6 Testing and Debugging

Main concepts to be covered

- Testing
- Debugging
- Test automation
- Manual methods

We have to deal with errors

- Early errors are usually *syntax errors*.
  - The compiler will spot these.
- Later errors are usually *logical errors*.
  - The compiler cannot help with these.
  - Also known as bugs.
- Some logical errors have no immediately obvious manifestation.
  - The may show up only under special conditions.

Prevention vs Detection
(Developer vs Maintainer)

- We can lessen the likelihood of errors.
  - Use software engineering techniques, like encapsulation, modularization, design patterns
- We can improve the chances of detection.
  - Use software engineering practices, like documentation and testing.
- We can develop detection skills.

Testing vs verification

- Testing can prove the *presence* of errors but *not* the *absence* of errors (in general).
- Verification can prove the absence of errors, but
  - Demands formal analysis (proof).
  - Harder to automate than testing.

Testing and debugging

- Testing searches for the presence of errors.
- Debugging searches for the source of errors.
  - The manifestation of an error may well occur some ‘distance’ from its source.

Testing and debugging techniques

- Unit testing (within BlueJ and eclipse)
- Test automation
- Regression testing
- Manual walkthroughs
- Print statements
- Debuggers

Unit testing

- Each unit of an application may be tested.
  - Method, class, module (package in Java).
- Can (should) be done during development.
  - Finding and fixing early lowers development costs.
  - A test suite should be constructed in advance
    - A test suite is an important complement to the specification of a software component.
    - Eg. a TCK (=Technology Compatibility Kit) for a JSR (= Java Specification Request) contains collections of test cases that help developers to check that a component conforms to the JSR.

Testing fundamentals

- Understand what the unit should do – its *contract*
  - Look for violations.
  - Use positive tests and negative tests.
- Test *boundaries*
  - Zero, One, Full.
    - Search an empty collection.
    - Add to a full collection.

Unit testing within BlueJ

- Objects of individual classes can be created.
- Individual methods can be invoked.
- Inspectors provide an up-to-date view of an object's state.
- This is a kind of simple *white-box* testing.

Test automation

- Good testing is a creative process, but ...
- ... thorough testing is time consuming and repetitive.
- *Regression testing* involves re-running tests.
- Use of a *test rig* or *test harness* can relieve some of the burden.
  - Classes are written to perform the testing.
  - Creativity focused in creating these.
  - Automatic test runs.

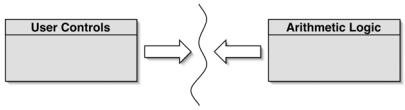
Test automation (2)

- Explore through the *sorted-bufferX* projects.
  - sorted-buffer1
    - Empty test class (develop no 2 from this).
  - sorted-buffer2
    - Manually written testcases.
  - sorted-buffer3
    - Uses a *test fixture*.

Modularization and interfaces

- Applications often consist of different modules.
  - E.g. so that different teams can work on them.
- The *interface* between modules must be clearly specified.
  - Supports independent concurrent development.
  - Increases the likelihood of successful integration.

Modularization in a calculator



- Each module does not need to know implementation details of the other.
  - User controls could be a GUI or a hardware device.
  - Logic could be hardware or software.

Debugging

- It is important to develop code-reading skills.
  - Debugging will often be performed on others' code.
- Techniques and tools exist to support the debugging process.

Manual walkthroughs

- Relatively underused.
  - A low-tech approach.
  - More powerful than appreciated.
- Get away from the computer!
- 'Run' a program by hand.
- High-level (Step) or low-level (Step into) views.

Tabulating object state

- An object's behavior is usually determined by its state.
- Incorrect behavior is often the result of incorrect state.
- Tabulate the values of all fields.
- Document state changes after each method call.

Verbal walkthroughs

- Explain to someone else what the code is doing.
  - They might spot the error.
  - The process of explaining might help you to spot it for yourself.
- Group-based processes exist for conducting formal walkthroughs or *inspections*. (XP, Agile,…)

Print statements

- The most popular technique.
- No special tools required.
- All programming languages support them.
- Only effective if the right methods are documented.
- Output may be voluminous!

Choosing a test strategy

- Be aware of the available strategies.
- Choose strategies appropriate to the point of development.
- Automate whenever possible.
  - Reduces tedium.
  - Reduces human error.
  - Makes (re)testing more likely.

Debuggers

- Debuggers are both language- and environment-specific.
  - BlueJ has an integrated debugger.
- Support breakpoints.
- Step and Step-into controlled execution.
- Call sequence (stack).
- Object state.

Review

- Errors are a fact of life in programs.
- Good software engineering techniques can reduce their occurrence.
- Testing and debugging skills are essential.
- Make testing a habit.
- Automate testing where possible.
- Practise a range of debugging skills.