Victoria University
of Wellington, New Zealand
Te Whare Wananga o te
Upoko o te Ika a Maui
Aotearoa


# SWEN221 Software Development

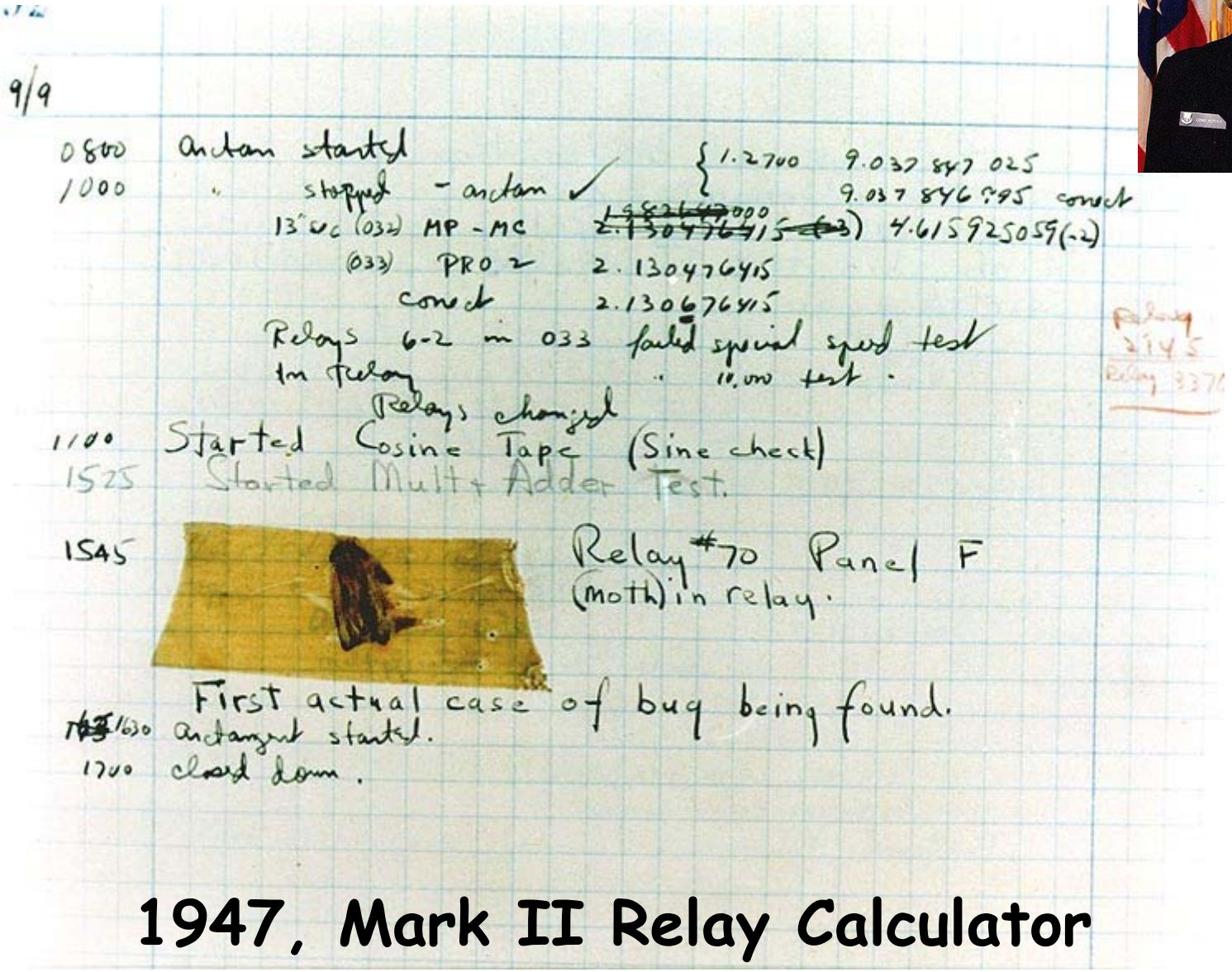
## Debugging

Thomas Kuehne

Victoria University

(slides modified from slides by David J. Pearce & Nicholas Cameron & James Noble & Petra Malik)

# Rear Admiral Grace Hopper



9/9

0800 Arctan started

1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 795 correct

13" UC (032) MP - MC ~~1.9826447000~~ ~~2.130476415 (-2)~~ 4.615925059(-2)

(033) PRO 2 2.130476415

correct 2.130676415

Relays 6-2 in 033 failed special speed test

in relay " 11.00 test.

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F (moth) in relay.

First actual case of bug being found.

1630 Arctangent started.

1700 closed down.

Relay 3145

Relay 3370

**1947, Mark II Relay Calculator**

# Debugging

- Bugs
  - programming puts them in
  - testing detects (**but not locates**) them

# Testing

# Debugging

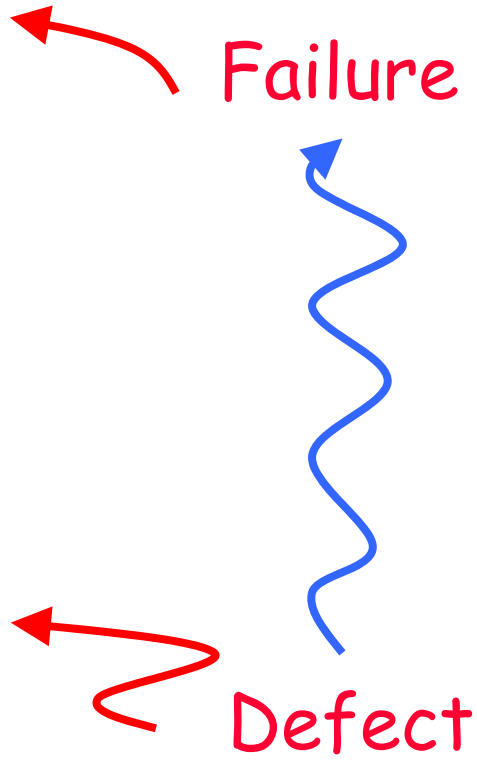
- Bugs
  - programming puts them in
  - testing detects (**but not locates**) them
  - debugging removes them
- Process of *finding* and *eliminating* bugs
  - often, locating the **defect** is hardest part
- Programmers spend more time debugging than writing new code

# Example

```
class BugManifestation {
 /**
  * @param input – should not be null
  */
 public static char[] convert(String input) {
  char[] cs = new char[input.length()];

  for(int i=0;i!=input.length();++i)
   cs[i] = input.charAt(i);

  return cs;
 }
 public static void main(String[] args) {
  String input = null;
  if(args.length > 0) { input = args[0]; }
  char[] bs = convert(input);
  …
}}
```



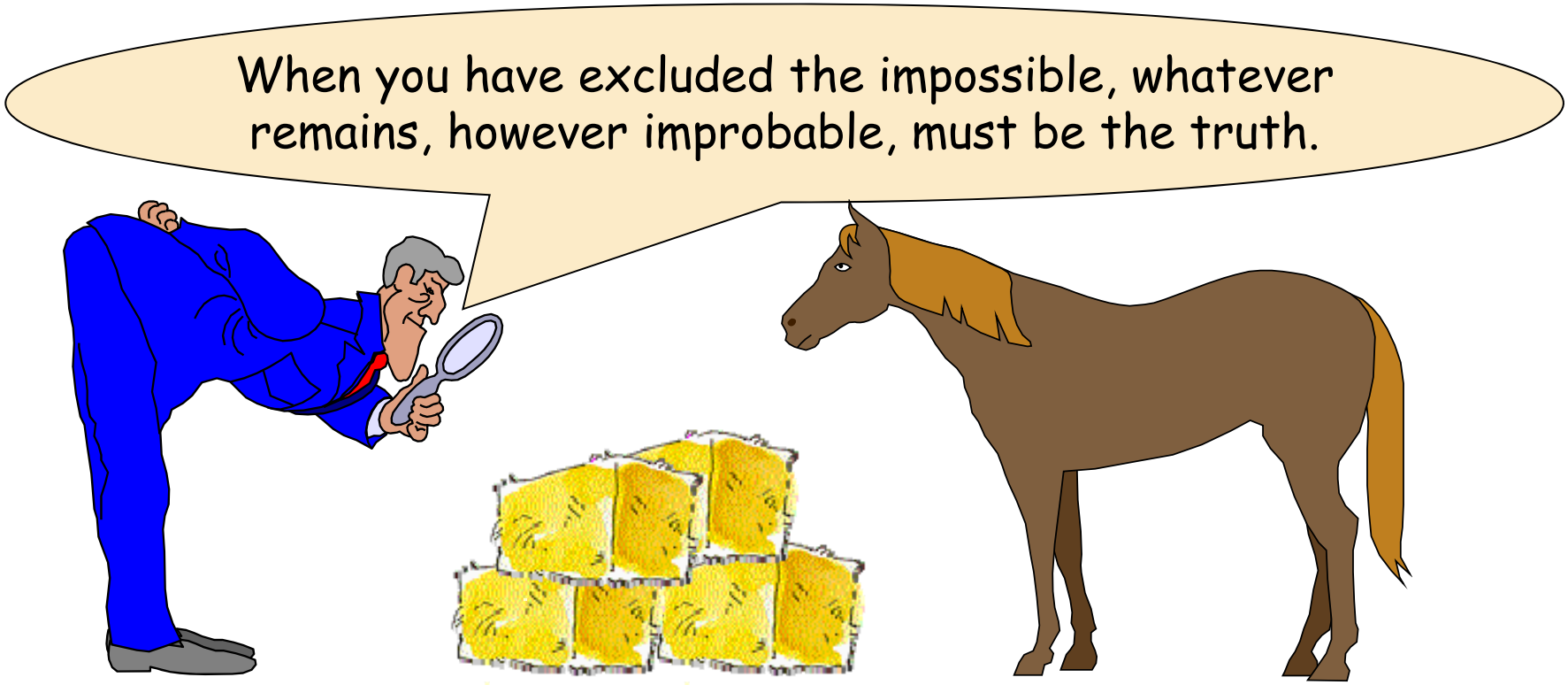
# Terminology

- **Defect** – Error manifest in code created by programmer
- **Infection** – Error manifest in program state
- **Propagation** – Bad program state leads to more bad states
- **Failure** – Program finally does something wrong

# Debugging Principles

- Basic approach to debugging
  - **Observe –** notice failure occurring
  - **Reproduce –** identify input(s) consistently resulting in failure
  - **Focus –** follow propagation trail
  - **Isolate –** identify defect
  - **Record –** add corresponding test case
  - **Correct –** fix the problem
  - **use deduction and experimentation**

# Debugging Principles



# Observing and Reproducing

- Observing a failure
  - Need to know what is right and wrong
  - a **NullPointerException** is typically undesired, but generally there is a need for an “oracle”.
  - Good test cases increase chance of observation

- Reproduction
  - Does a given input always cause an error?

```
public static void main(String[] args) {
 if(Calendar.getInstance().get(HOUR_OF_DAY) == 13) {
  // defect is in here
  …
 }
 // code continues here
 …
}
```

# Find + Isolate

- Focus on Defect
  - Can be a long and laborious task
  - Strategies:
    - Determine smallest input that causes the bug
    - Print debug information and/or use debugger
    - Form hypotheses and eliminate one by one
  - → Debugging requires considerable skill

- Isolate Problem
  - After zeroed in on **defect**, identify **problem**
  - What is at fault?
    - E.g. wrong method parameters (→ caller), incorrect algorithm, certain cases not handled at all, etc.

# Standard & Advanced Means

- Observe failed state
- Breakpoint
  - stop from where you want to start observing
- Watchpoint
  - stop with a condition
- Reversible debugging
  - roll back time
- Delta Debugging
  - automatically determine offending code

# Recording + Correcting

- Recording
  - Add appropriate test case to test suite
  - Then, can easily spot same or similar defect
  - Prevents reintroducing bugs (e.g., by fixing others) and not noticing
- Correcting
  - difficulty of fixing ranges from fixing typos to re-designing the architecture

# References


- David J. Agans: *Debugging: The Nine Indispensable Rules for Finding Even the Most Elusive Software and Hardware Problems*, AMACOM, 2002, ISBN 0814471684
- Andreas Zeller: *Why Programs Fail: A Guide to Systematic Debugging*, dpunkt.verlag, 2006 ISBN 3898642798
- Raimondas Lencevicius: *Advanced Debugging Methods*, Springer, 2000, ISBN 0792378954

# Don’t forget …

Labs start this week!