Week 11 Lecture 1

Fakes

Testing Requirements

- The sooner you find an error the easier it is to fix it.
  - That is why we only write a line or two before compiling.
- The faster you can test a system the more frequently you can test it.
- You need to completely control a system to test elements that depend on it.

Testing Speed

- If tests take a couple of second, like compilation, we can test frequently.
- Accessing a database or a network may take a long time.

Testing Control

- You may not have complete control over a database or network.
  - Other may be using it.
  - It may be costly to reset it.

Fakes provide speed and control

- Fakes are programs that mimic those we may not have complete control over such as databases or networks.
- The interface must be the same
  - Same functions, with the same parameters and return values.

Example

- The array database we have written can serve as a fake for the file database we have written.
- It is easy to reset and fast.
- The real database is more cumbersome.

Database Fake

- To create a fake we need to define the same functions that the real element would use.
- Here we are defining a fake database so we need to define the CRUD functions.
- We can then incorporate the fake database into the SUT by linking in the new file.

Database Interface (1)

- int add(attend_rec r);
  - Create a record to the database
- attend_rec find(int sno);
  - Read a record for student sno
- int modify(attend_rec r);
  - Update a record
- int delete(int sno);
  - Delete record sno

Database Interface (2)

- db: sequence of records
  - Array in fake
  - File system in real system
- student record is identical
  - s_no: unique number
  - name: string
  - grade: int

Database Interface (3)

- Function calls are identical
  - int add(attend_rec r);
  - attend_rec find(int sno);
  - int modify(attend_rec r);
  - int delete(int sno);