Testing Tips

Testing Your Code

- Unit Test
  - Test on an individual class
  - Write code to exercise each method.
- System Test
  - Test on the system as a whole
  - Provide input – check output

- try will run unit test on each class submitted.
- try will also run system test on the last submission.

Unit Testing

- Good practice to place the unit test of a class in the `main(String args[])` for that class
  - `java myClass` to test

  - We will use the Directory class as an example.

Unit Test

- The unit test is to exercise each function of your class to assure that it does what it advertises it should do.
  - `toString()` – Be sure to write a toString for every class so that you can easily print out the current state of an object.

Unit Test

- Accessor functions
  - Returns individual attributes about an object
    - getFullName()
    - getSize()

  - Tip:
    - Use these functions in your implementation of `toString()`

Unit Test

```
public String toString ()
{
    return getFullName () + “|” + getSize();
}
```

Unit Test

- Once toString() is in place, use it to assure that the object “looks” the way it should after each method.
  - I.e. Test Constructor

```
Directory D = new Directory(“foo”,
  true)
System.out.println (D.toString());
```

foo/|0

Unit Test

- You may wish to also check after placing the Directory into another directory

```
Directory D = new
  Directory(“foo”, true);
Directory B = = new
  Directory(“bar”, false);
B.addEntry (D);
System.out.println
  (D.toString());
```

bar/foo/|0

Unit Test

- Automated testing
  - Check returned value rather than printing.
    - Print error only if something is wrong

```
Directory D = new Directory(“foo”, true);
if (!D.toString().equals (“foo/|0”))
  System.err.println (“Problem with default
  constructor”)
```

Unit Test

- Testing Methods
  - Test for success
    - test cases where things go right
  - Test for failure
    - test cases when things go wrong!
- E.g. removeEntry()
    - Try removing item in directory
    - Try removing item not in directory
    - Try removing when directory is read-only.
    - Try removing when directory is empty

Unit Test

- Testing Methods
  - Testing visit.
    - Need a EntryProcessor

Unit Test

```
class TestProcessor implements EntryProcessor {

    public TestProcessor() {}
    void process (Entry item)  {
        System.out.println (item.getFullName());
    }
}
```

Unit Tests

- Methods that return values
  - Call the method and make sure they return the correct value.
    - Check several possible values
    - I.e. findEntry
      - Run on item in directory
      - Run on item not in directory
      - Run on an empty directory

Unit Testing

- Questions

System Testing

- Tests system as a whole
  - Run system with different input
  - Check output produced by system

  - If unit tests fail, system tests will probably fail as well.

Testing your work

- You can use try to test out your classes
  - `try cs2-grd project1-test infile`
    - Will run our solution on your test data in file infile
    - You can redirect the output into a file and then compare with your output
      - `try cs2-grd project1-test infile > correctSolution`
      - `java VFS < infile > mySolution`
      - `diff mySolution correctSolution`

Summary

- Testing
  - Unit testing
  - System testing

- A bit extra work but well worth it in the long run