

CS1632 Lecture 11

Writing testable code
Bill Laboon/Dustin Iser

What is testable code?

Code which can be easily tested in an automated fashion at multiple levels of abstraction.



Two key concepts

1. Ensure that events are repeatable.
2. Ensure code is segmented.



Ensure that events are repeatable

Things that make tests unrepeatable:

- Databases
- File systems
- Network connections
- 3rd party software
- Random number generators



Dependency Injection

Passing dependencies of a method in as parameters, as opposed to having them be hard-coded. This helps with testing as they can easily be replaced with test doubles or fakes.

See Lecture 11 example on GitHub.



Polymorphism

Polymorphism is the ability of an object to take on many forms.

See Lecture 11 example on GitHub.



Segmented code


Methods should do one thing and do it well.

```
// Bad
public int getNumMonkeysAndSetDatabase(Database d) {
    if (d != null) {
        _database = d;
    } else {
        _database = DEFAULT_DATABASE;
    }
    setDefaultDatabase(_database);
    int numMonkeys = getNormalizedMonkeys();
    if (numMonkeys < 0) {
        numMonkeys = 0;
    }
    return numMonkeys;
}
```

Segmented code

```
// Better
public void setDatabase(Database d) {
    if (d != null) {
        _database = d;
    } else {
        _database = DEFAULT_DATABASE;
    }
    setDefaultDatabase(_database);
}

public int getNumMonkeys() {
    int numMonkeys = getNormalizedMonkeys();
    if (numMonkeys < 0) {
        numMonkeys = 0;
    }
    return numMonkeys;
}
```



Give yourself something to test

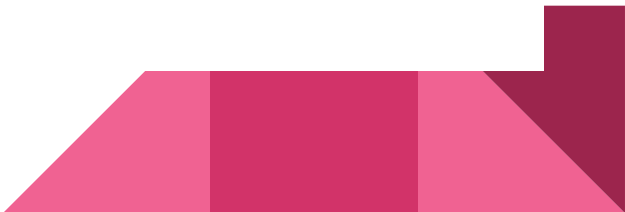
If you have something to assert, testing is much easier.

- Return values
- Thrown exceptions
- Modified accessible attributes

```
public void addMonkey(Monkey m) {  
    if (m != null) {  
        addMonkeyToMonkeyList(m);  
    }  
}
```

Give yourself something to test

```
// Better
public int addMonkey(Monkey m) throws NullMonkeyException {
    int toReturn = -1;
    if (m != null) {
        toReturn = addMonkeyToMonkeyList(m);
    } else {
        throw NullMonkeyException();
    }
    return toReturn;
}
```



DRY - Don't Repeat Yourself

- Don't copy and paste code from one section of your program to another.
- Don't have multiple methods with the same or similar functionality.
- Use generics and polymorphism to extend the same logic to objects of different types

Why?

When you fix a bug or add a feature to code that has been copy and pasted, you must now find all the places that code has been pasted and fix that bug or add that feature in those places as well.



Use the dominant paradigm of the language

Java was designed with object orientation in mind so use it in an object-oriented way.

Nothing will stop you from writing Java using a different paradigm, but you will be swimming against the current and your code will be difficult to understand.

