Week 12: Dealing with atrocious code

- Symptoms
- Causes
- Reactions
- Cures

We agreed (week 2) that this 1960's cliché is invalid

- and often harmful
- Who still believes it?
- How can it affect us?
- What should we do when we encounter it?

Any program that works is better than any program that doesn't work!

The limits of testing (validation)

- Testing, no matter how thorough, is never an after-the-fact cure for bad design or sloppy coding.
- The time to think about quality is a the beginning of a programming job.
- But what if someone else has already designed and coded the MUT? What about legacy code?

Impact of atrocious code

- It's extremely hard to test
- It's extremely hard to maintain or modify
- It's likely to contain non-obvious errors
  Why?
- Therefore, it's extremely expensive, time-consuming, and frustrating.
Some symptoms of atrocious (though possibly correct) code

- **Logic flow** is unnecessarily complicated.
- It’s full of unnecessary repetition.
- Functions are misplaced.
- **Names** are inappropriate, cryptic, or misleading.
  - of data
  - of modules (programs, classes, functions)
- **Commentary** is insufficient or misleading.
  - What’s common to all of those?
- Lack of understandability.

A computer program is two things

1. A **procedure** to be executed by a computer.
2. A **document** to be read and understood by a programmer.

We saw this before

```plaintext
IF CURRENT-MONTH = '01'
  MOVE +1 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '02'
  MOVE +2 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '03'
  MOVE +3 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '04'
  MOVE +4 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '05'
  MOVE +5 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '06'
  MOVE +6 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '07'
  MOVE +7 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '08'
  MOVE +8 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '09'
  MOVE +9 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '10'
  MOVE +10 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '11'
  MOVE +11 TO MONTH-NUMBER IN TRANSACTION
ELSE IF CURRENT-MONTH = '12'
  MOVE +12 TO MONTH-NUMBER IN TRANSACTION.
```

We found this code in a real (“senior”) programmer in a real organization! But it works, doesn’t it?

A slightly less obvious one

- We found that in a real application in a major insurance company.
- What was going through the programmer’s mind when he wrote that code?
- What should be done when such code is discovered by:
  - a maintenance programmer?
  - a professional tester?
  - a programming manager?
  - a quality-assurance reviewer?
Danger is more subtle

- Those two examples of unnecessary repetition were immediately obvious
  - The repeated pattern was short and the repetitions were close together.
- Many examples, however, are much less obvious.
  - The repetitions may be several pages apart or even in completely separate modules.
  - It is often the result of a mispaced function, i.e., performing some operation, transformation, activity, etc., in an inappropriate place.
- Maintenance programmers may fix a bug or implement an enhancement in one place and overlook the other places.

Quality and testing

- We noted (week 1) that there's much more to software quality than just testing

- But testing (validation) and internal quality are related.
  - A program that's poorly organized and hard to understand is likely to be extremely hard to validate.

- Should we bother even trying to test an atrocious program?
  - If not what should we do about it and when?

The myth of legacy code

- Current practitioners, including a surprising number of "experts" use that term to denote software developed before about 1985, assuming that:
  - The programmers didn't know about principles of good practice that we take for granted today.
  - They had to use crude programming languages and other tools that lacked the power of modern development.
  - The software they created is, therefore, of extremely poor quality and worth maintaining only temporarily and if it provides some essential function.
- This is invalid!
  - Starting with the structured revolution (ca. 1978) we understood almost all of the principles that apply today.
  - Much software of excellent quality was produced.

The solution

- Avoid writing atrocious programs in the first place
  - Build a staff of top (85th percentile) programmers
  - Do reviews and inspections of every deliverable
  - Enforce quality criteria on subcontractors

- Detect deviations early.
  - Provide remedial coaching to trainees
  - Show them good examples
  - Get rid of deadwood

  Can we do that?
Workshop exercise

- Specifications for an association membership system contain this logic:
  - For **regular** members,
    - charge $75.00 dues and
    - send (i.e. print labels for) the Journal and the Newsletter.
  - For **student** members,
    - charge $20.00 dues and
    - send only the Newsletter.
  - For **honorary** members,
    - don't charge dues, 
    - but send the Newsletter.
  - Count the number of members of each type, the total amount of dues by membership type, and the grand total of dues.

- Sketch a program fragment to implement this logic.

```java
for each member in roster
  if (mbr.type == regular)
    {dues = 75.00;
     sendJournal();
     sendNewsletter();
     ++regMbr.count;
     regMbrDues += dues;}
  else if (mbr.type == student)
    {dues = 20.00;
     sendNewsletter();
     ++stuMbrCount;
     stuMbrDues += dues;}
  else if (mbrType == honorary)
    {dues = 0.00;
     sendNewsletter();
     ++honMbrCount;}
  else throw noSuchMbrClass;

mbrCount = regMbrCount
   + stuMbrCount + honMbrCount;
```

Is this a reasonable sketch implementation?

- for each member in roster
  - if (mbr.type == regular)
    - {dues = 75.00;
      - sendJournal();
      - sendNewsletter();
      - ++regMbr.count;
      - regMbrDues += dues;}
  - else if (mbr.type == student)
    - {dues = 20.00;
      - sendNewsletter();
      - ++stuMbrCount;
      - stuMbrDues += dues;}
  - else if (mbrType == honorary)
    - {dues = 0.00;
      - sendNewsletter();
      - ++honMbrCount;}
  - else throw noSuchMbrClass;

mbrCount = regMbrCount
   + stuMbrCount + honMbrCount;

How can we improve it?