Objectives

- JUnit
- Coverage
- Collaboration

Review

- Describe the general testing process
- What is a set of test cases called?
- What is *unit testing*?
- What are the benefits of unit testing?
- What are the characteristics of good unit tests?
- What are the steps in a JUnit Test Case?
  - How do we implement those steps?
- What is test-driven development?

JUnit Review

- Put JUnit classes in a separate `tests` package
- Testing process:
  1. Set up – objects for testing
  2. Execute code
  3. Verify output
  4. Tear down

Tests are Methods

- Mark your testing method with `@Test`
  - From `org.junit.Test`

```
public class CalculatorTest {

    @Test
    public void addTest() {
        …
    }
}
```

Class for testing the Calculator class

A method to test the "add" functionality

- Convention: Method name describes what you're testing

Assert Methods

- Variety of assert methods available
- If fail, throw an exception
- Otherwise, test keeps executing
- All `static void`
- Example:
  `assertEquals(Object expected, Object actual)`

```
@Test
public void addTest() {
     …
     assertEquals(4, calculator.add(3, 1));
}
```

Set Up/Tear Down

- May want methods to set up objects for every test in the class
  - Called **fixtures**
  - If have multiple, no guarantees for order executed

```
@Before
public void prepareTestData() { ... }

@Before
public void setupMocks() { ... }

@After
public void cleanupTestData() { ... }
```

Executed before **each** test method

Example Set Up Method

```
private CD testCD;

@Before
public void setUp() {
     testCD = new CD("CD title", 100, 1997,
                     "CD Artist", 11);
}
```

`@Before` Executed before **each** test method
Can use `testCD` in test methods

Expecting an Exception

- Handling Error Cases
  - Sometimes an exception *is* the expected result

Add an "expected" attribute:

```
@Test(expected=IndexOutOfBoundsException.class)
public void testIndexOutOfBoundsException() {
   ArrayList emptyList = new ArrayList();
   Object o = emptyList.get(0);
}
```

Test case passes iff exception thrown

JUNIT IN ECLIPSE

Using JUnit in Eclipse

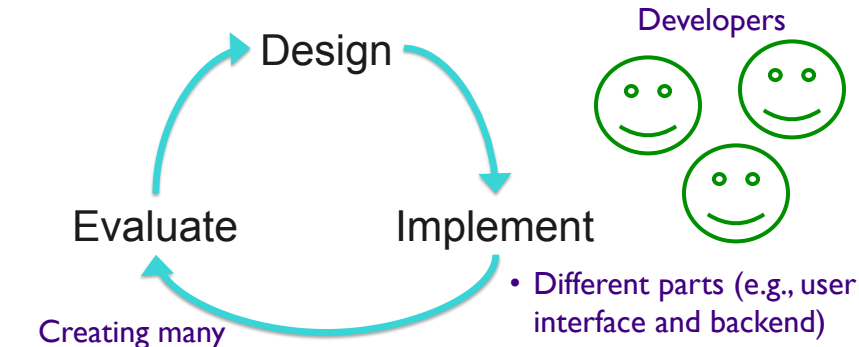
- Eclipse can help make our job easier
  - Automatically execute tests (i.e., methods)
  - We can focus on coming up with tests

Unit Testing & JUnit Summary

- Unit Testing: testing smallest component of your code
  - For us: class and its methods
- JUnit provides framework to write test cases and run test cases automatically
  - Easy to run again after code changes
- JUnit Resources available from Course Page's "Resource" Link, under Java
  - API
  - Tutorials

VERSION CONTROL

Problems in Collaborating on Code



Creating many prototypes
- What if don't like recent prototype? Need to go back to older version

- Different parts (e.g., user interface and backend)
- > 1 developer implementing concurrently
  - *What if one introduces a bug?*

Version Control Features

- Synchronization
  - Lets people share files
  - Stay up-to-date with the latest version
- Backup and Restore
  - Files are saved as they are edited
  - Revert to a specific version/checkpoint
- Track changes to code
  - Save comments explaining why change happened
  - Stored in the VCS, not the file
  - Track how, why a file evolves over time
- Track ownership
  - Tags every change with the name of the person who made it

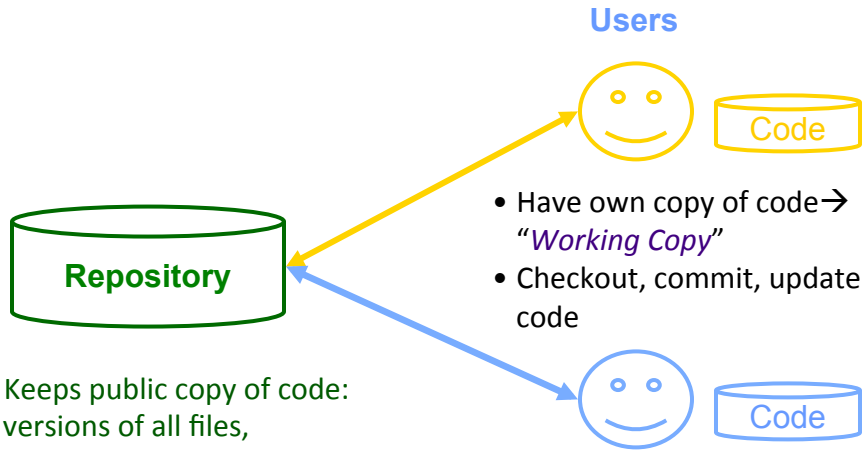
Version Control Features

- Short-term undo
  - Messed up a file? Go back to the last good version
- Long-term undo
  - Created a bug a year ago? Jump back to see change you made.
- Sandboxing
  - Making a big change? Make temporary changes in isolated area, test, work out kinks before "checking in" your changes
- Branching and merging
  - Branch a copy of your code into a separate area, modify it in isolation (tracking changes separately)
  - Later, merge work into common area

Version Control Systems

- Popular Version Control Systems
  - CVS, Subversion, Git, …
- Terms used are common for most version control systems
- We will use Subversion with Subclipse
  - Mark Phippard, a W&L grad works on both
    - Chief Architect at CollabNet, the company that founded Subversion
    - Subclipse lead

Using Version Control

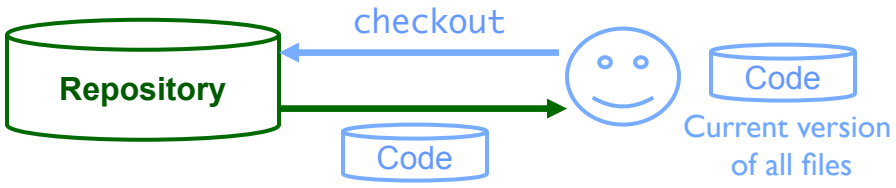


- Have own copy of code→ "*Working Copy*"
- Checkout, commit, update code

- Keeps public copy of code: versions of all files, comments about changes, who made changes

Using Version Control: checkout

- To start, need to **checkout** your working copy of the code



Using Version Control: `commit`

- After you make changes that you *want others to see*, `commit` your version
  - Include comments about what changes you made and why

- Checks for conflicts
- Updates each modified file
- Records comments with updated files

Using Version Control: `commit`

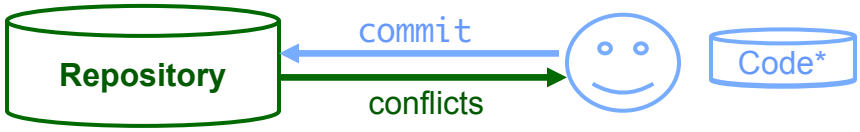
- After you make changes that you *want others to see*, `commit` your version
  - Include comments about what changes you made and why

- Checks for conflicts
- Updates each modified file
- Records comments with updated files

Other people's code doesn't change

Using Version Control: `commit`

- After you make changes that you *want others to see*, `commit` your version



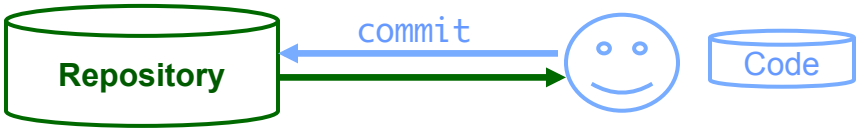
- Checks for conflicts: code conflicts with recent changes in the public copy
- Update code, fix conflicts
- Try commit again

Using Version Control: `update`

- To see the *current* version of the code, `update` your repository
  - Resolve conflicts

Using Version Control: `add, delete`

- You need to **`add`** and **`delete`** files and directories to the repository, then **`commit`**



- Create new records for added files
- Close records for deleted files
  - Files not deleted from repository

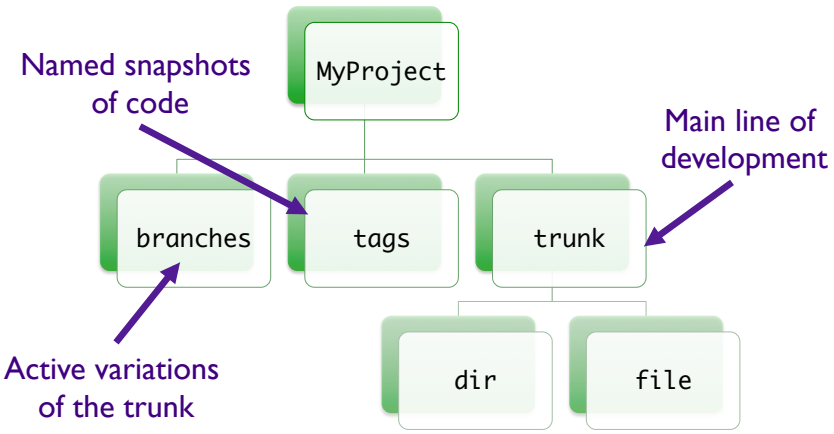
- Add, delete files and directories
- Commit

Version Control Advice

- Does not eliminate need for communication
  - Process becomes much more difficult if developers do not communicate
- Before picking up again, **update** your working copy
- **Commit** only after you've tested code and you're fairly sure it works
  - Write descriptive comments so your team members know what you did and why

Code Organization

- Organize code into appropriate structure



SUBCLIPSE

Subclipse

- Plugin for Eclipse
- Installation instructions online
  - Includes Configuration instructions too

Subversion Practice due by end of day today

SVN Rules

- Communicate with your teammates
- Describe your committed changes in comments
  - Let's your teammates know what you're doing—what changes to look for
- Don't commit the `bin` directory/class files
- Don't commit files that are specific to your account
  - .classpath, .project, .settings

Project 1: Testing Practice

- Given: a `Car` class that only has enough code to compile
- Your job: Create a ***good*** set of test cases that ***thoroughly***/***effectively*** test `Car` class
  - Find faults in my faulty version of `Car` class
  - Start: look at code, think about how to test, set up JUnit tests
  - Written analysis of process
- Due next Wednesday
  - Wed, Fri – work on the project