

CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2014
Doc 1 Introduction
Jan 23, 2014

Copyright ©, All rights reserved. 2014 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

References

Object-Oriented Design Heuristics, Riel,
Addison Wesley, 1996

Reading

Jan 28 - Big Ball of Mud, <http://www.laputan.org/mud/mud.html>
What Compsi textbooks don't tell you: Real world code sucks,
[http://www.theregister.co.uk/2012/12/21/financial_software_disasters/
print.html](http://www.theregister.co.uk/2012/12/21/financial_software_disasters/print.html)

Jan 30 - Refactoring, Chapters 1 & 2

Feb 4 - Refactoring, Chapters 3 & 4

Crashing

Last Day to Drop or Add Class

Last day to file for graduation for spring or summer

Feb 4

Course Web Site

<http://www.eli.sdsu.edu/index.html>

CS 635 Spring 14

Lecture Notes

Assignments

Wiki

Course Portal

Syllabus

Reading Assignments

Languages

Java, C++, Ruby, Objective C or Smalltalk

Preferred Languages

Java

Smalltalk

Objective C

Ruby

C++ is **STRONGLY** Discouraged

I have not used C++ in over 10 years

I don't like the language

It is very difficult to grade

Each additional language make grading harder

It is extremely hard to deal with GUI assignments in C++

Assignments are often harder in C++

What about C#

I don't know the C# libraries

Which makes it too hard to grade assignments

What this course is about

Writing quality OO code
Design Patterns
Coupling & Cohesion

Unit Testing
Refactoring

Scale Changes Everything



Remedial Material

Formatting

Format your code uniformly and consistently to show the black structure of your code

```
public void commandAction(Command c, Displayable d) {
    if (c == restartCmd) {
theGame.restart();
    } else if (c == levelCmd) {
        Item[] levelItem = {
new Gauge("level", true, 9 theGame.getLevel());
Form f = new Form("Change Level", levelItem);
        f.addCommand(OkCmd);
        f.addCommand(cancelCmd);
f.setCommandListener(this);
        Display.getDisplay(this).setCurrent(f);
    } else if (c == exitCmd) {
destroyApp(false);
notifyDestroyed();
    }
}
```

```
public void commandAction(Command c, Displayable d) {  
    if (c == restartCmd) {  
        theGame.restart();  
    } else if (c == levelCmd) {  
        Item[] levelItem = {  
            new Gauge("level", true, 9 theGame.getLevel())};  
        Form f = new Form("Change Level", levelItem);  
        f.addCommand(OkCmd);  
        f.addCommand(cancelCmd);  
        f.setCommandListener(this);  
        Display.getDisplay(this).setCurrent(f);  
    } else if (c == exitCmd) {  
        destroyApp(false);  
        notifyDestroyed();  
    }  
}
```

Grading Policy - Formatting

Each method with such poor formatting
loses 1 point per method

Name Structure - Language Conventions

	Java	Smalltalk	C#	Ruby
Class	PascalCase	PascalCase	PascalCase	PascalCase
Method	camelCase	camelCase	PascalCase	foo_bar
Field	camelCase	camelCase	camelCase	@foo_bar
Parameter	camelCase	camelCase	camelCase	foo_bar
Local Variable	camelCase	camelCase	camelCase	foo_bar

PascalCase ArrayList

camelCase courseSize

Grading Policy - Names

Each name that does not following your languages naming structure

Loses 1 point/name

Up to 20 points/assignment

Avd brvtns

brvtns r hrd t rd

n brvtn cn stnd fr dffrnt thngs

tmp - tmpr r tmprtr

Dffrnt ppl wll brvt dffrntl

Ds tcmlpt s dn't hv t typ lng nms

Avoid Abbreviations

Abbreviations are hard to read


An abbreviation can stand for different things


tmp - temporary or temperature


Different people will abbreviate differently

IDEs autocomplete so don't have to type long names

Describe What "flag" is Used For

 if (flag) {
 ...
}

 if (foundDuplicate) {
 ...
}

 flag
flagStatus
computeFlag

Do not help understand code

Variables 1 through N



```
String s1;  
String s2;
```



```
String fileContents;  
String pattern;
```

Who can remember the difference between s1 and s2?

Avoid Names With No Meaning

 MyLinkedList

Who are you?

What makes your LinkedList different?

 temp

All variables are temporary

```
swap = a;
```

```
a = b;
```

```
b = swap
```

Guidelines - Class Names

Use nouns

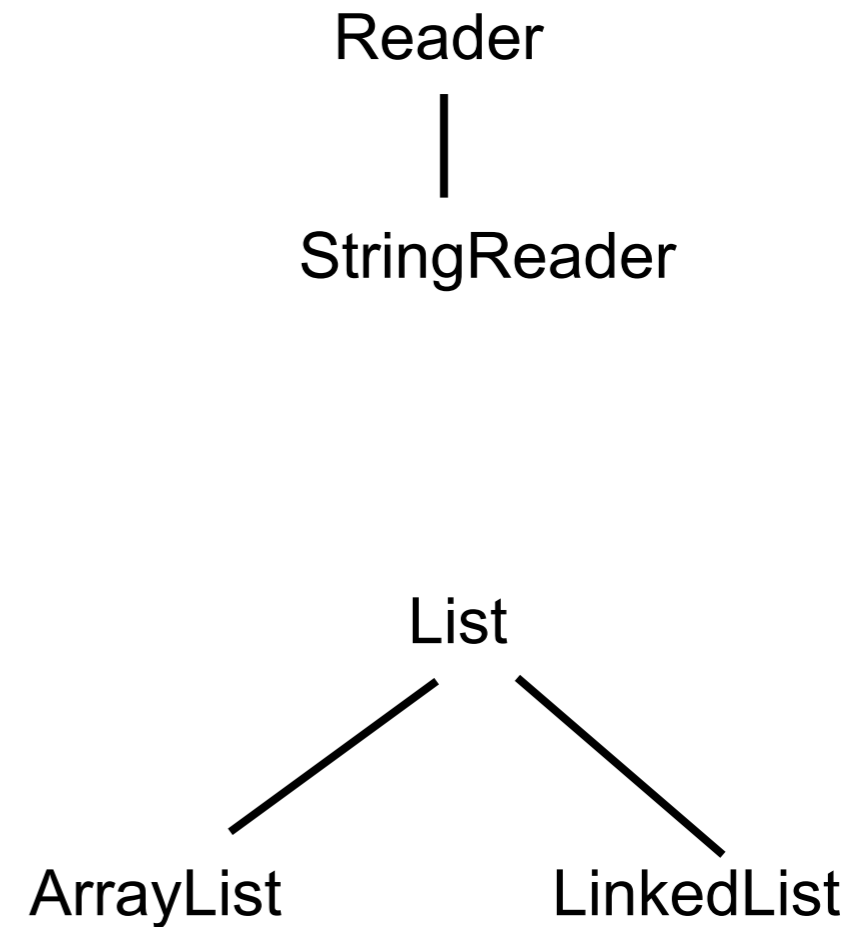
No abbreviations

Superclass

Single word to convey its purpose

Subclass

Prepend adjective to superclass name



Guidelines - Method/Function/Procedure Names

Describe what method does

Use verb to describe an action

```
add(int index, E element)  
clear()
```

If returns a value name what it returns

```
iterator()  
subList(int fromIndex, int toIndex)
```

If returns boolean value make it true/false statement

```
isEmpty()  
contains(Object o)
```

Guidelines - Variables, Fields, Parameters

Use names that indicate role variable is playing

If declare variable types don't use type as name

Use plurals to indicate collections

Make boolean variable names true/false statement

isVisible, hasMultipleParts,



```
public void execute(Vector vector) {  
    Stack s;  
}
```



```
public void execute(Vector commands) {  
    Stack commandsExecuted;  
}
```

Summary

Use names to help the reader understand the code

Follow language conventions

Avoid abbreviations

Use names that indicate role item is playing

Remedial Points

Each assignment I indicate points for different things

Remedial points (names, formatting) are in addition to those

Assignment 1

Review

Define

Object
Class

What are the Benefits of OO

Issues?

```
public class A {  
    public int x;  
    public int y;  
    public int z;  
}
```


Issues?

```
class Stack
  def initialize
    @elements = Array.new
  end

  def empty?
    return @elements.empty?
  end

  def push(element)
    @elements.push(element)
  end

  def pop
    @elements.pop
    return elements
  end
end
```

A verses B

```
public class A {  
    public int x;  
    public int y;  
    public int z;  
}
```

```
public class B {  
    private int x;  
    private int y;  
    private int z;  
  
    public int getX() { return x;}  
    public int getY() { return y;}  
    public int getZ() { return z;}  
    public void setX(int value) {x = value;}  
    public void setY(int value) {y = value;}  
    public void setZ(int value) {z = value;}  
}
```

Heuristics

Keep related data and behavior in one place

A class should capture one and only one key abstraction

Heuristics

Beware of classes that have many accessor methods defined in their public interface

Do not create god classes/objects in your system

Beware of classes that have too much noncommunicating behavior

Learning, Understanding & Memorization

One disease long life

No disease short life