References

Object-Oriented Design Heuristics, Chapter 2
Why is OO Good?
Does your code achieve those properties of goodness?
struct Stack {
    float[] elements
    int topOfStack
}

void push(stack *Stack,float elementToAdd) {
    stack.elements[topOfStack++] = elementToAdd;
}
public class Stack {
    public float[] elements
    public int topOfStack
}

public class StackStuff{
    public void push(stack Stack,float elementToAdd) {
        stack.elements[topOfStack++] = elementToAdd;
    }
}
Terms

Class
A blueprint to create objects
Includes attributes and methods that the created objects all share

Object
Allocated region of storage
Both the data and the instructions that operate on that data
Instance of a class
Abstraction

“Extracting the essential details about an item or group of items, while ignoring the unessential details.”
   Edward Berard

“The process of identifying common patterns that have systematic variations; an abstraction represents the common pattern and provides a means for specifying which variation to use.”
   Richard Gabriel
Encapsulation

Enclosing all parts of an abstraction within a container
Information Hiding

Hiding of design decisions in a computer program

Hide decisions are most likely to change,
To protect other parts of the program
Class

Represents an abstraction

Encapsulates data and operations of the abstraction

Hide design decisions/details
OOP to me means only
messaging,
local retention and
protection and hiding of state-process, and
extreme LateBinding of all things
I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea.
I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea.

The big idea is "messaging"
I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea.

The big idea is "messaging"

I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages.
I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea.

The big idea is "messaging"

I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages

I wanted to get rid of data
I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea.

The big idea is "messaging"

I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages

I wanted to get rid of data

The key in making great and growable systems is much more to design how its modules communicate rather than what their internal properties and behaviors should be
Perspective is worth 80 IQ points.
Classes and Objects

Abstraction

Hide data

Hide design decisions

Messages
Relevant Heuristics

2.8 A class should capture one and only one key abstraction

2.9 Keep related data and behavior in one place
Signs of Poor OO Design

Data Classes

Helper functions
Data Class

class Point {
    private int x;
    private int y;

    public void setX(int newX) {
        x = newX;
    }

    public int getX() {
        return x;
    }

    public void setY(int newY) {
        y = newY;
    }

    public int getY() {
        return y;
    }
}

Class with
get/set methods
constructor
No or very few other methods
Helper method

Method in class that
   Does not access any field (data member, instance variables)
   Just uses parameters

Sign that Data and Operations are not being kept together
## Assignment Results

<table>
<thead>
<tr>
<th>Classes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Classes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Accessor</th>
<th>Helper</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Helper Method - Example

class CrosswordPuzzle {
    public void someMethodThatDoesStuff {
        bunch of stuff not shown
        count = vowelCount(aString);
        blah
    }

    private int vowelCount(String word) {
        int vowelCount = 0;
        for (int k = 0; k< word.length(); k++) {
            char current = word.charAt(k);
            if ( (current == 'a') || (current == 'e' ) || (current == 'i') || (current == "o") || (current == "u") )
                vowelCount++;
        }
        return vowelCount;
    }
}
Is this better? Why

class CrosswordPuzzle {
    public void someMethodThatDoesStuff {
        bunch of stuff not shown
        count = aString.vowelCount();
        blah
    }
}

class Character {
    public boolean isVowel() {
        return (this == 'a') || (this == 'e') || (this == 'i') || (this == 'o') || (this == 'u');
    }
}
Linked List Example

class LinkedList {
    private head;
    private tail;

    public LinkedList() { //some code}

    public boolean add(int index, Object element) { //blah}

    public Object get(int index) { //some code}

    public Object remove(int index) { //some code}

    public boolean remove(Object element) { //some code}

    public boolean removeLastOccurrence(Object element) {}
Node Class

Data Class

What are the operations?

```java
class Node {
    Object value;
    Node previous;
    Node next;
}
```
Heuristic

A method to help solve a problem, commonly informal

"rules of thumb"
2.1 All data should be hidden within its class

```java
public class Foo {
    public int x;
    public int y;
}

public class Foo {
    private int x;
    private int y;

    public int getX() { return x; }
    public int getY() { return y; }

    public void setX(int newX) {
        x = newX
    }

    public void setY(int newY) {
        y = newY
    }
}
```

How is the version on the right better than the version on the left?
Information Hiding

class LinkedList {
    private int size;
    private Node head;

    }

class LinkedList {
    private int size;
    private Node head;

    public int size() {
        return size;
    }

    public Node head() {
        return head;
    }
}
head
LinkedList data = new LinkedList();
data.addFirst("A");
LinkedList data = new LinkedList();
data.addFirst("A");
data.addFirst("B");
LinkedList data = new LinkedList();
data.addFirst("A");
data.addFirst("B");
data.addFirst("C");
LinkedList data = new LinkedList();
data.addFirst("A");
data.addFirst("B");
data.addFirst("C");

Node head = data.head();
head.next = new Node("Z");
Information Hiding - Copies verses Reference

class LinkedList {
    private int size;
    private Node head;

    public int size() {
        return size;
    }

    public Node head() {
        return head;
    }
}
Information Hiding

class LinkedList {
    private int size;
    private Node head;

    public void addFirst(Node newData) {
        newData.next(head);
        head = newData;
    }
}
Information Hiding

class LinkedList {
    private int size;
    private Node head;

    public void addFirst(Object data) {
        head = new Node(data, head);
    }

    public Node getFirst() {
        return head;
    }
}