Testing Software

Charles Carter

September 7, 2017

Table of Contents

Purposes of testing

- **Verification and Validation**

Purposes of testing

- **Verification and Validation**
- **Verification** building the thing right, i.e., does it contain any defects?

Purposes of testing

- **Verification and Validation**
- **Verification** building the thing right, i.e., does it contain any defects?
- **Validation** building the right thing, i.e., does it do what it is supposed to do?

Black/White Box Testing

- **Black Box and White Box**

Black/White Box Testing

- **Black Box and White Box**
- **Black Box** testing without knowledge of the program, that is, looking only at inputs and outputs

Black/White Box Testing

- **Black Box and White Box**
- **Black Box** testing without knowledge of the program, that is, looking only at inputs and outputs
- **White Box** testing with knowledge of the program, that is, looking at state and flow of control

Kinds of tests

- **Unit, Integration, Regression, and Acceptance Tests**

Kinds of tests

- **Unit, Integration, Regression, and Acceptance Tests**
- **Unit tests** tests each individual class, package, library, or module as a stand-along program

Kinds of tests

- **Unit, Integration, Regression, and Acceptance Tests**
- **Unit tests** tests each individual class, package, library, or module as a stand-along program
- **Integration tests** tests combinations of classes, packages, libraries, or modules as a complete application

Kinds of tests

- **Unit, Integration, Regression, and Acceptance Tests**
- **Unit tests** tests each individual class, package, library, or module as a stand-along program
- **Integration tests** tests combinations of classes, packages, libraries, or modules as a complete application
- **Regression tests** tests current modifications against previously releases in order to verify that past releases are not broken

Kinds of tests

- **Unit, Integration, Regression, and Acceptance Tests**
- **Unit tests** tests each individual class, package, library, or module as a stand-along program
- **Integration tests** tests combinations of classes, packages, libraries, or modules as a complete application
- **Regression tests** tests current modifications against previously releases in order to verify that past releases are not broken
- **Acceptance tests** test whether the user find the software acceptable for the intended purposes

Object Oriented Testing — Practices

Test all classes individually

- Create a public, static main class as an entry point to the application

Object Oriented Testing — Practices

Test all classes individually

- Create a public, static main class as an entry point to the application
- In the main class, instantiate objects of all classes

Object Oriented Testing — Practices

Test all classes individually

- Create a public, static main class as an entry point to the application
- In the main class, instantiate objects of all classes
- For each object, exercise every method (including overloaded methods)

Object Oriented Testing — Practices

Test all classes individually

- Create a public, static main class as an entry point to the application
- In the main class, instantiate objects of all classes
- For each object, exercise every method (including overloaded methods)
- Test all return values against their "correct" values

Object Oriented Testing — Practices

Test all classes individually

- Create a public, static main class as an entry point to the application
- In the main class, instantiate objects of all classes
- For each object, exercise every method (including overloaded methods)
- Test all return values against their "correct" values
- How do you test VOID methods?

Procedural Testing — Practices

Test all modules individually

- Create a script as an entry point to the application

Procedural Testing — Practices

Test all modules individually

- Create a script as an entry point to the application
- In the script file, import all the modules (libraries, modules)

Procedural Testing — Practices

Test all modules individually

- Create a script as an entry point to the application
- In the script file, import all the modules (libraries, modules)
- For each module, exercise all subprocedures and fuctions

Procedural Testing — Practices

Test all modules individually

- Create a script as an entry point to the application
- In the script file, import all the modules (libraries, modules)
- For each module, exercise all subprocedures and fuctions
- Test all return values against their “correct” values

Procedural Testing — Practices

Test all modules individually

- Create a script as an entry point to the application
- In the script file, import all the modules (libraries, modules)
- For each module, exercise all subprocedures and fuctions
- Test all return values against their “correct” values
- How do you test subprocedures (named blocks called for their side effects)?

Afterword

Software quality assurance (SQA) is as important as implementing software, if not more so. This presentation briefly covers only unit testing. Study of testing (as a software discipline) is lengthy and arduous.