COMP110: Principles of Computing

Test Driven Development

Learning outcomes

By the end of this session, you will...

- asdf

Test driven development

Unit testing

- A **unit test** or **test case** is a piece of code that verifies a unit (e.g. a function or class) of a program

Unit testing

- A **unit test** or **test case** is a piece of code that verifies a unit (e.g. a function or class) of a program
- E.g. verifies that a function called with a particular set of parameters returns the expected result

Unit testing

- A **unit test** or **test case** is a piece of code that verifies a unit (e.g. a function or class) of a program
- E.g. verifies that a function called with a particular set of parameters returns the expected result
- The following might be unit tests for a `factorial` function:
  - `factorial(1)== 1`
  - `factorial(2)== 2`
  - `factorial(3)== 6`
  - `factorial(4)== 24`

Why do unit testing?

- Can find problems that normal testing misses

Why do unit testing?

- Can find problems that normal testing misses
- **Bottom-up** testing — if the **parts** work properly, it's easier to make the **whole** work properly

Why do unit testing?

- Can find problems that normal testing misses
- **Bottom-up** testing — if the **parts** work properly, it's easier to make the **whole** work properly
- When code is **changed**, can verify that nothing was broken

Caveats

- Have to spend time writing tests

FALMOUTH UNIVERSITY

Caveats

- Have to spend time writing tests
    - Not really a drawback — good unit tests will probably **save** more time in debugging than it takes to write them

Caveats

- Have to spend time writing tests
    - Not really a drawback — good unit tests will probably **save** more time in debugging than it takes to write them
- Can give a false sense of security

Caveats

- Have to spend time writing tests
  - Not really a drawback — good unit tests will probably **save** more time in debugging than it takes to write them
- Can give a false sense of security
  - Unit tests can't cover 100% of a complex program — **not a substitute** for other forms of testing

Test driven development (TDD)

- A development process that advocates writing the unit tests **first**

Test driven development (TDD)

- A development process that advocates writing the unit tests **first**
- Repeat the following three steps:

Test driven development (TDD)

- A development process that advocates writing the unit tests **first**
- Repeat the following three steps:
    1. **Red**: create a new test case, which should initially **fail**

Test driven development (TDD)

- A development process that advocates writing the unit tests **first**
- Repeat the following three steps:
  1. **Red**: create a new test case, which should initially **fail**
  2. **Green**: write code to make the new test **succeed** (without causing the other test cases to fail)

Test driven development (TDD)

- A development process that advocates writing the unit tests **first**
- Repeat the following three steps:
  1. **Red**: create a new test case, which should initially **fail**
  2. **Green**: write code to make the new test **succeed** (without causing the other test cases to fail)
  3. **Refactor**: **improve** the code, ensuring that all tests still **succeed**

Why TDD?

- All the benefits of **unit testing**, plus...

Why TDD?

- All the benefits of **unit testing**, plus...
- Often easier to convert a **user story** into test cases rather than directly into code

Why TDD?

- All the benefits of **unit testing**, plus...
- Often easier to convert a **user story** into test cases rather than directly into code
- Writing the bare minimum of code to make the test "green" lets you **focus on user stories**, not on **over-generalisation** or **non-essential functionality**

FALMOUTH
UNIVERSITY

Why TDD?

- All the benefits of **unit testing**, plus...
- Often easier to convert a **user story** into test cases rather than directly into code
- Writing the bare minimum of code to make the test “green” lets you **focus on user stories**, not on **over-generalisation** or **non-essential functionality**
  - **KISS**: Keep It Simple, Stupid
  - **YAGNI**: You Aren’t Gonna Need It

Red

- Create a new test case, which should initially **fail**

Red

- Create a new test case, which should initially **fail**
- Write only enough code to allow the test case to compile and run, e.g. write a **stub** function

Red

- Create a new test case, which should initially **fail**
- Write only enough code to allow the test case to compile and run, e.g. write a **stub** function
- What if the test succeeds?

Red

- Create a new test case, which should initially **fail**
- Write only enough code to allow the test case to compile and run, e.g. write a **stub** function
- What if the test succeeds?
  - Maybe you already implemented that feature?

Red

- Create a new test case, which should initially **fail**
- Write only enough code to allow the test case to compile and run, e.g. write a **stub** function
- What if the test succeeds?
  - Maybe you already implemented that feature?
  - Maybe the test case is wrong?

Red

- Create a new test case, which should initially **fail**
- Write only enough code to allow the test case to compile and run, e.g. write a **stub** function
- What if the test succeeds?
  - Maybe you already implemented that feature?
  - Maybe the test case is wrong?
  - Maybe your unit testing code is broken?

Green

- Add the **bare minimum** of code to make the new test case succeed

Green

- Add the **bare minimum** of code to make the new test case succeed
    - **K**eep **I**t **S**imple, **S**tupid!

Green

- Add the **bare minimum** of code to make the new test case succeed
  - **K**eep **I**t **S**imple, **S**tupid!
- Verify that **all** unit tests now succeed

Green

- Add the **bare minimum** of code to make the new test case succeed
  - **K**eep **I**t **S**imple, **S**tupid!
- Verify that **all** unit tests now succeed
- What if old tests now fail?

Green

- Add the **bare minimum** of code to make the new test case succeed
  - **K**eep **I**t **S**imple, **S**tupid!
- Verify that **all** unit tests now succeed
- What if old tests now fail?
  - Fix it

Green

- Add the **bare minimum** of code to make the new test case succeed
  - **K**eep **I**t **S**imple, **S**tupid!
- Verify that **all** unit tests now succeed
- What if old tests now fail?
  - Fix it
  - **Or** revert and start again — can be faster than debugging

Green

- Add the **bare minimum** of code to make the new test case succeed
  - **K**eep **I**t **S**imple, **S**tupid!
- Verify that **all** unit tests now succeed
- What if old tests now fail?
  - Fix it
  - **Or** revert and start again — can be faster than debugging
  - (you **did** commit before you started, right?)

Refactor

- E.g. remove duplication, improve names, add documentation, apply design patterns, ...

Refactor

- E.g. remove duplication, improve names, add documentation, apply design patterns, ...
- To generalise or not to generalise?

Refactor

- E.g. remove duplication, improve names, add documentation, apply design patterns, ...
- To generalise or not to generalise?
- **Do** generalise if it makes the code **simpler**

FALMOUTH UNIVERSITY

Refactor

- E.g. remove duplication, improve names, add documentation, apply design patterns, ...
- To generalise or not to generalise?
- **Do** generalise if it makes the code **simpler**
- **Don't** generalise because you "might" need it later

Refactor

- E.g. remove duplication, improve names, add documentation, apply design patterns, ...
- To generalise or not to generalise?
- **Do** generalise if it makes the code **simpler**
- **Don’t** generalise because you “might” need it later
  - **Y**ou **A**ren’t **G**onna **N**eed **I**t!
  - Wait until it **is** needed in another cycle

Refactor

- E.g. remove duplication, improve names, add documentation, apply design patterns, ...
- To generalise or not to generalise?
- **Do** generalise if it makes the code **simpler**
- **Don't** generalise because you "might" need it later
  - You **A**ren't **G**onna **N**eed **I**t!
  - Wait until it **is** needed in another cycle
- Verify that **all** unit tests still succeed