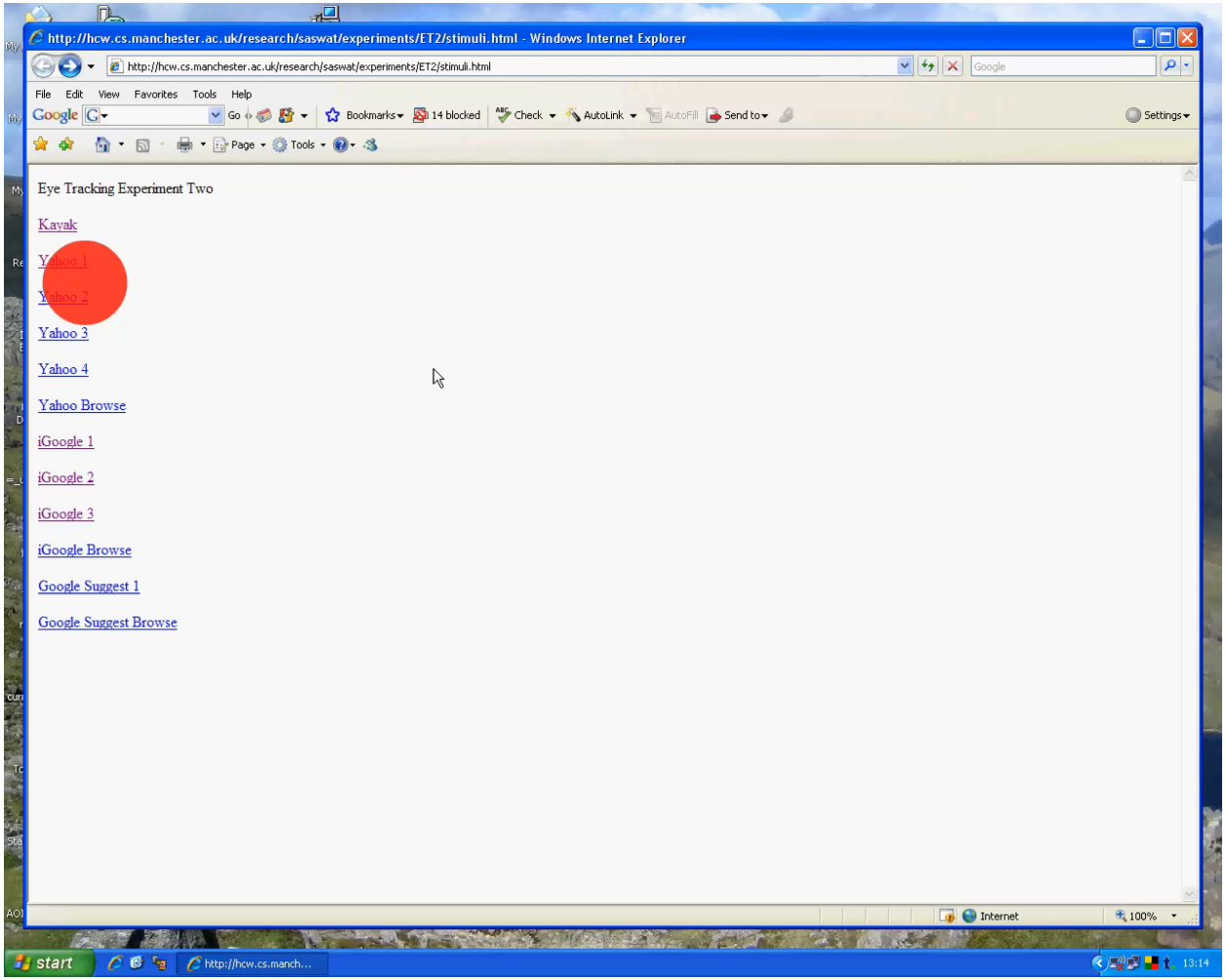
- Can you find any instances of the Singleton and Factory Method patterns in the Marauroa code?
  - Use Mark Grand Chapter 5 (in moodle) as a reference
    - Forces and Consequences
  - In which situations are they used?
  - Why?

- Can you spot any problems with some uses of Singleton in the code?
- How might use of the Factory pattern relate to testability?

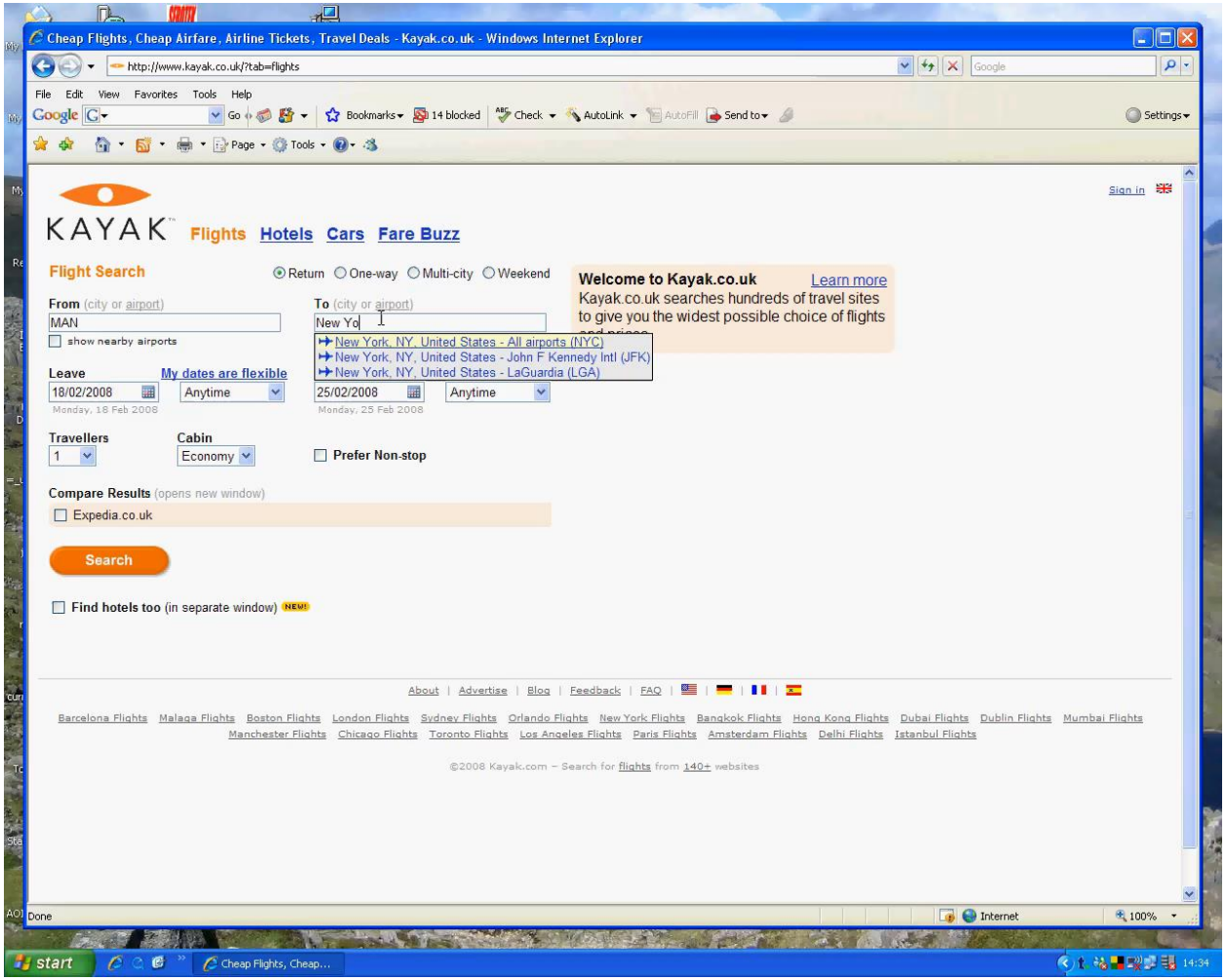
UI Patterns


Yahoo! - Windows Internet Explorer
http://hcw.cs.manchester.ac.uk/research/saswat/experiments/ET2/y2.html
Eye Tracking Experiment Two
Kayak
Yahoo 1
Yahoo 2
Yahoo 3
Yahoo 4
Yahoo Browse
iGoogle 1
iGoogle 2
iGoogle 3
iGoogle Browse
Google Suggest 1
Google Suggest Browse
Downloading picture http://us.i1.yimg.com/us.yimg.com/i/ww/sp/trough_1.6.gif...
start

UI Patterns


http://hcw.cs.manchester.ac.uk/research/saswat/experiments/ET2/stimuli.html - Windows Internet Explorer
http://hcw.cs.manchester.ac.uk/research/saswat/experiments/ET2/stimuli.html
Eye Tracking Experiment Two
Kayak
Yahoo 1
Yahoo 2
Yahoo 3
Yahoo 4
Yahoo Browse
iGoogle 1
iGoogle 2
iGoogle 3
iGoogle Browse
Google Suggest 1
Google Suggest Browse
start

UI Patterns


Cheap Flights, Cheap Airfare, Airline Tickets, Travel Deals - Kayak.co.uk - Windows Internet Explorer
http://www.kayak.co.uk/?tab=flights
KAYAK
Flights
Hotels
Cars
Fare Buzz
Sign in
Flight Search
Return
One-way
Multi-city
Weekend
From (city or airport)
MAN
show nearby airports
To (city or airport)
New Yo
New York, NY, United States - All airports (NYC)
New York, NY, United States - John F Kennedy Intl (JFK)
New York, NY, United States - LaGuardia (LGA)
Welcome to Kayak.co.uk
Learn more
Kayak.co.uk searches hundreds of travel sites to give you the widest possible choice of flights
Leave
My dates are flexible
18/02/2008
Anytime
25/02/2008
Anytime
Travellers
Cabin
Economy
Prefer Non-stop
Compare Results (opens new window)
Expedia.co.uk
Search
Find hotels too (in separate window)
About | Advertise | Blog | Feedback | FAQ
Barcelona Flights Malaga Flights Boston Flights London Flights Sydney Flights Orlando Flights New York Flights Bangkok Flights Hong Kong Flights Dubai Flights Dublin Flights Mumbai Flights
Manchester Flights Chicago Flights Toronto Flights Los Angeles Flights Paris Flights Amsterdam Flights Delhi Flights Istanbul Flights
©2008 Kayak.com – Search for flights from 140+ websites
Done
Internet
100%
start

Coding styles: Defensive programming

- *A blanket term for various practices to increase code stability once it is in production*
- Pros
  - "Defend against the impossible, because the impossible will happen"
  - Systems will not fail, but try as hard as possible to continue
    - Even in the face of unexpected user input or actions
  - Aids maintenance
    - Higher quality code: readable and comprehensible
- Cons
  - The impossible cannot happen so why defend against it?
    - New people join the team, new features, etc
  - Hurts maintenance
    - Hides bugs, bad for testability
  - Defensive coding often degenerates into paranoid programming

Defensive programming example

- No braces around `if`:

```
if (a == 1)¶
»   run();¶
```

- !!

```
if (a == 1)¶
»   init();¶
»   run();¶
```

- Set your IDE to add braces when you save?

```
if (a == 1) {¶
»   init();¶
}¶
run();¶
```

A note on `assert`

- `assert` is used to verify the correctness of an invariant in the code
- Should never trigger in production code
  - Turned off by default
  - Should be on during testing
  - Turn on with `-ea` option on the `java` command

- Do not use `assert` to check public method parameters!
  - Use Exceptions
- Do use `assert` to check post-conditions

A reasonable use of `assert`

```
public Foo acquireFoo(int id) {
  Foo result = null;
  if (id > 50) {
    result = fooService.read(id);
  } else {
    result = new Foo(id);
  }
  assert result != null;

  return result;
}
```

- http://stackoverflow.com/questions/2758224/what-does-the-java-assert-keyword-do-and-when-should-it-be-used

Paranoid programming vs testability

```
public House(Door door, Window window, Roof roof, List<Room> rooms) {¶
»   assert door != null;¶
»   assert window != null;¶
»   assert roof != null;¶
»   assert rooms != null;¶
»   assert rooms.size() != 0;¶

»   this.door = door;¶
»   this.window = window;¶
»   this.roof = roof;¶
»   this.rooms = rooms;¶
}¶

public void lockUp() {¶
»   door.lock();¶
»   window.lock();¶
}¶
```

- How do I test the `lockUp()` method?

Paranoid programming vs testability

```
@Test¶
public void testSecure() {¶
»   Door door = new Door();¶
»   Window window = new Window();¶

»   House house = new House(door, window, null, null);¶

»   house.lockUp();¶

»   assertTrue(door.isLocked());¶
»   assertTrue(window.isLocked());¶
}¶
```

- Only need a `Door` and `Window` to test `lockUp()`
- But the assertions will fail!

Coding styles: Offensive programming

- "*The best defence is a good offence*" – Anon
- Pros
  - Don't silently ignore errors, let them happen!
  - Force bugs to be detected and fixed as soon as possible
  - Simpler code (less error checking)
  - Errors shipped to the customer will be obvious
    - Easier to get management buy-in to fix them?
    - *No* results better than *wrong* results?
- Cons
  - Errors shipped to the customer will be obvious
    - Low quality more obvious
    - Errors visible to everyone
  - More crashes
    - Real-time life-critical systems won't try to recover

After the Easter break

- In the team study sessions you will continue to work on the coursework and you will have your second mentoring session
- In the workshop we will learn about making large scale changes to existing software safely