

Compsci 290/Mobile, Java

Owen Astrachan

Landon Cox

January 16, 2018

Java Classes and Objects

- **Class encapsulates state and behavior**
 - State is typically private
 - Android uses `mInstanceVar` convention
- **Class is an object factory**
 - Calling `new` creates a new instance
 - Everything in Java is a pointer/reference

Classes and Objects

- **Classes communicate and collaborate**
 - Parameters: use-a, send and receive
 - Containment: has-a, aggregate or responsible
 - Inheritance: is-a, extends, specializes
- **Inheritance and interfaces**
 - More on this throughout semester, especially at beginning

Tell Don't Ask

- **Tell objects what you want them to do, do not ask questions about state, make a decision, then tell them what to do**

(Pragmatic Programmers, LLC)

- Think declaratively, not procedurally
- Don't ask for a map, then walk through the map
- Instead of iteration, apply to all
 - Breaks when we don't want to apply to all

- **Rules are made to be broken**

- Reduce coupling, better code

Law of Demeter

- **Don't talk to objects, don't call methods. The more you talk, the more you rely on something that will break later**
 - Call your own methods
 - Call methods of parameter objects
 - Call methods if you create the object
- **Do *NOT* call methods on objects returned by calls**

```
List all = obj.getList();  
all.addSpecial(key, getValue());  
obj.addToList(key, getValue()); // ok here
```

Open Closed Principle

- **Classes and Programs will be changed ...**
 - Open to extension
 - Closed to modification
- **What does this mean?**
 - If not modified, don't need to be re-tested on a Unit testing basis
 - Extension can be by design, by language features

Loose Coupling

- **We want classes to be loosely coupled**
 - Independent of each other in that they interact via APIs
 - Changes in one class have minimal impact on other classes except via APIs and those should be changed infrequently
- **Applications and programs change**
 - Minimize the "ripple" effect through the system

High Cohesion

- **Classes capture one abstraction**
 - Create more classes when you need them, don't be a class miser or misanthrope (word abuse)
- **Keep things simple, strive for simplicity**
 - Don't use Swiss-army knife approach, one tool for one purpose
- **Loose coupling and high cohesion, goals for programming**

Design Patterns


“... describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”

Christopher Alexander, quoted in GOF

- **Name**
 - good name is a handle for the pattern, builds vocabulary
- **Problem**
 - when applicable, context, criteria to be met, design goals
- **Solution**
 - design, collaborations, responsibilities, and relationships
- **Forces and Consequences**
 - trade-offs, problems, results from applying pattern: help in evaluating applicability


Odyssey of the Mind

Design patterns: an essential component of CS curricula

Full Text:  [PDF](#)

Authors: [Owen Astrachan](#) [Duke University](#)
[Garrett Mitchener](#) [Duke University](#)
[Geoffrey Berry](#) [Duke University](#)
[Landon Cox](#) [Duke University](#)




 1998 Article



[Bibliometrics](#)

- Citation Count: 45
- Downloads (cumulative): 974
- Downloads (12 Months): 19
- Downloads (6 Weeks): 3

OO overkill: when simple is better than not

Full Text:  [PDF](#)

Author: [Owen Astrachan](#) [Computer Science Department, Duke University](#)


Published in:



- Proceeding
[SIGCSE '01](#) Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education
Pages 302-306

Charlotte, North Carolina, USA



 2001 Article



[Bibliometrics](#)

- Citation Count: 8
- Downloads (cumulative): 337
- Downloads (12 Months): 6
- Downloads (6 Weeks): 0

MolecularBalls

- https://en.wikipedia.org/wiki/Abstract_factory_pattern
- <https://coursework.cs.duke.edu/ola/molecules>
- **Where do MolecularBalls come from?**
 - Is the source of the ball important?
 - Should Main Program be aware of source?
- **Is there more than one kind of Ball?**
 - Bouncing behavior?
 - Color
 - What's state and what's behavior?

Writing Programs

- **Always do the hard part first. If the hard part is impossible, why waste time on the easy part? Once the hard part is done, you're home free.**
- **Always do the easy part first. What you think at first is the easy part often turns out to be the hard part. Once the easy part is done, you can concentrate all your efforts on the hard part.**
- **Whenever possible, re-use, share, borrow, but do not steal code**

Design Patterns

- **MVC, aka Observer/Observable**
 - Separate concerns, especially important for GUIs
- **Composite**
 - Container is/contains Layout/View, File/Directory
- **Factory**
 - Separate creation from class, install new creators

- **Proxy/Adapter**

- Stand-in with same interface, adapt interface as needed

- **Decorator**

- Is-a and Has-a, e.g., Filters and java I/O

- **Command**

- Function/request object, undoable action