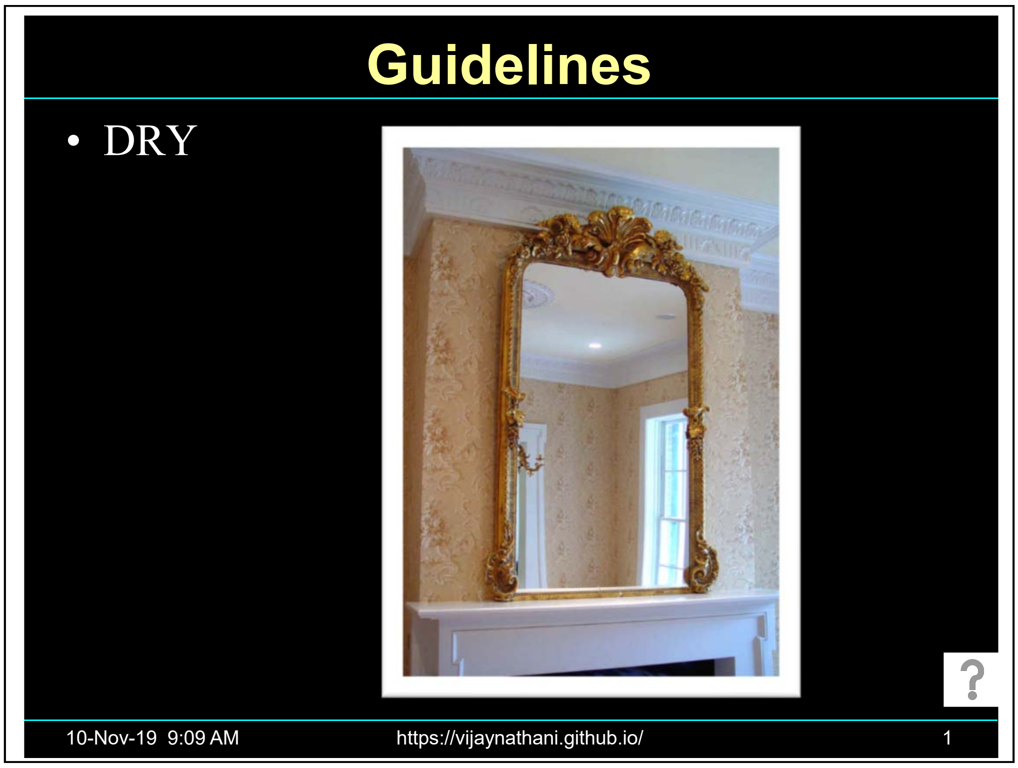


Guidelines

- DRY

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 1

Q021: BookRentals; Q11: BookRental; Q33 – SurveyData; Q37 – FOC, TT

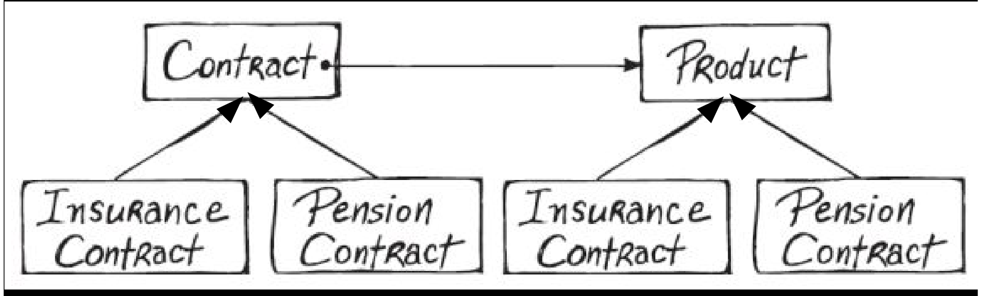
Q10 – Jbutton – Only for Java.

Q34 – replace; Q35 – BookRental

Q70 – Array Scan, Java only.

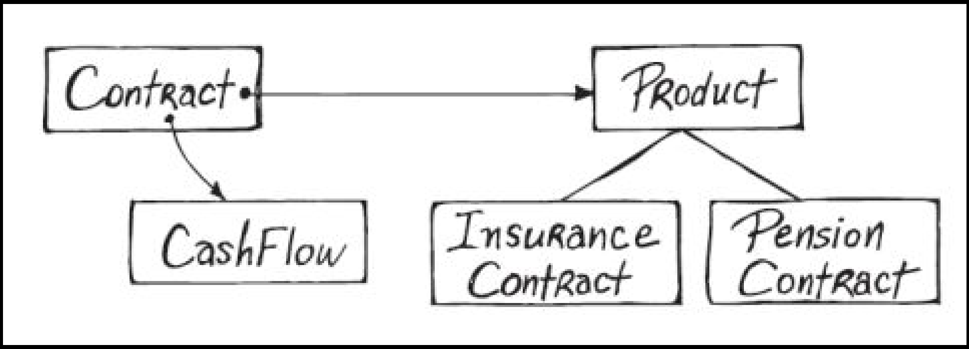
There are three numbers in software: 0, 1, and infinity. 0 represents the things we do not do in a system (we do those for free). 1 represents the things we do once and only once. But at the moment we do something twice, we should treat it as infinitely many and create cohesive services that allow it to be reused.

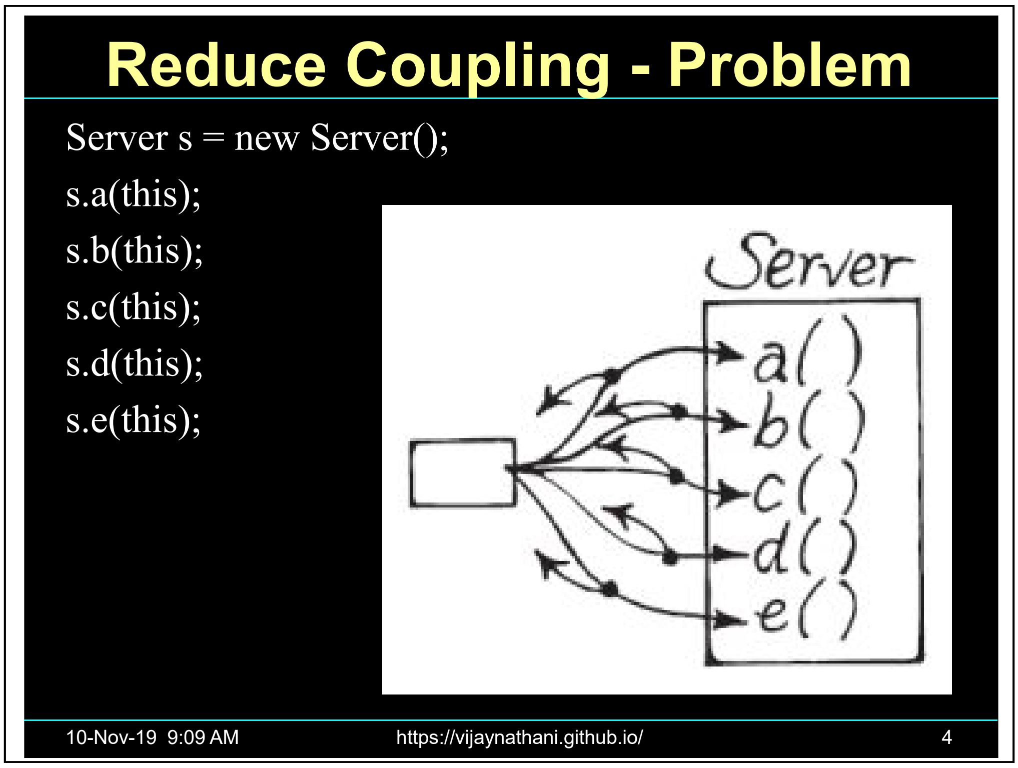
Improve



Contract
Product
Insurance Contract
Pension Contract
Insurance Contract
Pension Contract

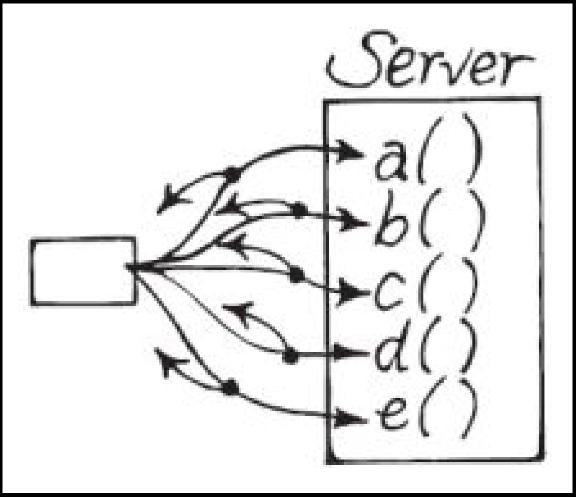
Eliminate Parallel Hierarchies

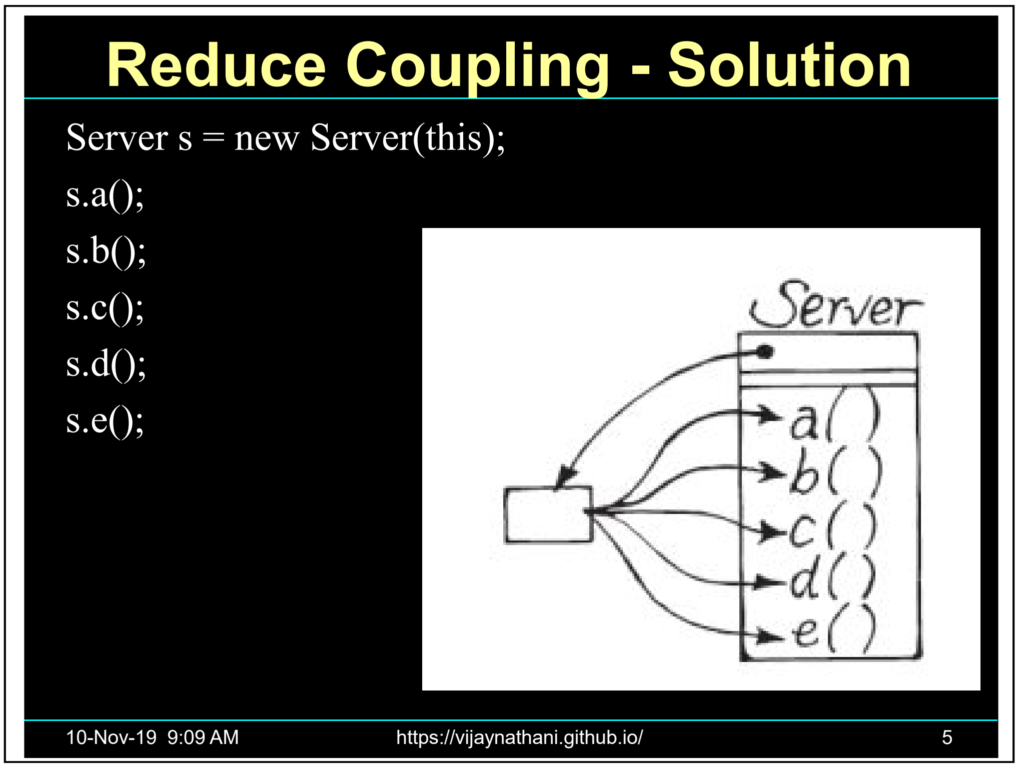




Reduce Coupling - Problem

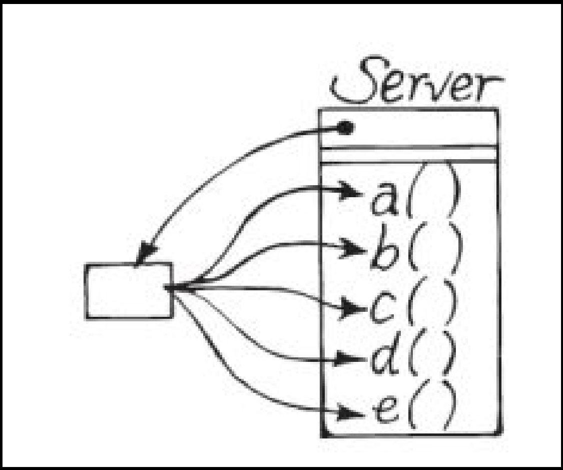
```
Server s = new Server();
s.a(this);
s.b(this);
s.c(this);
s.d(this);
s.e(this);
```



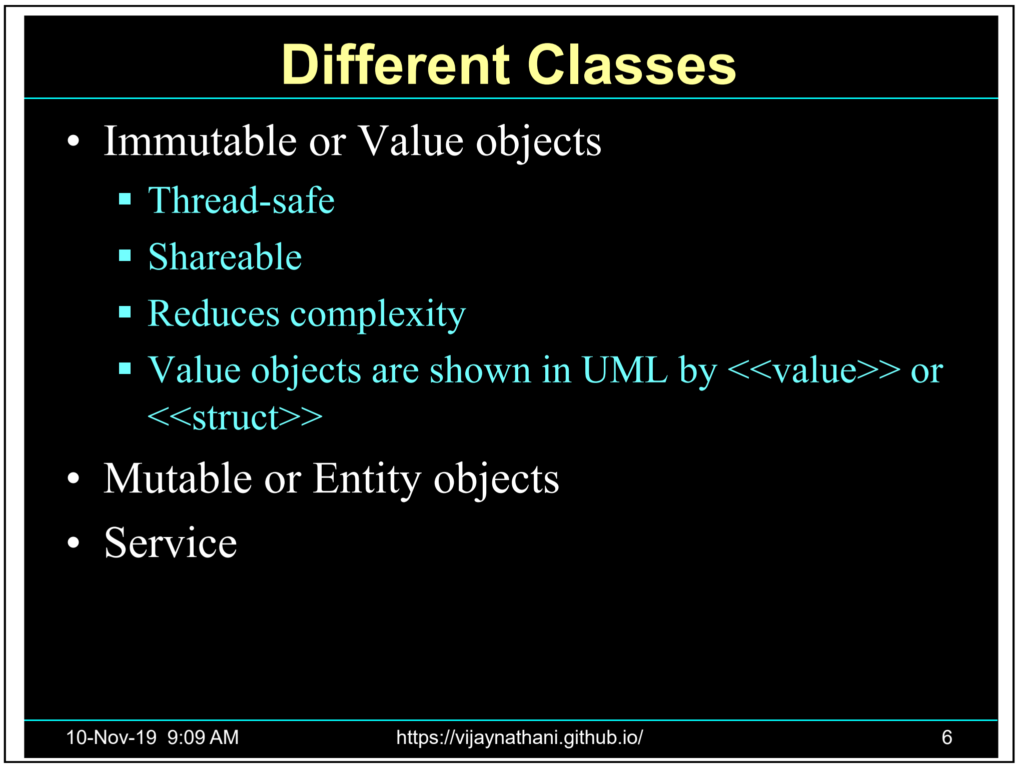


Reduce Coupling - Solution

```
Server s = new Server(this);
s.a();
s.b();
s.c();
s.d();
s.e();
```



Better still: use Observer design pattern



Different Classes

- Immutable or Value objects
  - Thread-safe
  - Shareable
  - Reduces complexity
  - Value objects are shown in UML by <<value>> or <<struct>>
- Mutable or Entity objects
- Service

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 6

Q08 – Participant. Example of immutable using JUnit.

Service – Stateless.

Mutable. Keep state-space small, well-defined. Make clear when its legal to call which method.

Reference objects (Entities) and Value objects.

> Keep classes immutable unless there is a good reason to do otherwise.
>
> Enforcing contracts is a great way to reduce complexity. Simplifies development effort
>
> All instance variables should be “private final”.
>
> Value objects are not in any specific tier. They should be useable from all tiers.

================================

Immutable Example:

> Color, Line Style, Name, OrderNumber, ZipCode
>
> Company class has company address, company fax, company name, company Telephone, etc.
>
> Integers with limitations - Percentage(0 to 100%), Quantity (>=0)
>
> Arguments used in service methods

Exceptions that are domain logical

JDK examples

Bad – Date, Calendar, Dimension

Good - TimerTask

Just as String does not belong to any tier.

For many applications makes sense to

Use immutable classes most of the time

Handle mutations only in a part of the application

This limits complexity to one part

Rest of the application uses immutable classes to reduce complexity and enhance understanding.

E.g. Historic information, read from the database, should be immutable.

Example: A credit card engine

It needs to read details about Customer, Card and Transaction.

In the scope of this engine, all these classes are immutable.

Getters should normally return immutable objects.

java.util.collections.unmodifiableList

System.Collections.Generic. IEnumerable<T> or List<T>.AsReadOnly

----

Entities have an identity. So two account entities with same balance are not the same. They are almost always persistent.

Values are usually a part of Entity.

DTO (Data Transfer Objects) are different from Value objects. DTO is a

-purpose: data transfer – technical construct

- bunch of data – not necessarily coherent

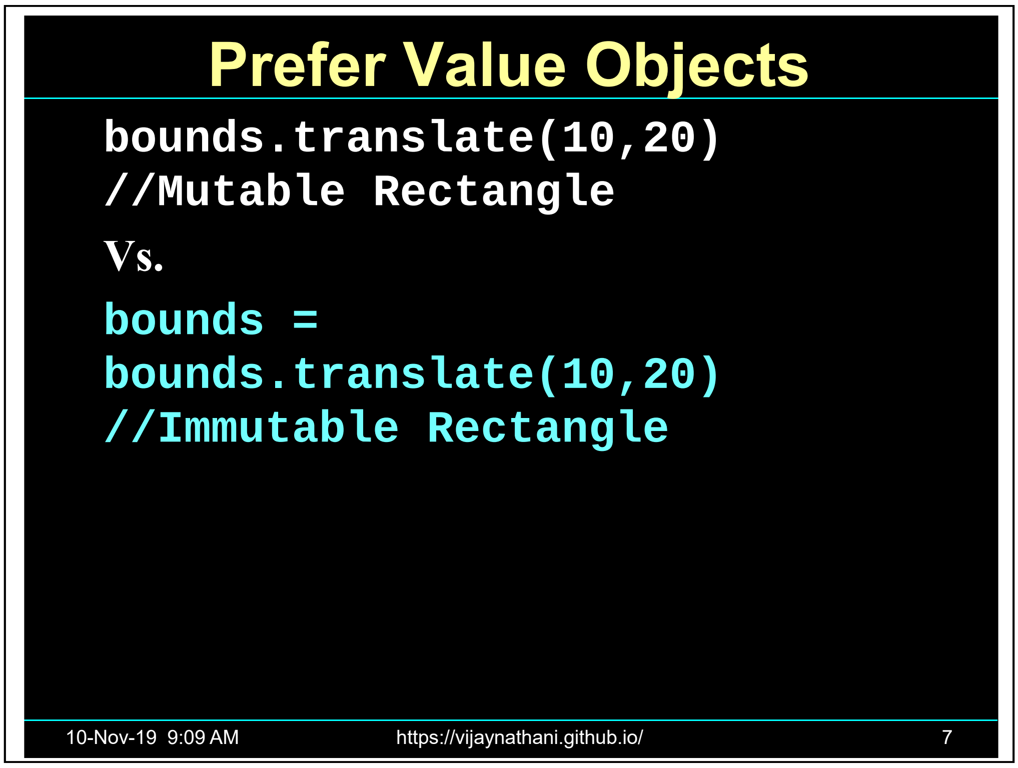
- no / little behavior

Value Object

-Purpose: domain representation

-High coherent data

-Rich on behavior



Prefer Value Objects

```
bounds.translate(10,20)
//Mutable Rectangle
```

Vs.

```
bounds =
bounds.translate(10,20)
//Immutable Rectangle
```

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 7

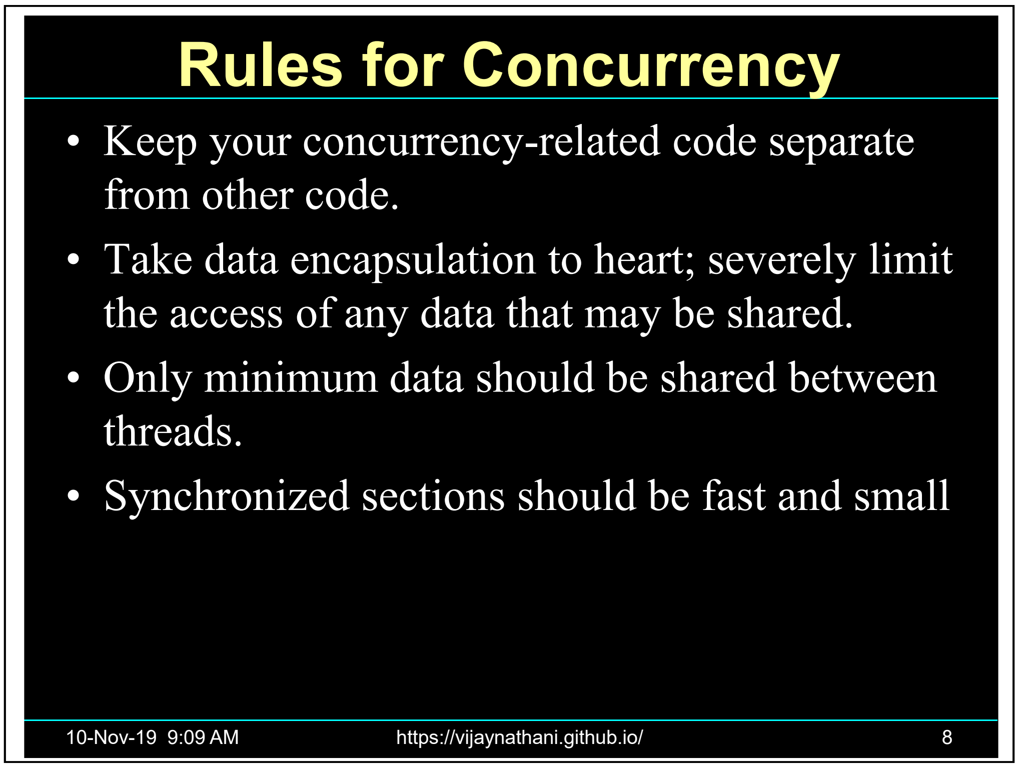
In modern functional languages, by default a variable is immutable i.e. its value cannot change

e.g. in F#

```
let x = 1 //the value of x cannot be changed.
let mutable y = 1 // the value of y can change.
```

In Scala

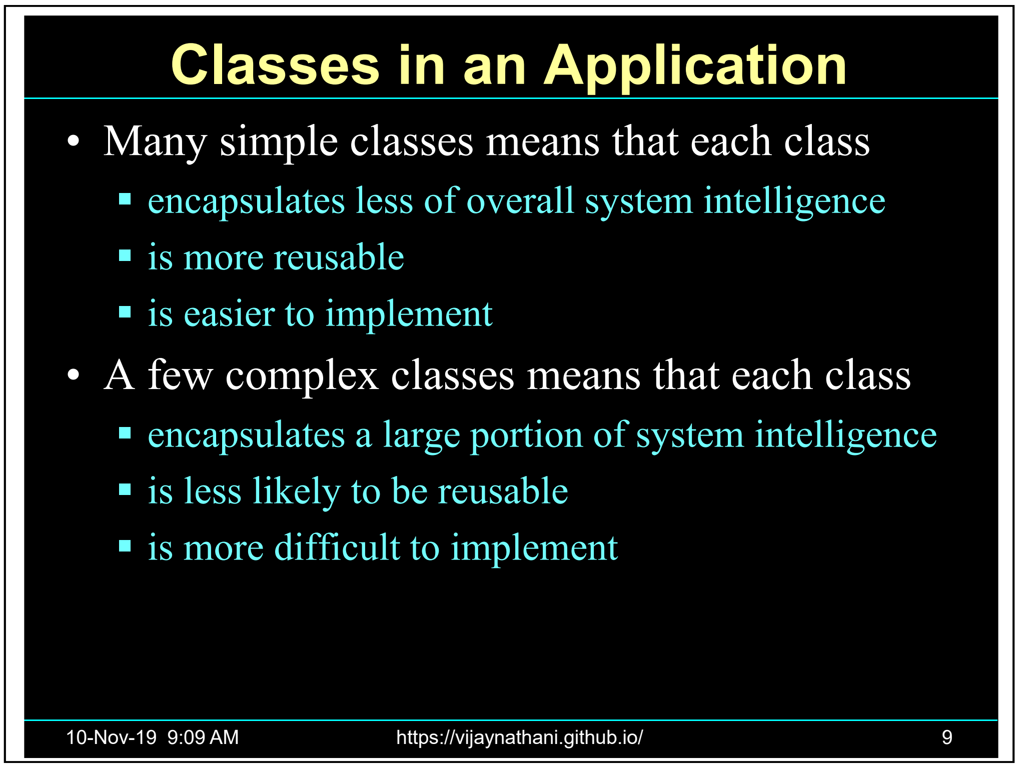
```
val x = 1; //x is immutable
var y = 1; //y is mutable
```



Rules for Concurrency

- Keep your concurrency-related code separate from other code.
- Take data encapsulation to heart; severely limit the access of any data that may be shared.
- Only minimum data should be shared between threads.
- Synchronized sections should be fast and small

Return copies of data or use immutable objects.



Classes in an Application

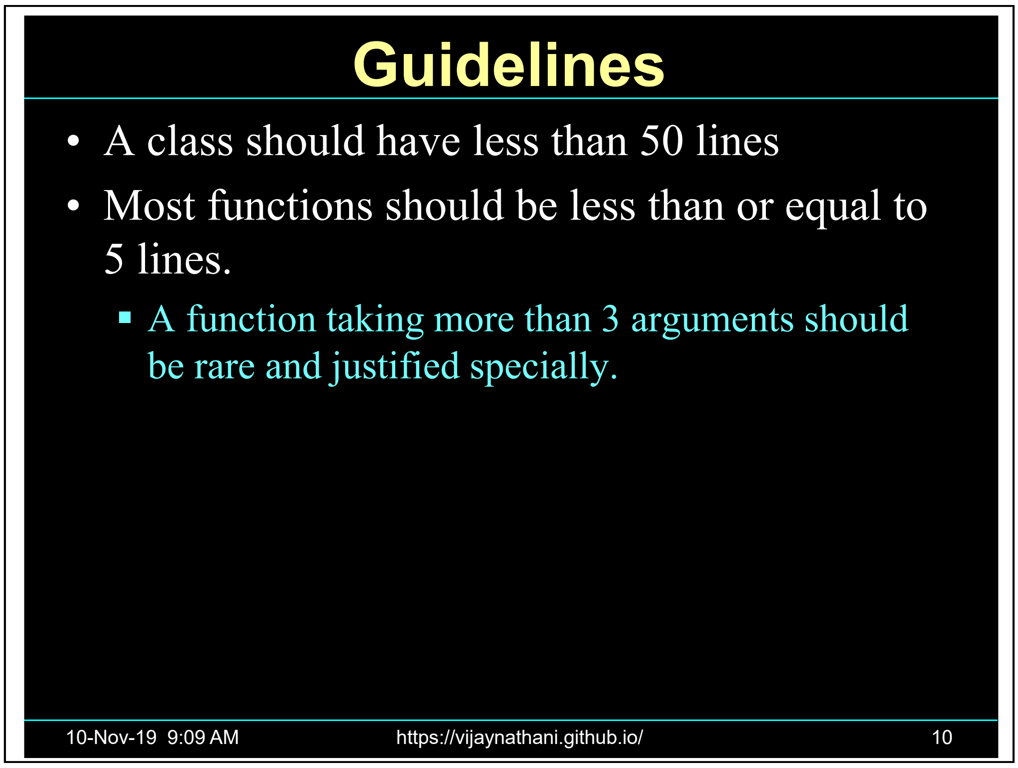
- Many simple classes means that each class
  - encapsulates less of overall system intelligence
  - is more reusable
  - is easier to implement
- A few complex classes means that each class
  - encapsulates a large portion of system intelligence
  - is less likely to be reusable
  - is more difficult to implement

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 9

Lots of little pieces

Classes are cohesive

Methods do only one thing.

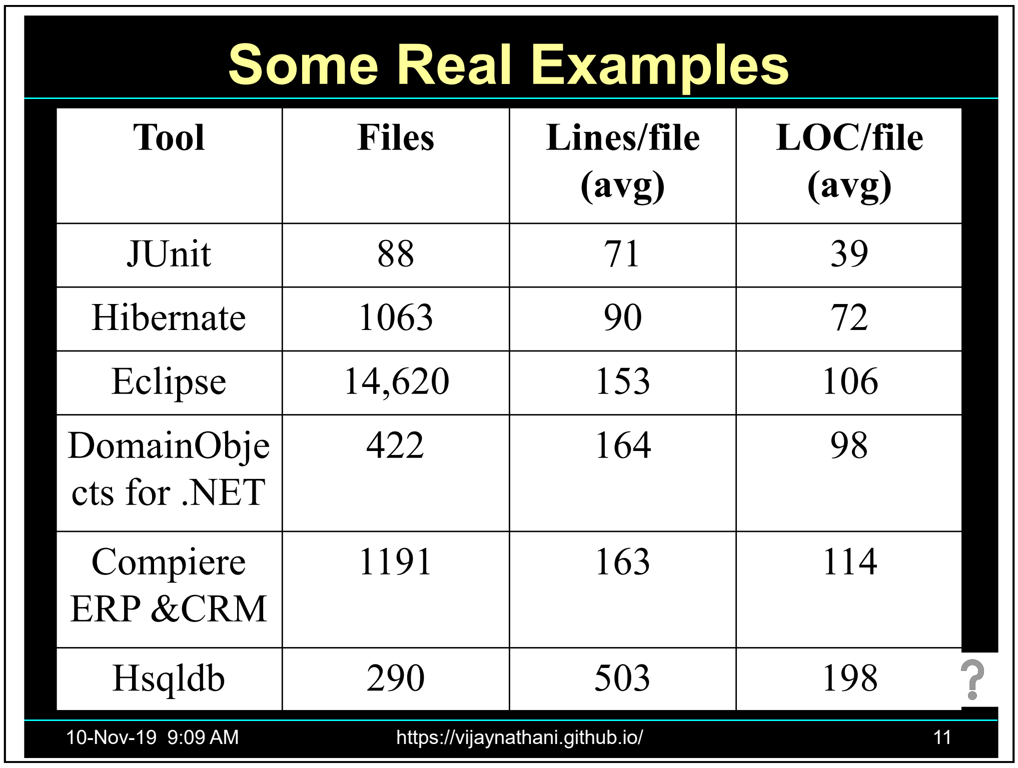


Guidelines

- A class should have less than 50 lines
- Most functions should be less than or equal to 5 lines.
  - A function taking more than 3 arguments should be rare and justified specially.

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 10

Note: On a Home PC - 1 Million function calls take 8 milliseconds and 1 Million objects are created in 23 milliseconds



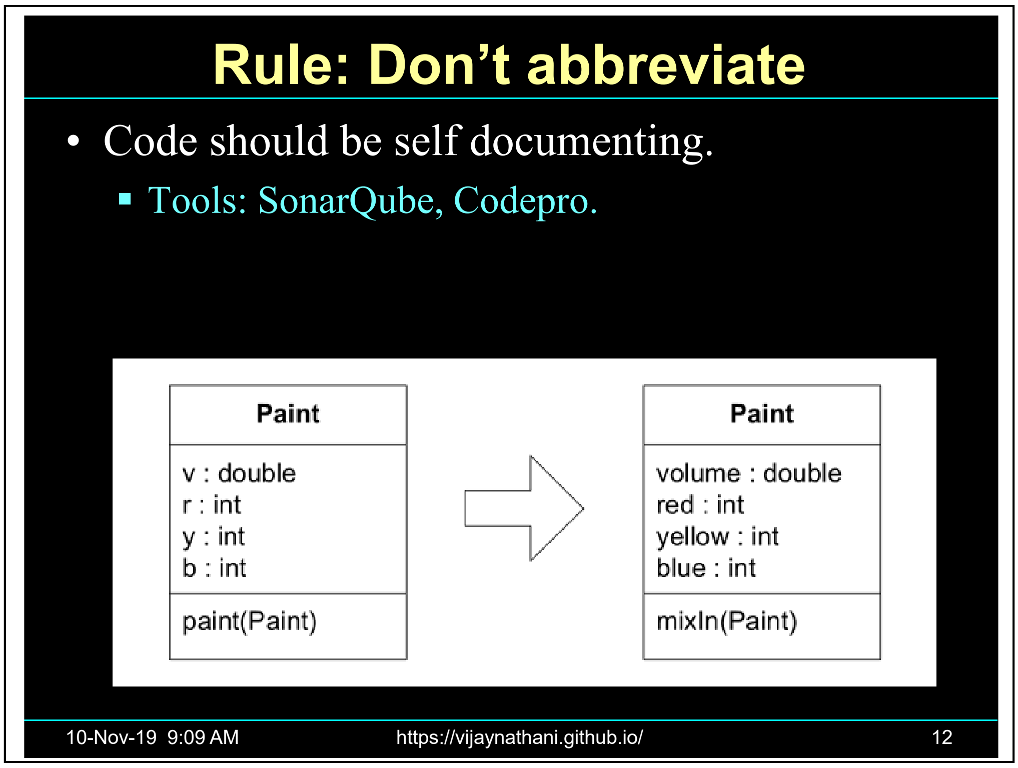
Some Real Examples

Tool	Files	Lines/file (avg)	LOC/file (avg)
JUnit	88	71	39
Hibernate	1063	90	72
Eclipse	14,620	153	106
DomainObjects for .NET	422	164	98
Compiere ERP &CRM	1191	163	114
Hsqldb	290	503	198

?

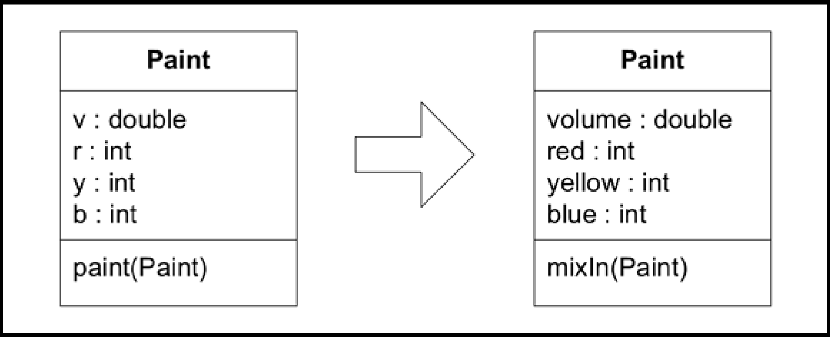
Q56srp – Restaurants

Q59srp - Customers



Rule: Don’t abbreviate

- Code should be self documenting.
  - Tools: SonarQube, Codepro.



10-Nov-19 9:09 AM https://vijaynathani.github.io/ 12

Public API’s have to be documented i.e. every class, function, interface, exceptions. Mutable objects that can / cannot be modified.

This is possible by choosing the right variable and function names.

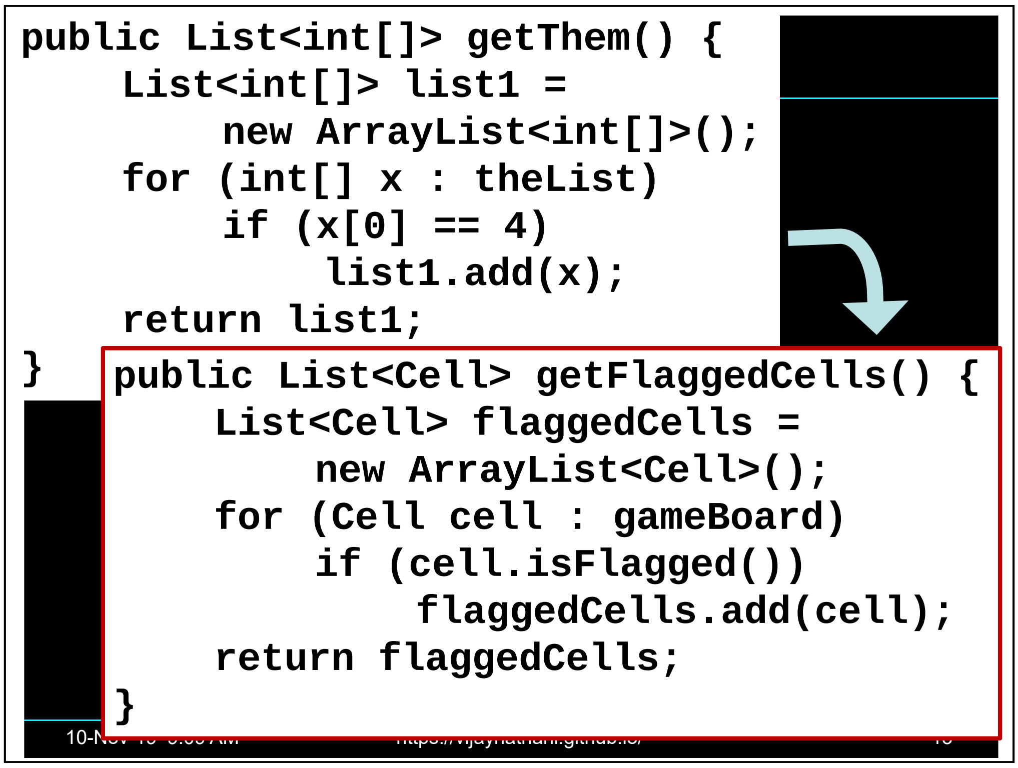
Comments are secondary because they tend to lie

Java: Checkstyle, PMD

C#: SytleCop+,FxCop, Simian, Ncover, NDepend for cyclomatic complexity.

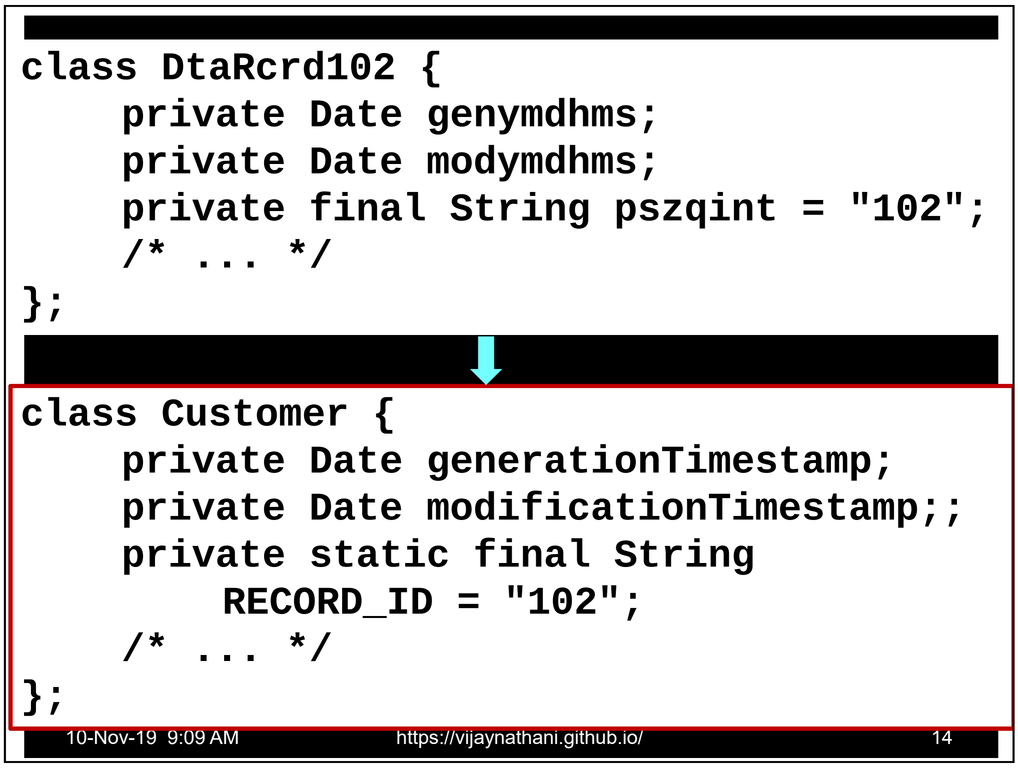
C++: Simian or PMD CPD for duplication, coverity for source code analysis

JavaScript: JSHint



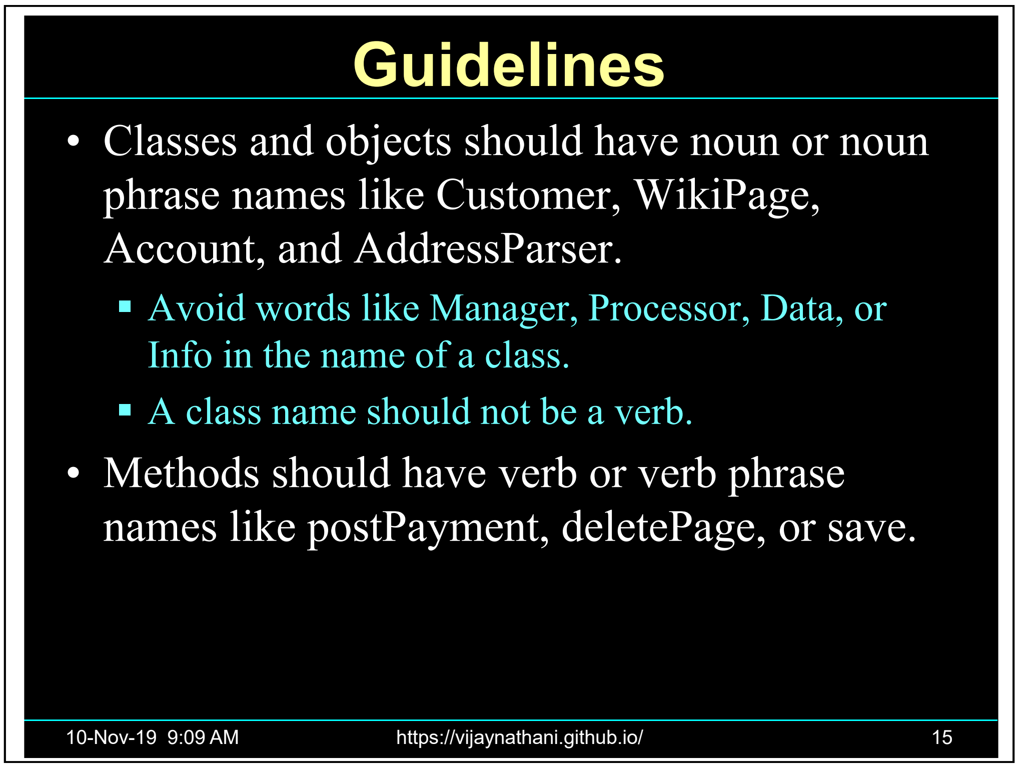
```
public List<int[]> getThem() {
    List<int[]> list1 =
        new ArrayList<int[]>();
    for (int[] x : theList)
        if (x[0] == 4)
            list1.add(x);
    return list1;
}
```

```
public List<Cell> getFlaggedCells() {
    List<Cell> flaggedCells =
        new ArrayList<Cell>();
    for (Cell cell : gameBoard)
        if (cell.isFlagged())
            flaggedCells.add(cell);
    return flaggedCells;
}
```



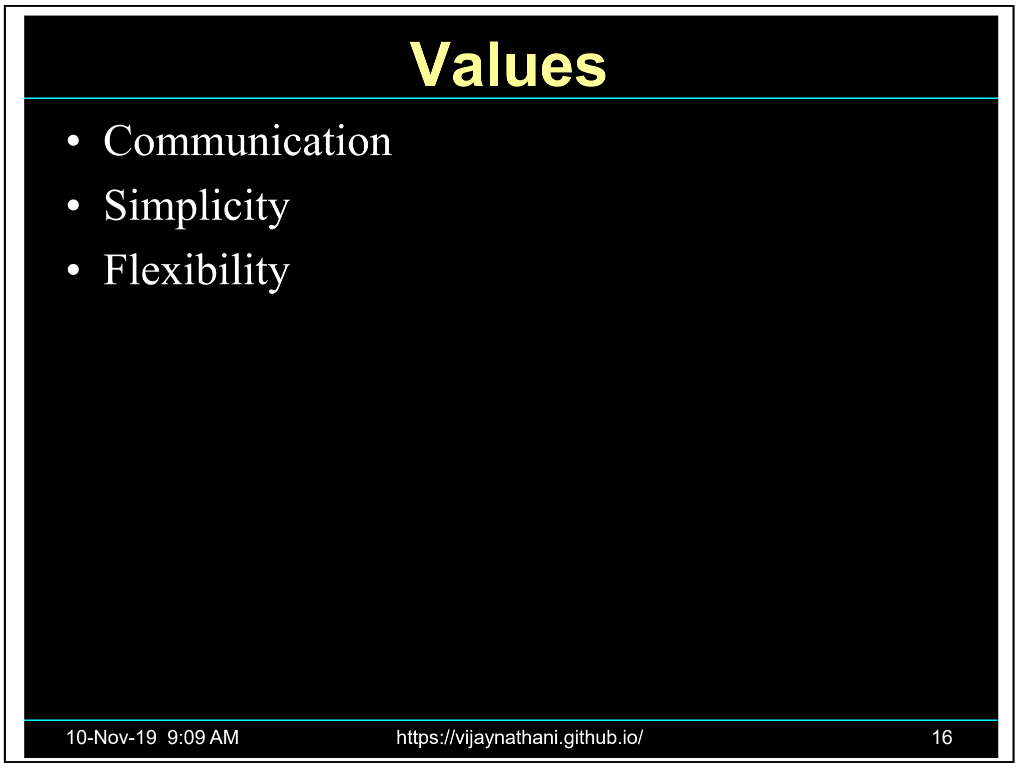
```
class DtaRcrd102 {
    private Date genymdhms;
    private Date modymdhms;
    private final String pszqint = "102";
    /* ... */
};
```

```
class Customer {
    private Date generationTimestamp;
    private Date modificationTimestamp;;
    private static final String
        RECORD_ID = "102";
    /* ... */
};
```



Guidelines

- Classes and objects should have noun or noun phrase names like Customer, WikiPage, Account, and AddressParser.
  - Avoid words like Manager, Processor, Data, or Info in the name of a class.
  - A class name should not be a verb.
- Methods should have verb or verb phrase names like postPayment, deletePage, or save.



Values

- Communication
- Simplicity
- Flexibility

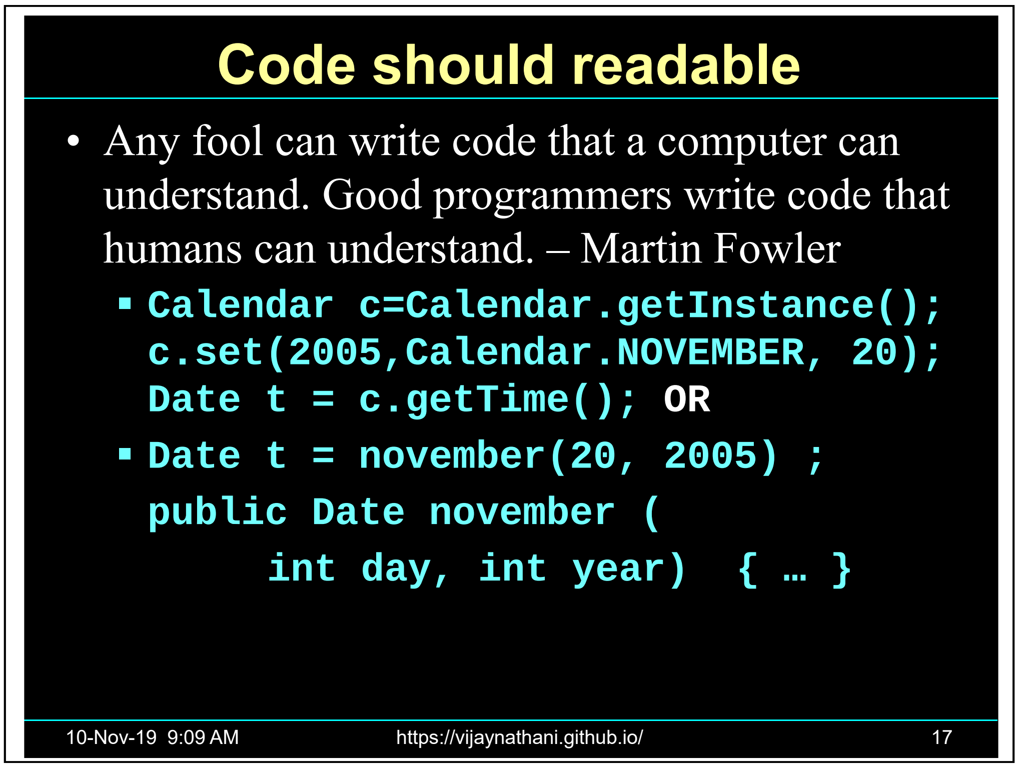
10-Nov-19 9:09 AM https://vijaynathani.github.io/ 16

Mostly they complement each other.

Code communicates well, when a reader can understand it, modify it and use it.

Eliminating excess complexity, makes the program easier to understand, modify and use.

Flexibility means that the program can be changed.



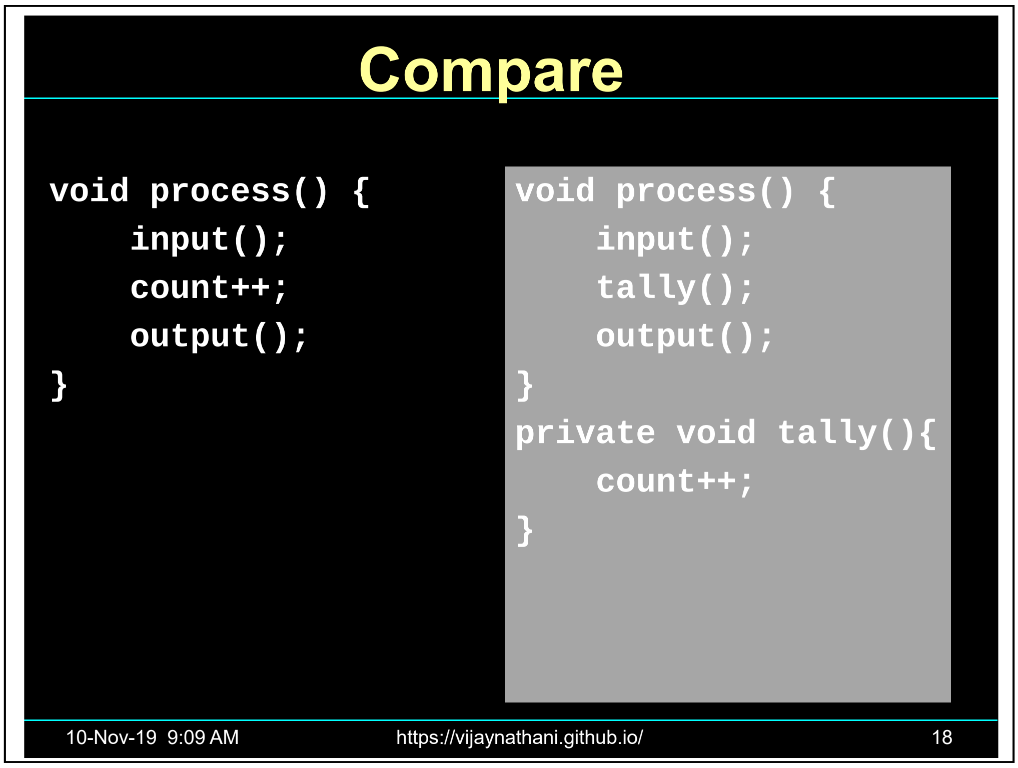
Code should readable

- Any fool can write code that a computer can understand. Good programmers write code that humans can understand. – Martin Fowler

  -
    ```
    Calendar c=Calendar.getInstance();
    c.set(2005,Calendar.NOVEMBER, 20);
    Date t = c.getTime(); OR
    ```

  -
    ```
    Date t = november(20, 2005) ;
    public Date november (
         int day, int year)  { … }
    ```

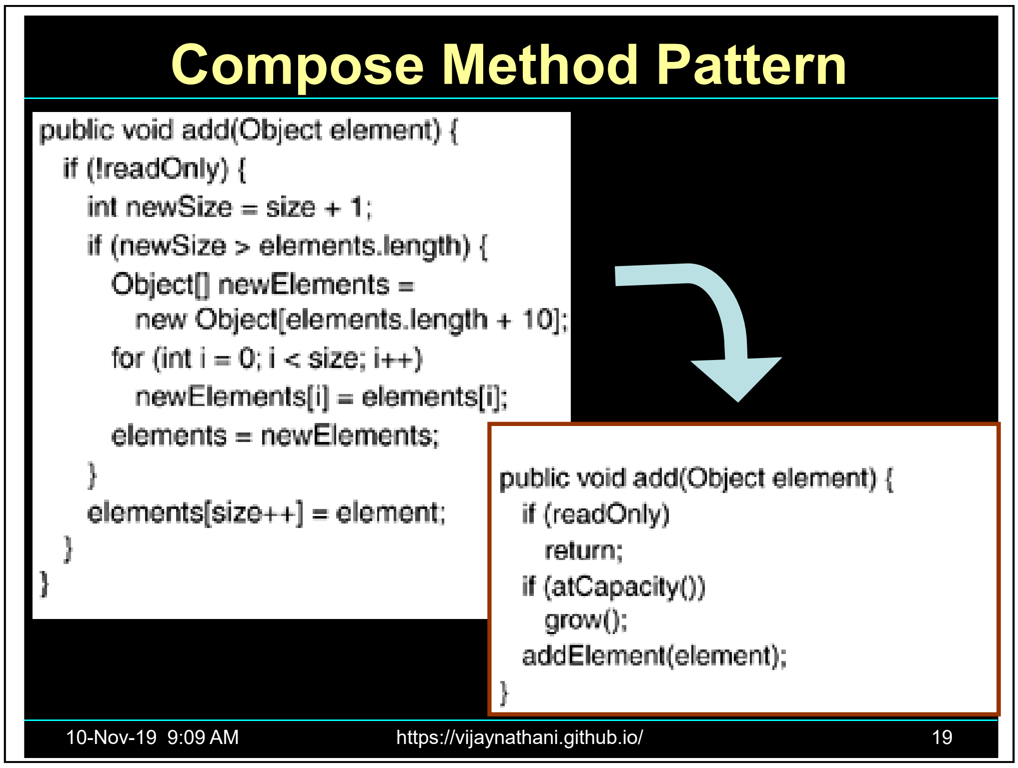
10-Nov-19 9:09 AM https://vijaynathani.github.io/ 17



Compare

```
void process() {
    input();
    count++;
    output();
}
```

```
void process() {
    input();
    tally();
    output();
}
private void tally(){
    count++;
}
```



Compose Method Pattern

```
public void add(Object element) {
  if (!readOnly) {
    int newSize = size + 1;
    if (newSize > elements.length) {
      Object[] newElements =
        new Object[elements.length + 10];
      for (int i = 0; i < size; i++)
        newElements[i] = elements[i];
      elements = newElements;
    }
    elements[size++] = element;
  }
}
```

```
public void add(Object element) {
  if (readOnly)
    return;
  if (atCapacity())
    grow();
  addElement(element);
}
```

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 19

Benefits and Liabilities

+Efficiently communicates what a method does and how it does what it does.

+Simplifies a method by breaking it up into well-named chunks of behavior at the same level of detail.

–Can lead to an overabundance of small methods.

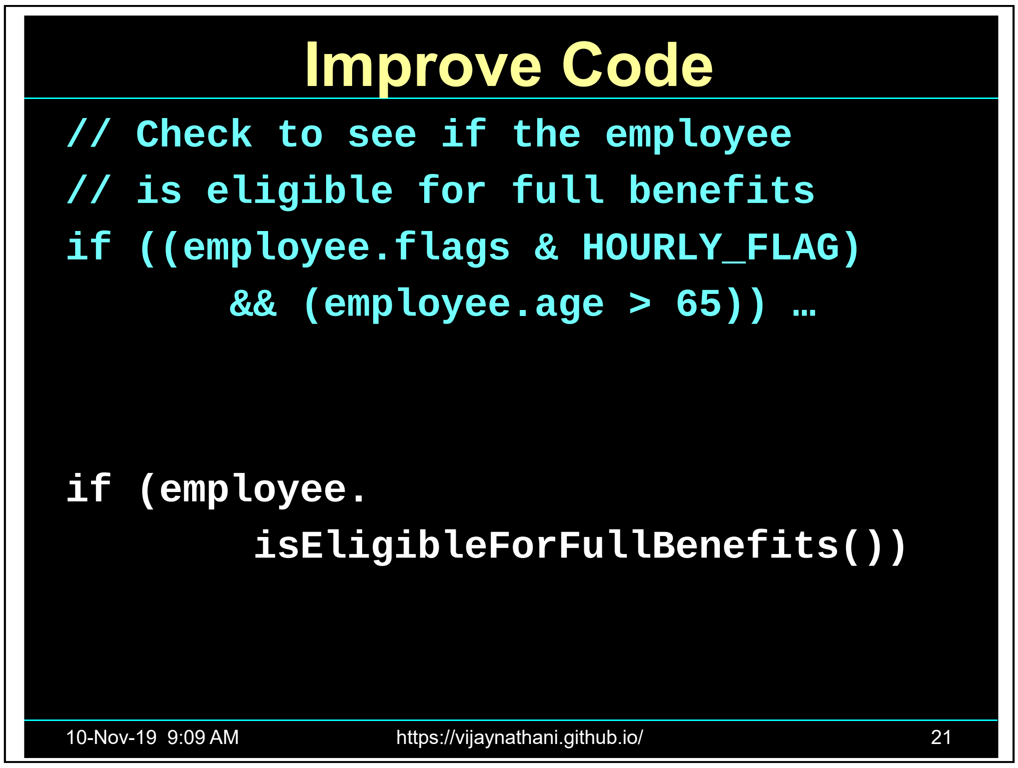
–Can make debugging difficult because logic is spread out across many small methods.

Improve

```
flags |= LOADED_BIT;
```

- Solution: Extract to a message

```
void setLoadedFlag() {
    flags |= LOADED_BIT;
}
```

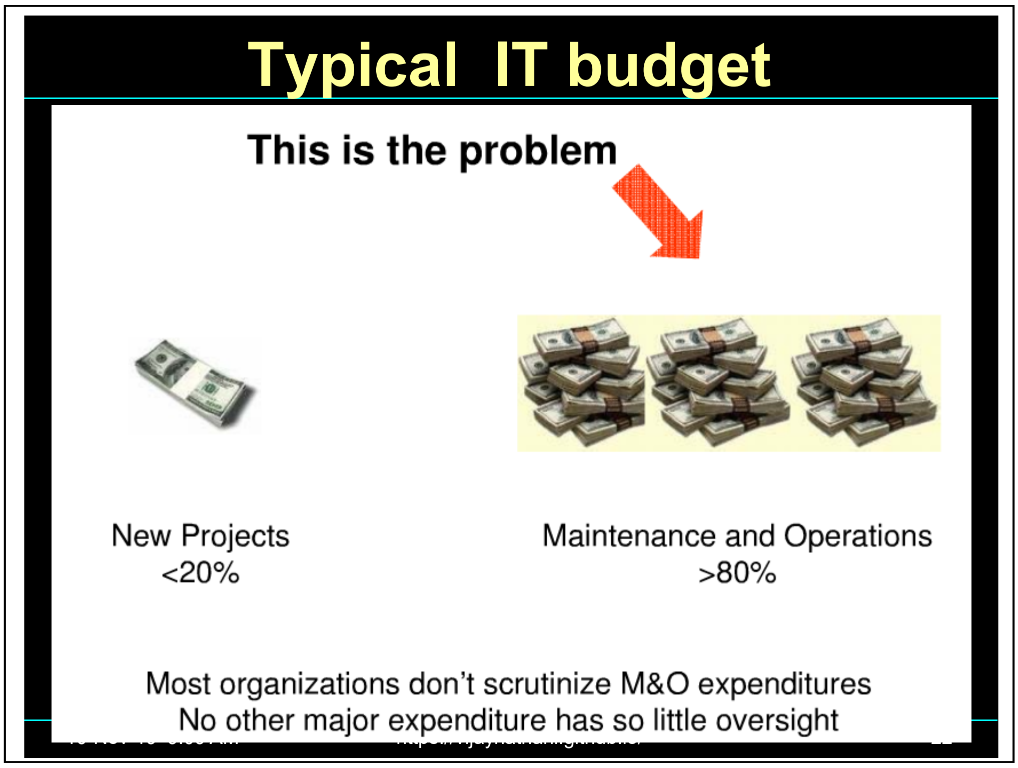


Improve Code

```
// Check to see if the employee
// is eligible for full benefits
if ((employee.flags & HOURLY_FLAG)
       && (employee.age > 65)) …
```

```
if (employee.
        isEligibleForFullBenefits())
```

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 21



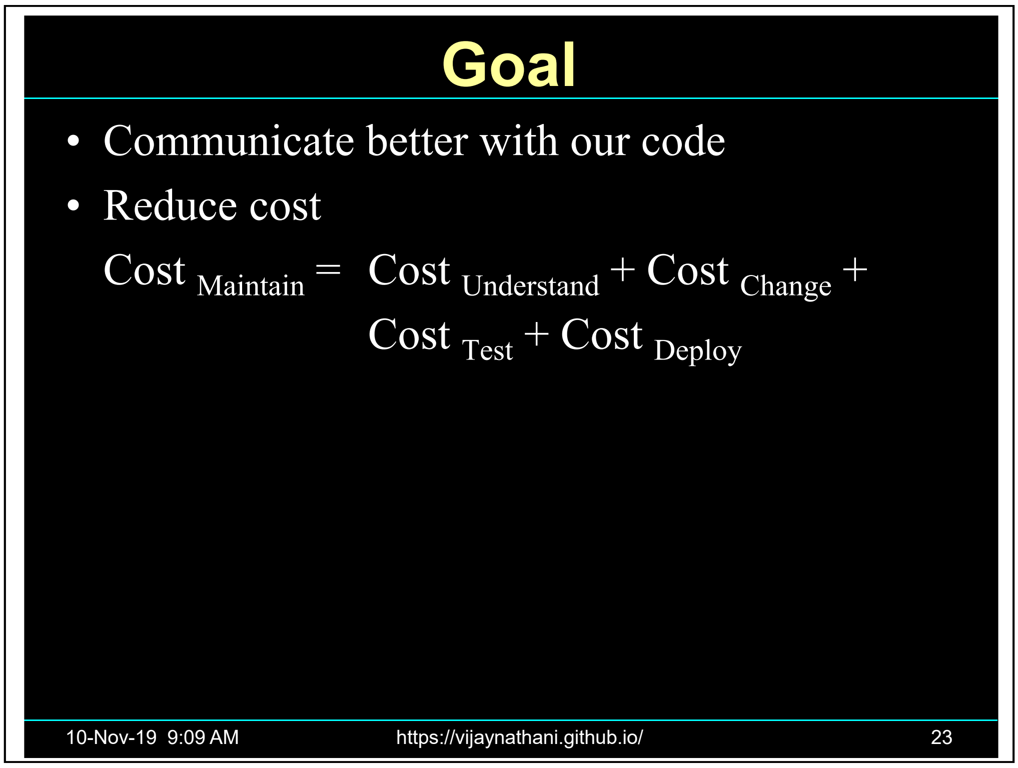
Typical IT budget

This is the problem

New Projects
<20%

Maintenance and Operations
>80%

Most organizations don’t scrutinize M&O expenditures
No other major expenditure has so little oversight



Goal

- Communicate better with our code
- Reduce cost

$$\text{Cost}_{\text{Maintain}} = \text{Cost}_{\text{Understand}} + \text{Cost}_{\text{Change}} + \text{Cost}_{\text{Test}} + \text{Cost}_{\text{Deploy}}$$

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 23

Falls between design patterns and Java language manual

Isolate the concurrent portions of the code.

Most programs follow a small set of laws:

•Programs are read more often than they are written

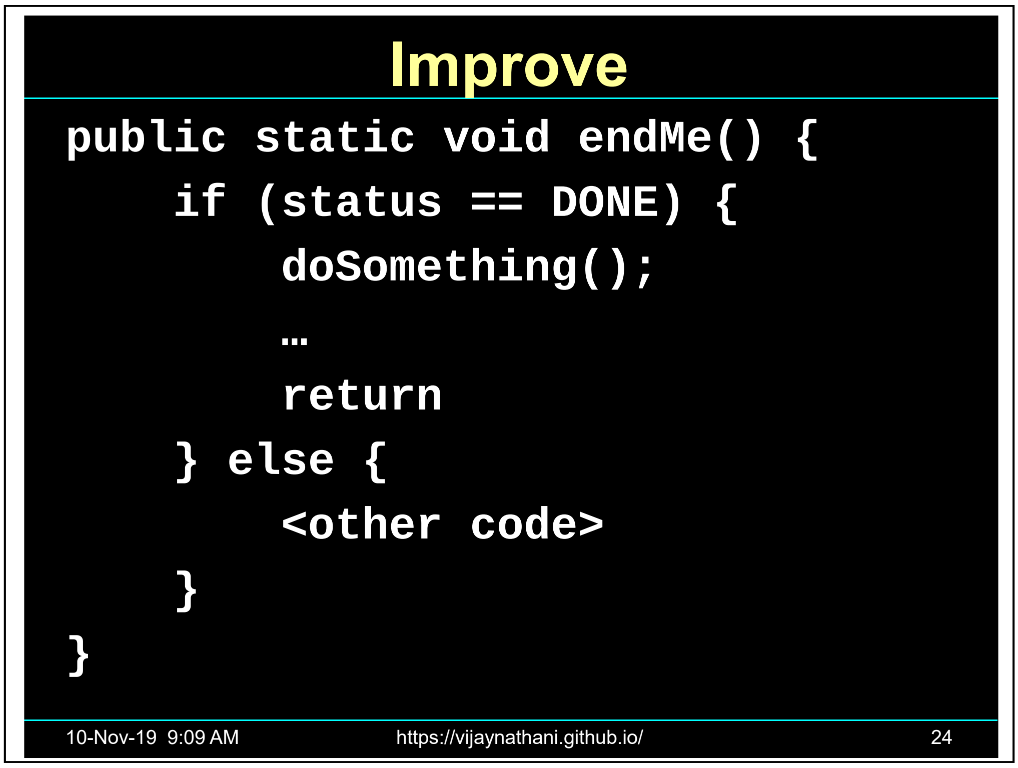
•There is no such thing as “done”. Much more investment will be spent in modifying programs than in developing them initially.

•They are structured using a basic set of state and control flow concepts

•Readers need to understand programs in detail and in concept

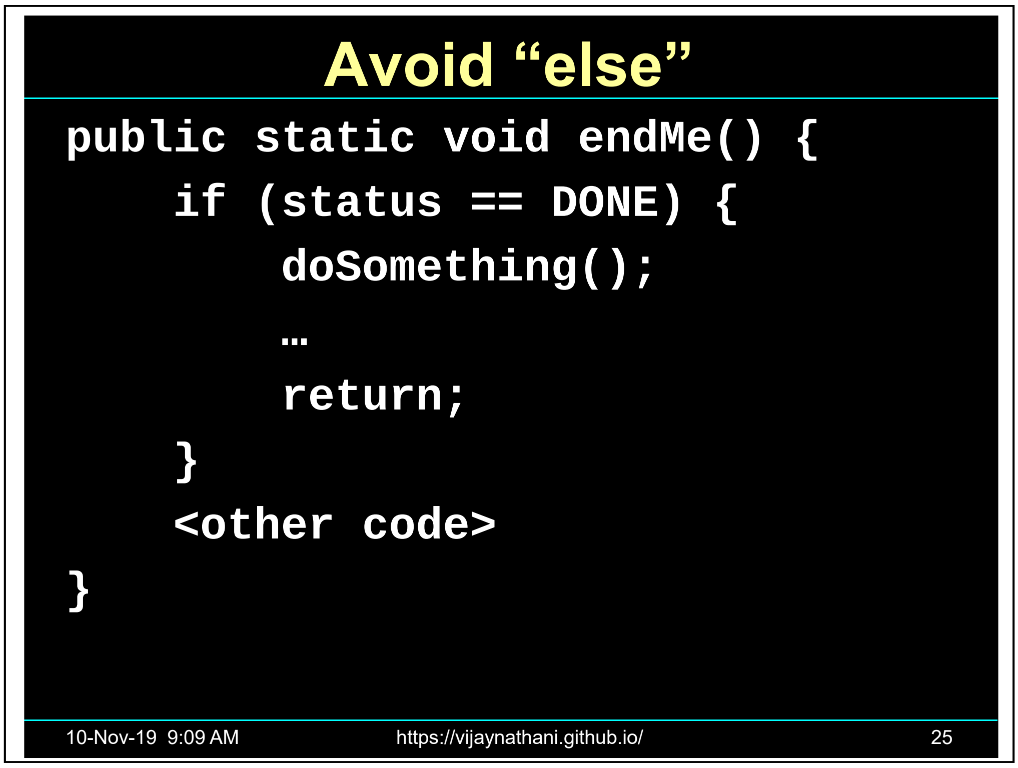
Cost to understand code is high. So maintenance is costly. Code will need to change in unanticipated ways.

When code is clear we have fewer defects and smoother development also.



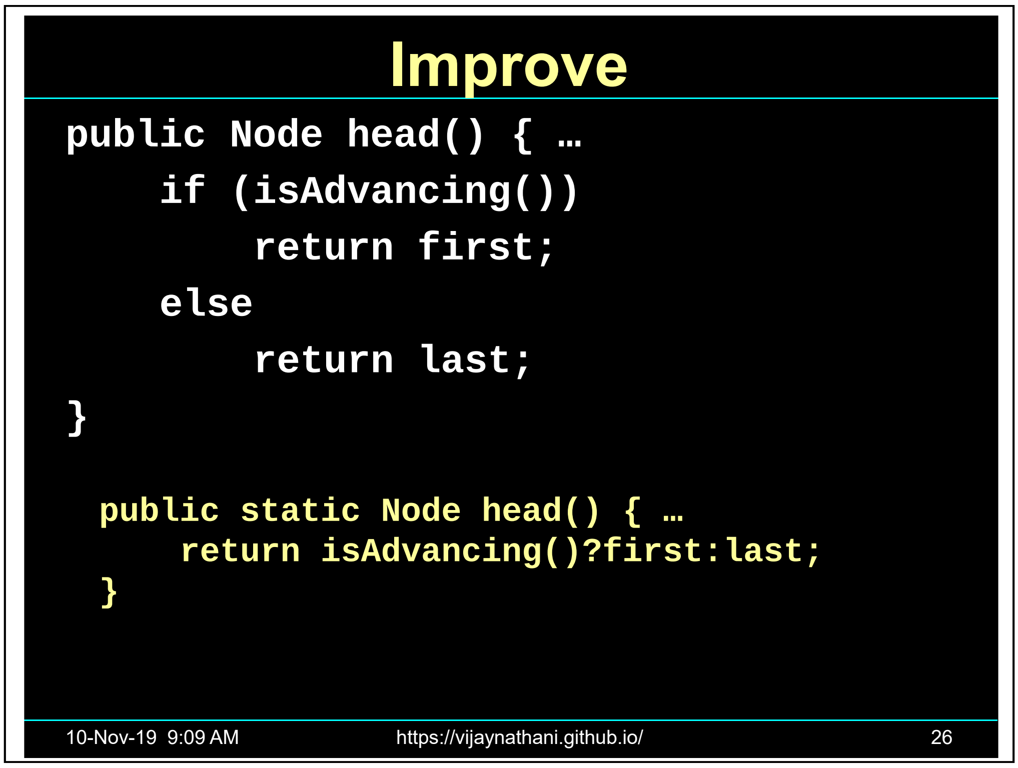
Improve

```
public static void endMe() {
    if (status == DONE) {
        doSomething();
        …
        return
    } else {
        <other code>
    }
}
```



Avoid “else”

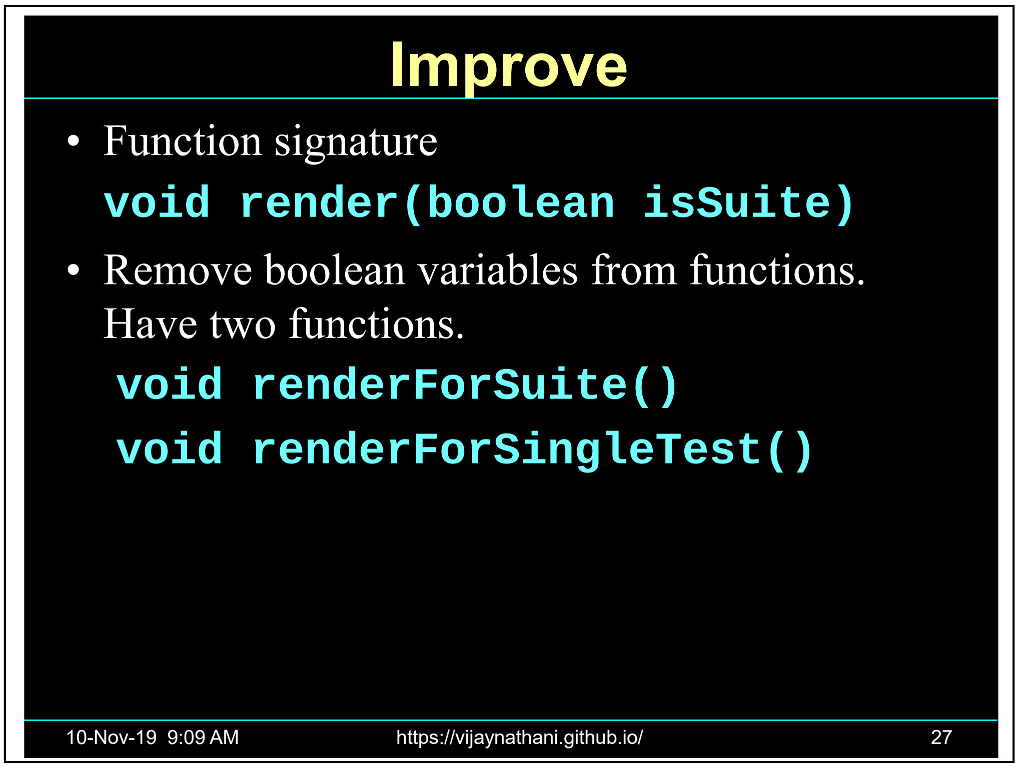
```
public static void endMe() {
    if (status == DONE) {
        doSomething();
        …
        return;
    }
    <other code>
}
```



Improve

```
public Node head() { …
    if (isAdvancing())
        return first;
    else
        return last;
}
```

```
public static Node head() { …
    return isAdvancing()?first:last;
}
```



Improve

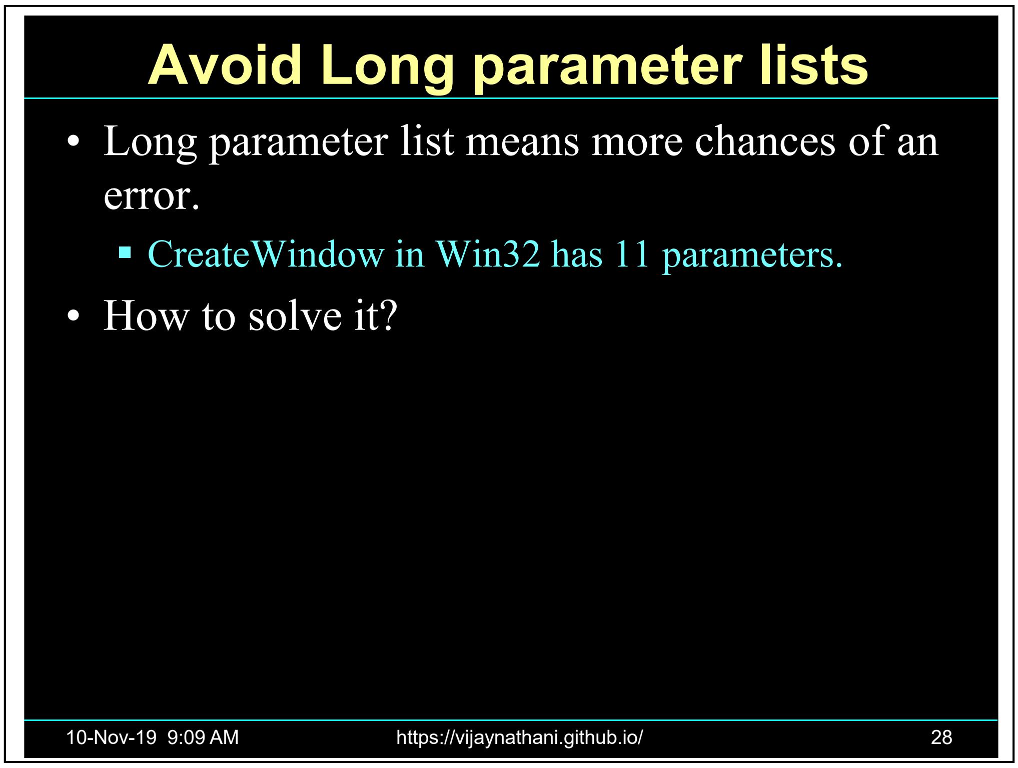
- Function signature

```
void render(boolean isSuite)
```

- Remove boolean variables from functions. Have two functions.

```
void renderForSuite()
void renderForSingleTest()
```

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 27

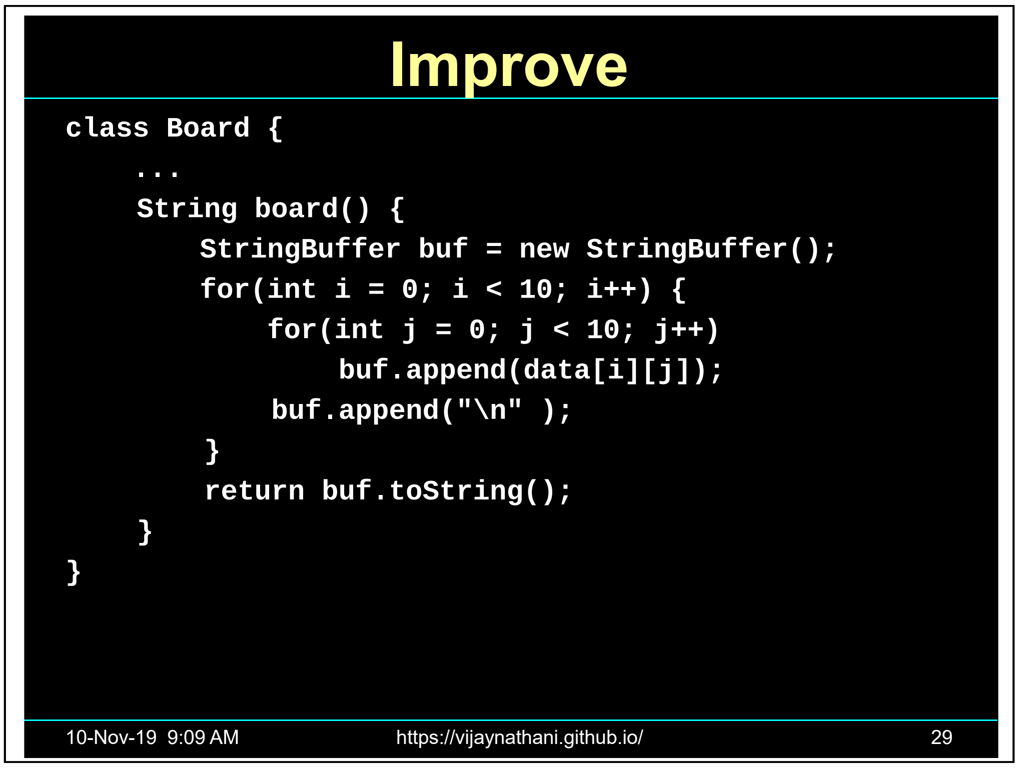

Avoid Long parameter lists

- Long parameter list means more chances of an error.
  - CreateWindow in Win32 has 11 parameters.
- How to solve it?

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 28

Break the method or

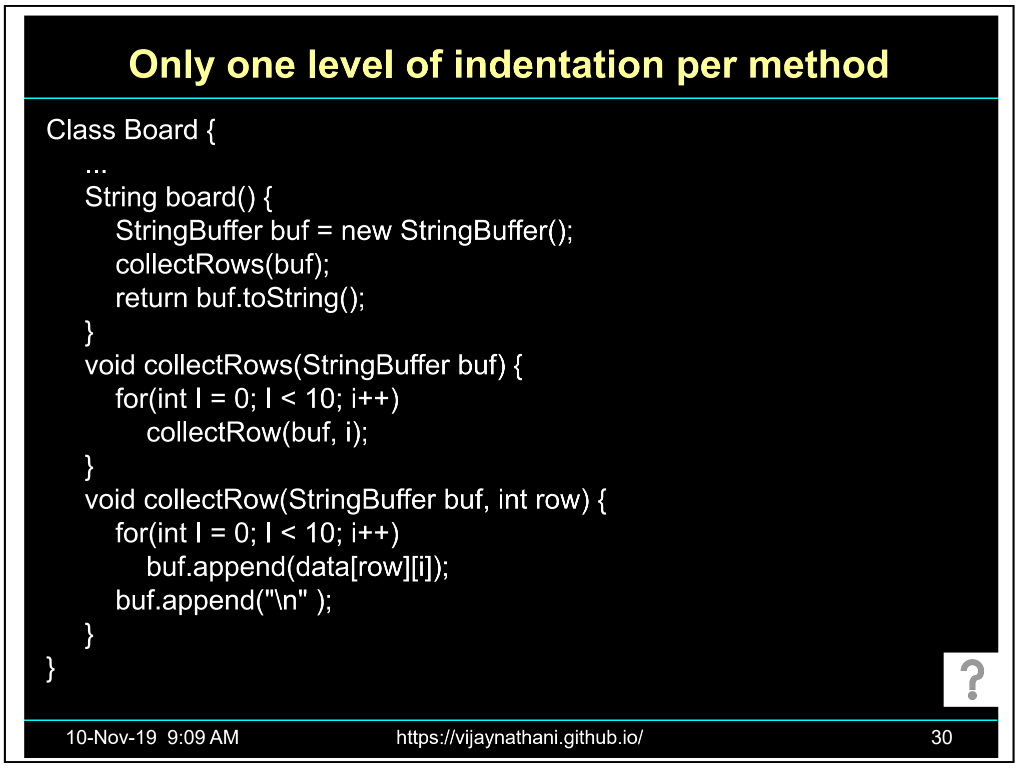
Create Helper class to hold parameters



Improve

```
class Board {
    ...
    String board() {
        StringBuffer buf = new StringBuffer();
        for(int i = 0; i < 10; i++) {
            for(int j = 0; j < 10; j++)
                buf.append(data[i][j]);
            buf.append("\n" );
        }
        return buf.toString();
    }
}
```

10-Nov-19 9:09 AM   https://vijaynathani.github.io/   29



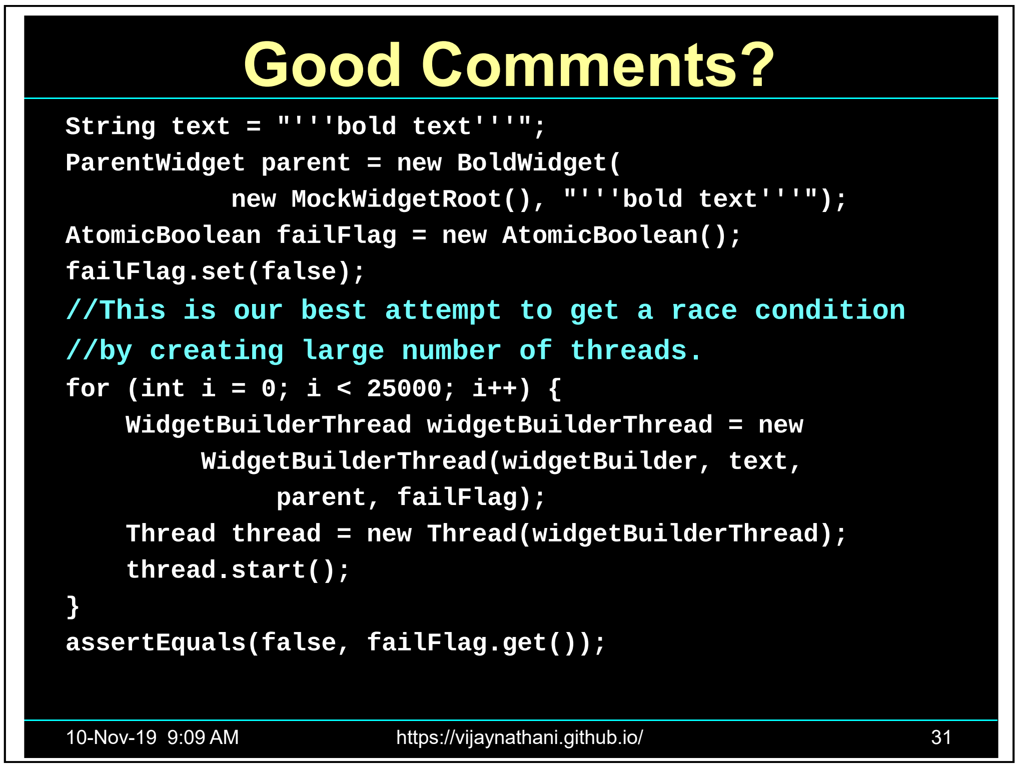
Only one level of indentation per method

```
Class Board {
    ...
    String board() {
       StringBuffer buf = new StringBuffer();
       collectRows(buf);
       return buf.toString();
    }
    void collectRows(StringBuffer buf) {
       for(int I = 0; I < 10; i++)
          collectRow(buf, i);
    }
    void collectRow(StringBuffer buf, int row) {
       for(int I = 0; I < 10; i++)
          buf.append(data[row][i]);
       buf.append("\n" );
    }
}
```

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 30

Self documenting code

Q20 – inch; Q06 – NO_GROUPING; Q07 – addHoliday; Q21 – full name in English; Q22 – complexPassword; Q23 – TokenStream; Q25 - orderItems



Good Comments?

```
String text = "'''bold text'''";
ParentWidget parent = new BoldWidget(
           new MockWidgetRoot(), "'''bold text'''");
AtomicBoolean failFlag = new AtomicBoolean();
failFlag.set(false);
//This is our best attempt to get a race condition
//by creating large number of threads.
for (int i = 0; i < 25000; i++) {
    WidgetBuilderThread widgetBuilderThread = new
         WidgetBuilderThread(widgetBuilder, text,
              parent, failFlag);
    Thread thread = new Thread(widgetBuilderThread);
    thread.start();
}
assertEquals(false, failFlag.get());
```

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 31

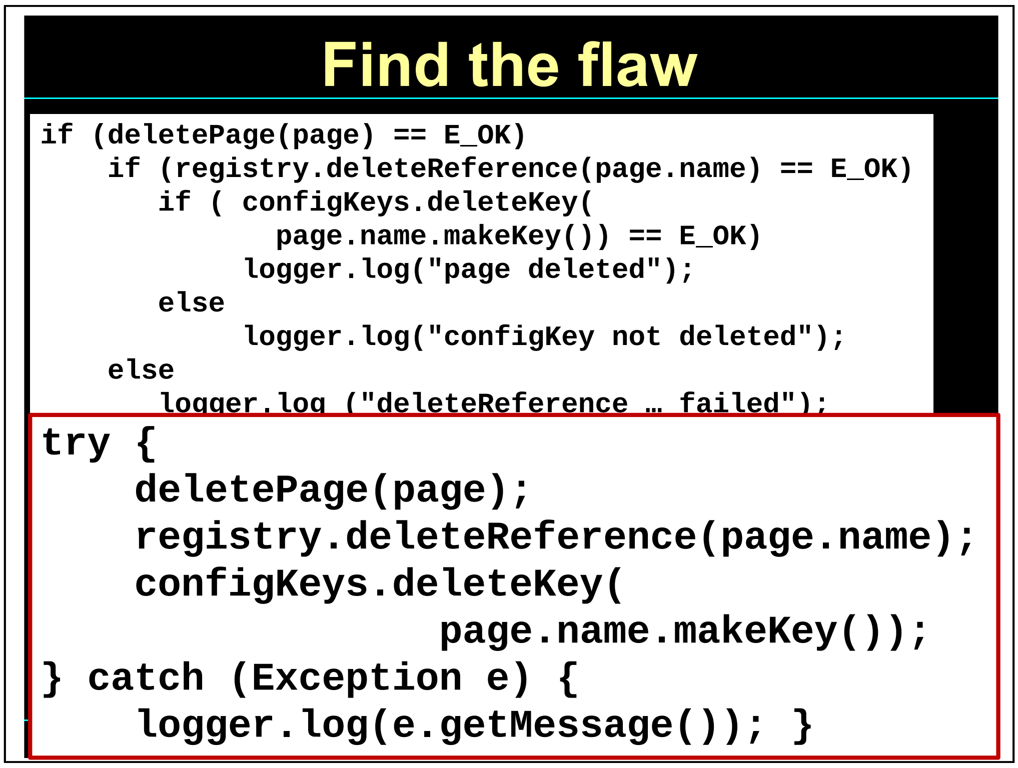
Comment to

• explain WHY we are doing something?

•Also for external documentation. Javadocs public API

•To give warnings: e.g. Don't run unless you want to kill this program.

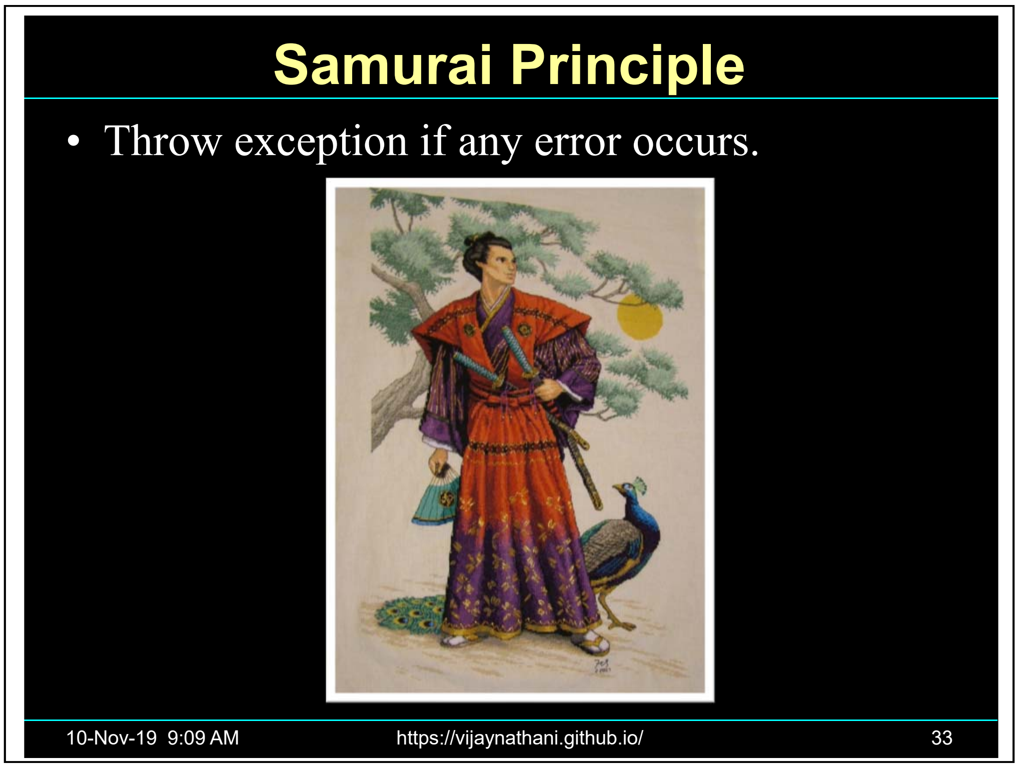
•Todo comments



Find the flaw

```
if (deletePage(page) == E_OK)
    if (registry.deleteReference(page.name) == E_OK)
       if ( configKeys.deleteKey(
              page.name.makeKey()) == E_OK)
            logger.log("page deleted");
       else
            logger.log("configKey not deleted");
    else
       logger.log ("deleteReference … failed");
```

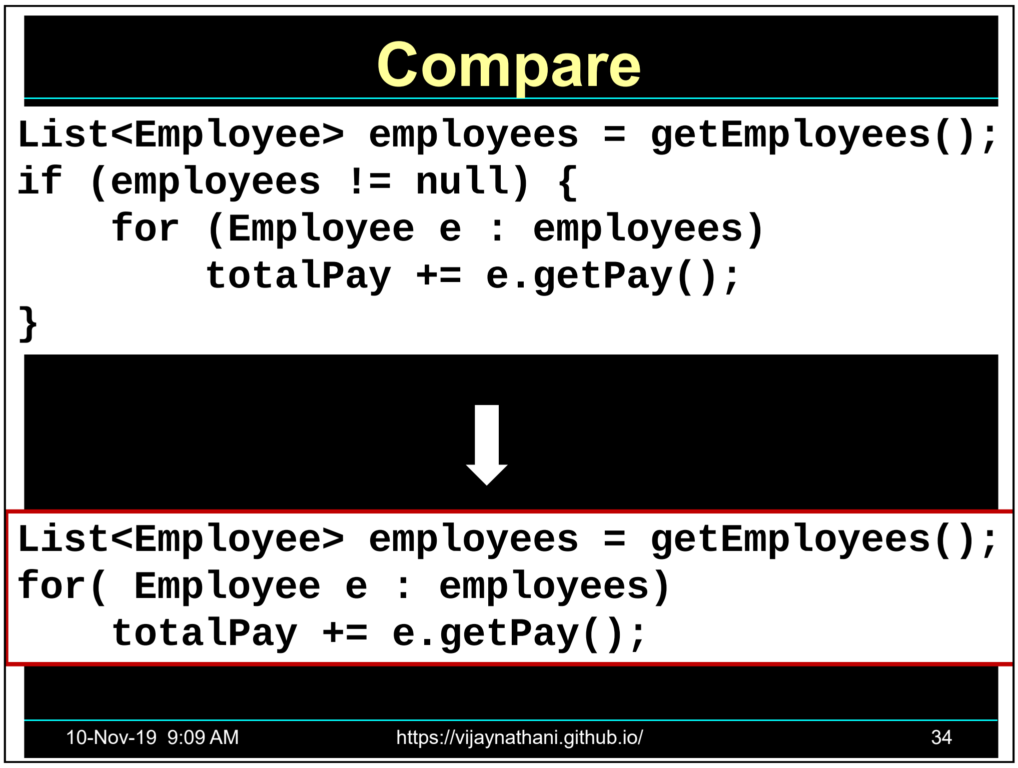
```
try {
    deletePage(page);
    registry.deleteReference(page.name);
    configKeys.deleteKey(
                page.name.makeKey());
} catch (Exception e) {
    logger.log(e.getMessage()); }
```



Samurai Principle

- Throw exception if any error occurs.

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 33

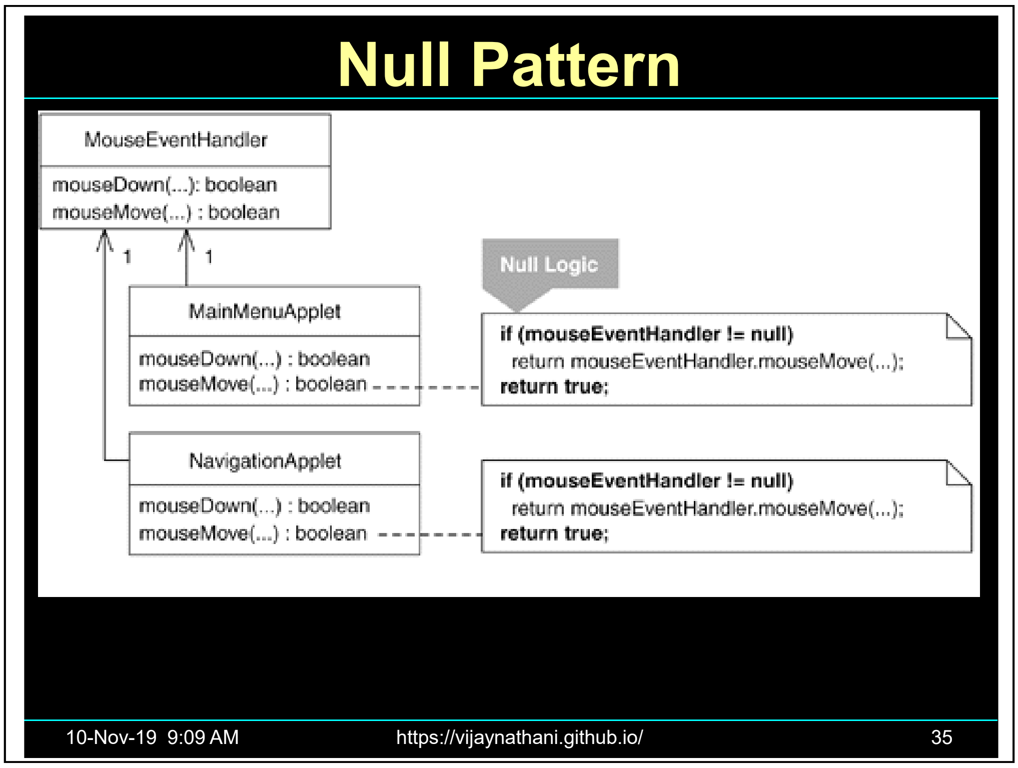


Compare

```
List<Employee> employees = getEmployees();
if (employees != null) {
    for (Employee e : employees)
        totalPay += e.getPay();
}
```

```
List<Employee> employees = getEmployees();
for( Employee e : employees)
    totalPay += e.getPay();
```

Java has Collections.emptyList() for this. It is immutable.



Null Pattern

```
if (mouseEventHandler != null)
  return mouseEventHandler.mouseMove(...);
return true;
```

```
if (mouseEventHandler != null)
  return mouseEventHandler.mouseMove(...);
return true;
```

10-Nov-19 9:09 AM https://vijaynathani.github.io/ 35

Null pattern

Avoid NullPointerException in code

Return “” instead of null for String class

Return an array with zero elements instead of null.

========================

Benefits and Liabilities

+ Prevents null errors without duplicating null logic.

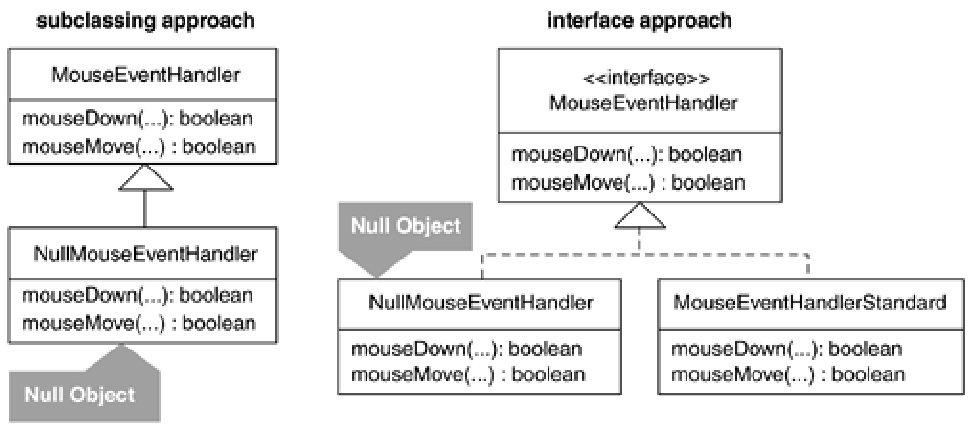
+ Simplifies code by minimizing null tests.

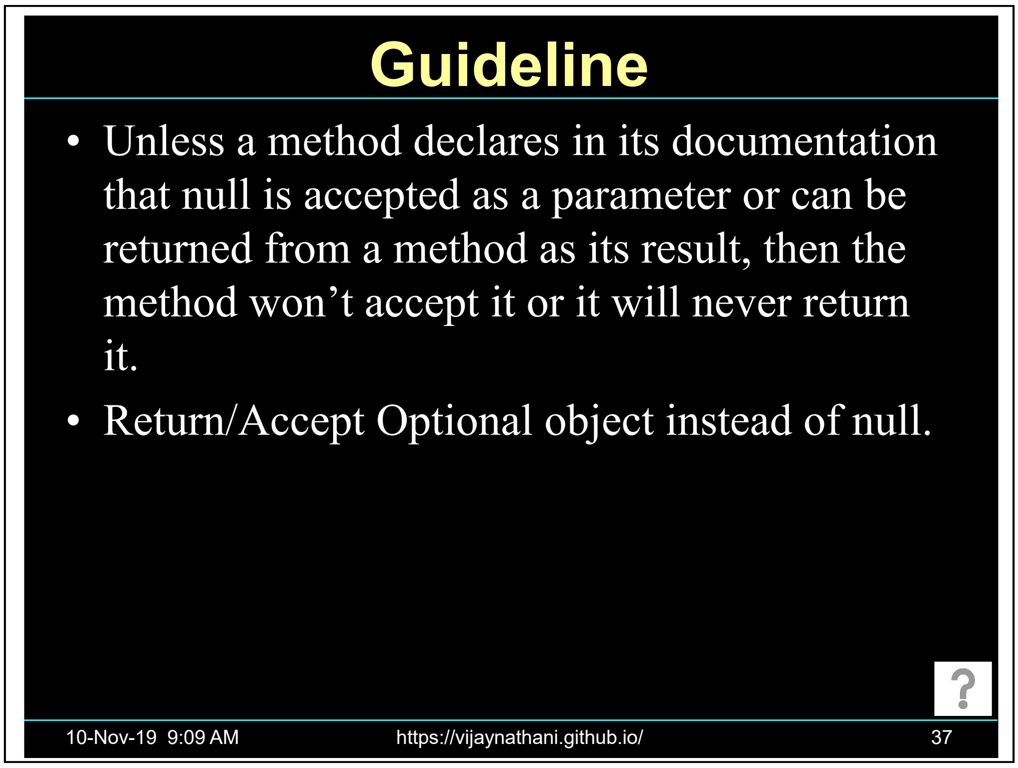
– Complicates a design when a system needs few null tests.

– Can yield redundant null tests if programmers are unaware of a Null Object implementation.

– Complicates maintenance. Null Objects that have a superclass must override all newly inherited public methods.

Null Object by Interface





Guideline

- Unless a method declares in its documentation that null is accepted as a parameter or can be returned from a method as its result, then the method won't accept it or it will never return it.
- Return/Accept Optional object instead of null.

?

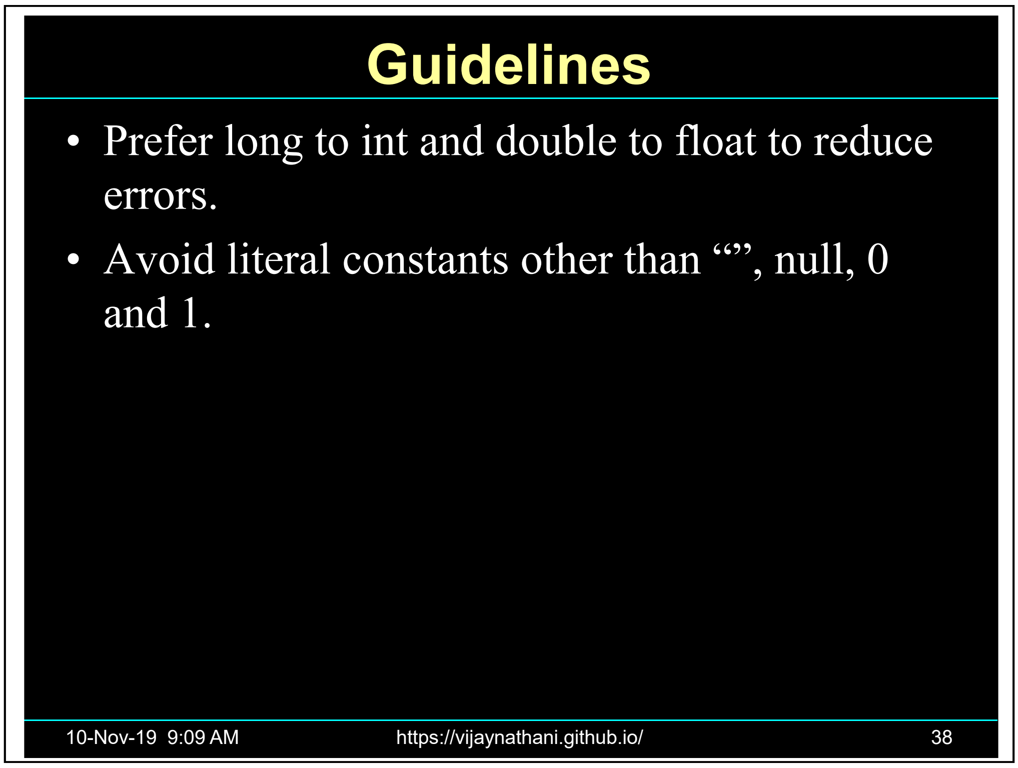
10-Nov-19 9:09 AM https://vijaynathani.github.io/ 37

Q28Artists

Optional class is present in Java 8. If you are using older versions of Java, then it is also present in Guava library.

Optional has been coded for C# and stored in same directory other examples. It is also present is an open source library https://github.com/nlkl/Optional

Optional is present in C++17, not in older versions



Guidelines

- Prefer long to int and double to float to reduce errors.
- Avoid literal constants other than “”, null, 0 and 1.