SE 333/433 Software Testing & Quality Assurance

Code-smells and bad-design practices

Last Week

McCall's Model

Use		Factor (11)		Criteria (25)		(41)

Product operation — Usability, Integrity, Efficiency, Correctness, Reliability

Product revision — Maintainability, Testability, Flexibility

Product transition — Reusability, Portability, Interoperability

Criteria: Operability, Training, Communicativeness, I/O volume, I/O rate, Access control, Access audit, Storage efficiency, Execution efficiency, Traceability, Completeness, Accuracy, Error tolerance, Consistency, Simplicity, Conciseness, Instrumentation, Expandability, Generality, Self-descriptiveness, Modularity, Machine independence, S/w system independence, Comms commonality, Data commonality

Metrics

Chidamber and Kemerer OO Metrics (CK - 1994)

- Weighted methods per class (MWC)
- Depth of Inheritance tree (DIT)
- Number of children (NOC)
- Coupling between object classes (CBO)
- Response for class (RFC)
- Lack of cohesion metric (LCOM)

28

McCall's quality factors

Product Operation: Correctness, Reliability, Usability, efficiency, Integrity

McCall Software Quality Factors

Product Revision: Testability, Flexibility, Maintainability

Product Transition: Portability, Reusability, Interoperability

6

2

Functional vs Non-Functional Requirements

Functional Requirements

Non-Functional Requirements

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Functionality of the program
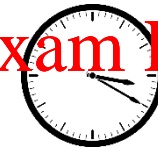
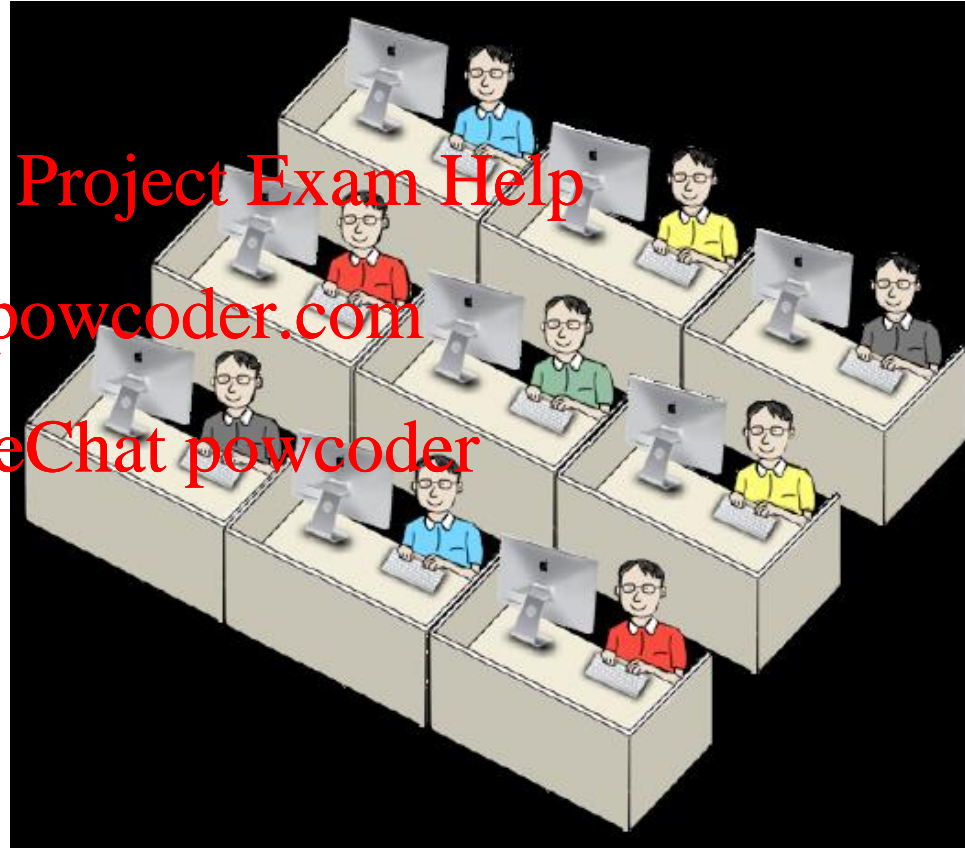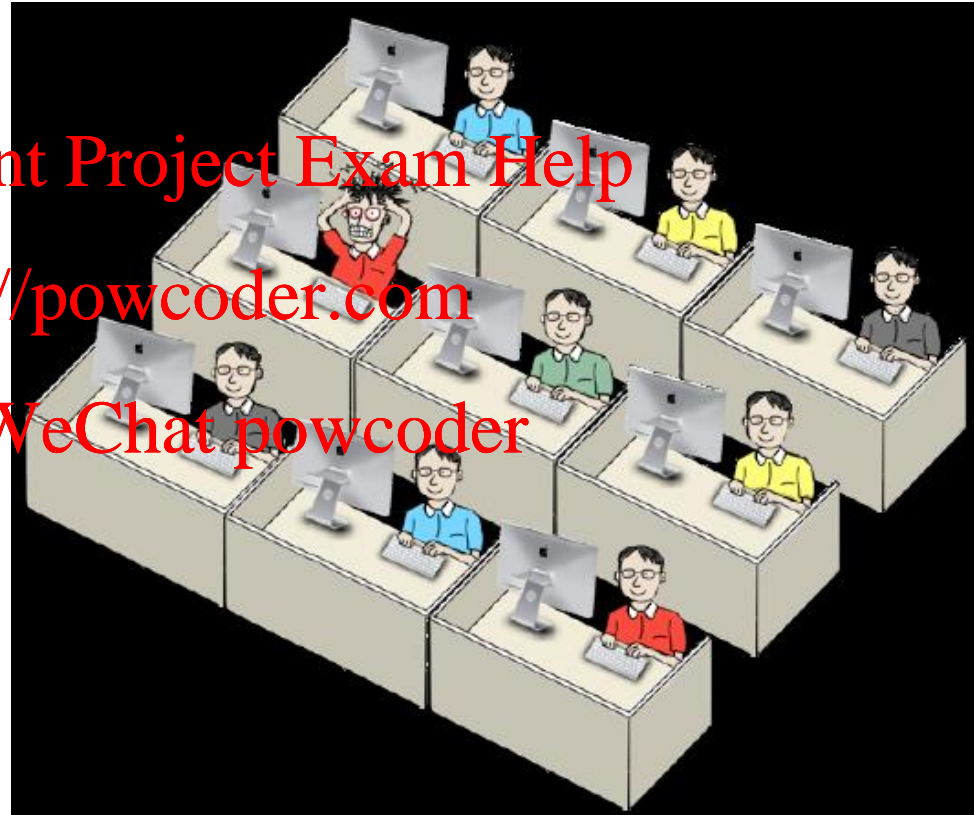Execution Time

Size

Battery

Memory

- Text chat
- Friends list

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- Text chat
- Friends list

- Voice Communication

Assignment Project Exam Help

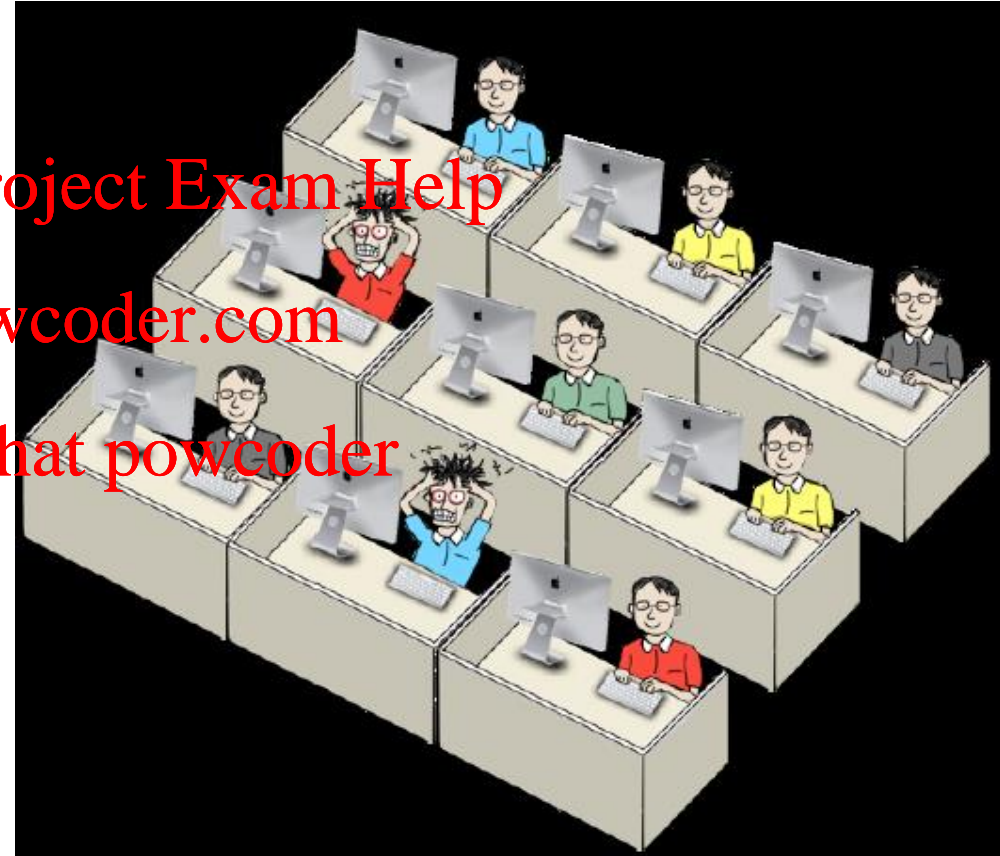https://powcoder.com

Add WeChat powcoder

- Text chat
- Friends list
- Voice Communication

- Text chat
- Friends list

- Voice
  Communication

-Execution time

-Battery consumption
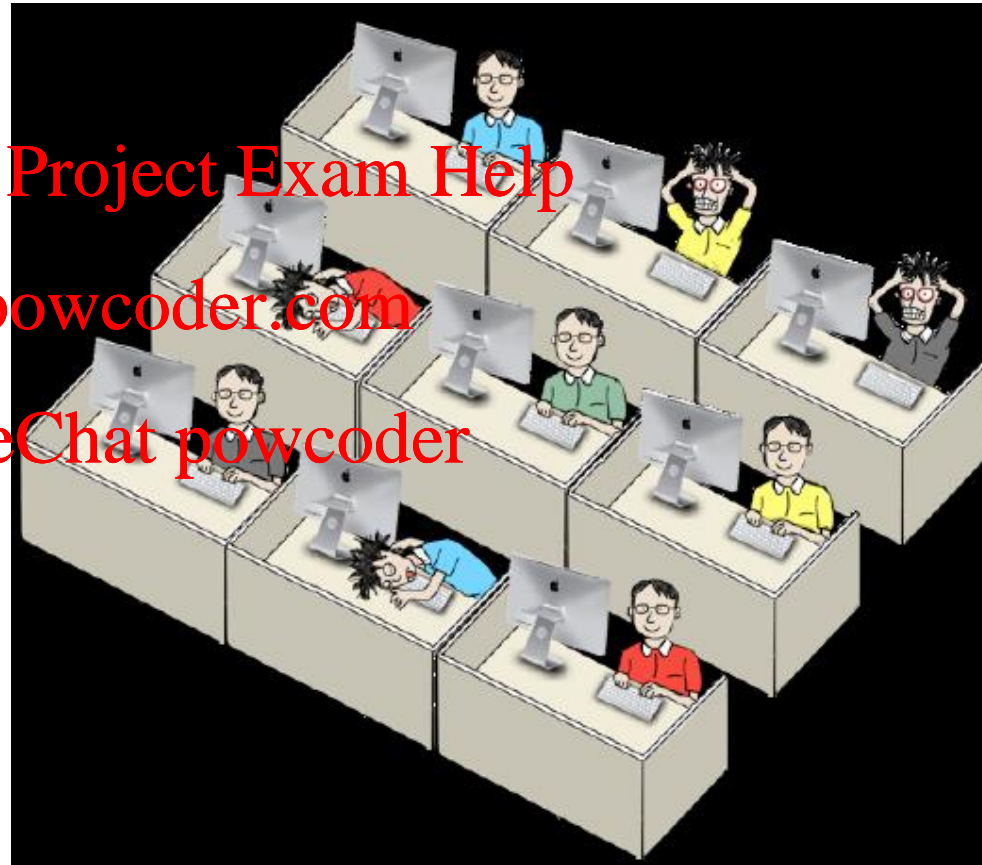
-Size

- Text chat
- Friends list

- Voice Communication

-Execution time

-Battery consumption

-Size

-Memory leak

Functional Requirements
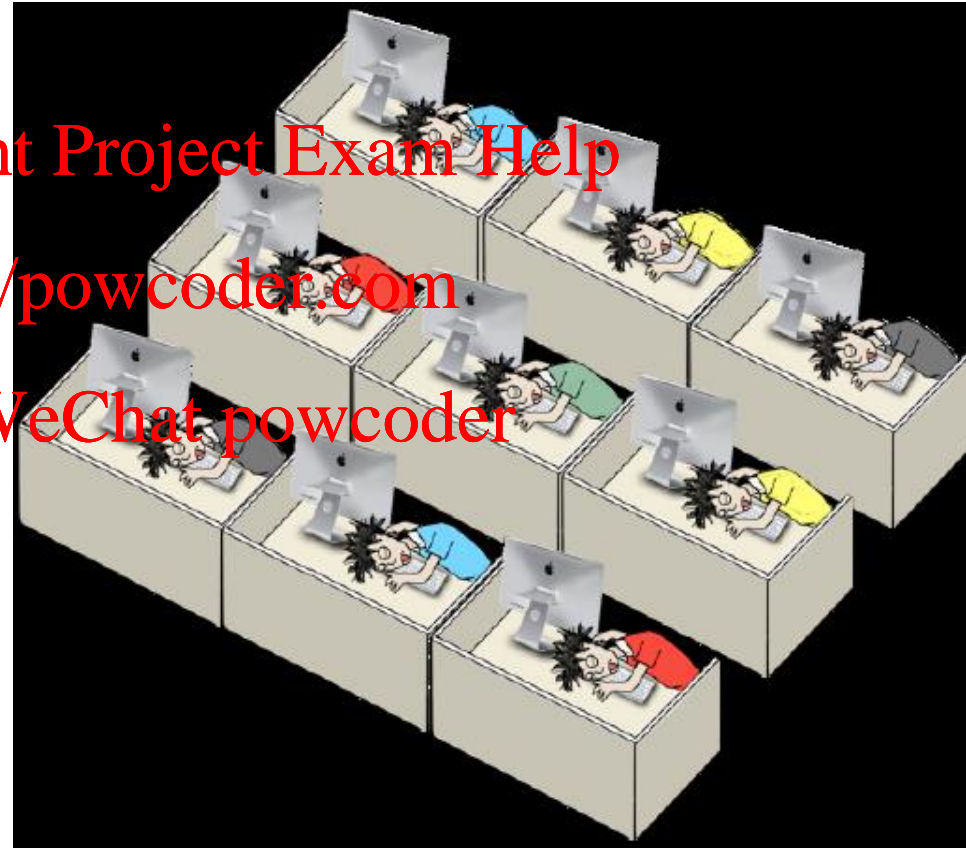
Non-Functional Requirements

humans have to define these

a machine can optimize these

Functional Requirements

Non-Functional Requirements

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

humans have to define these

a machine can optimize these

- Software Refactoring:
  - Antipatterns Detection
  - Antipatterns Correction (Refactoring Recommendation)

```
2  public class PassByReference{
3      public static class Holder{
4          public String value;
5          public Holder( String value ){
6              this.value = value;
7          }
8      }
9
10     public static void passByReference( Holder holder ){
11         holder = new Holder( "test" );
12     }
13
14     public static void main( String[] args ){
15         Holder holder = new Holder( "unchanged" );
16         passByReference( holder );
17         System.out.println( holder.value );
18     }
19 }
```

Another System Model (Data-Bases)

Assignment Project Exam Help

Forward engineering

https://powcoder.com

Add WeChat powcoder

Program (in Java)

Refactoring

Model transformation

Reverse engineering

System Model (in UML)

Another Program (Java)

Model space

Source code space

- # Refactoring

  - The process of improving a code after it has been written by changing its internal structure without changing the external behavior (Fowler et al., '99)

  - Examples: *Move method, extract class, move attribute, …*

- # Two main refactoring steps

  1. detection of code fragments to improve (e.g., Anti pattern)
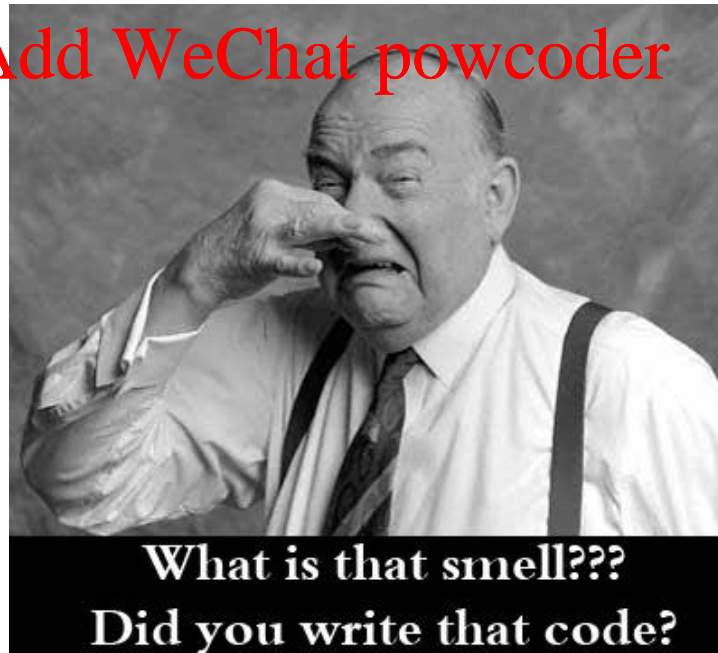 2. identification of refactoring solutions

Learning objectives

- Anti-patterns: Overview

- Examples of anti-patterns

- How to detect anti-patterns using quality metrics?

What is that smell???
Did you write that code?
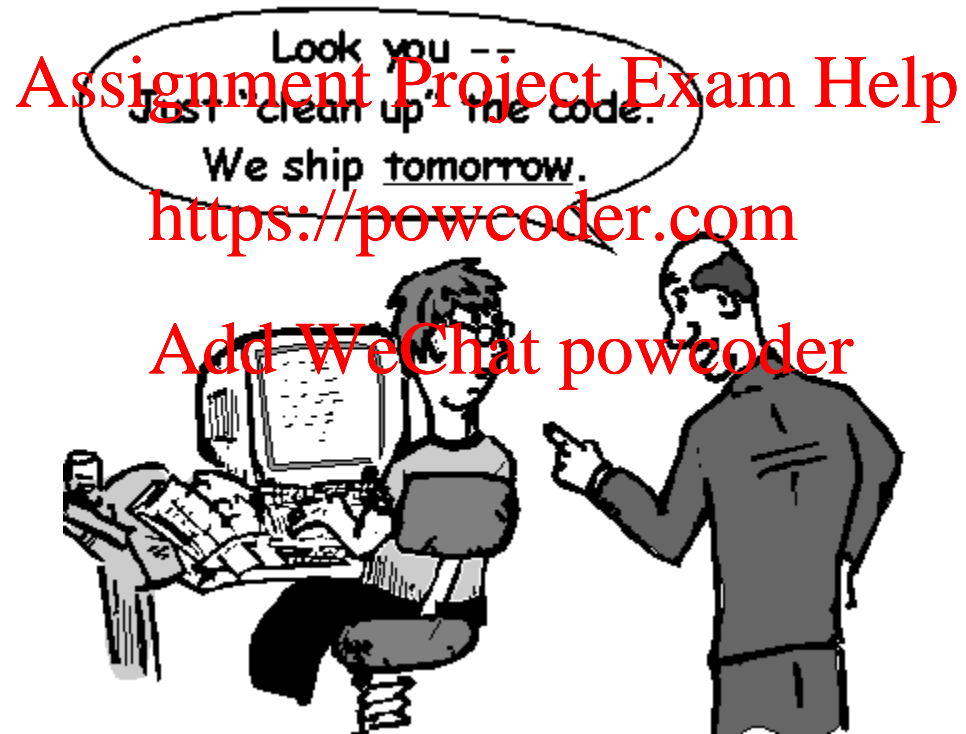
Design defects/antipatterns

- Design defects/antipatterns are poor coding and design choices introduced during different phases of software development

  - Anomalies, code smells, bad smells...

  - Make the design harder to understand, to change

  - Design situations that adversely affect the development of a software (**not bugs**)

Management Antipatterns

Examples of antipatterns

- Duplicated code
- Long method
- God class
- Long parameter list
- Message chain
- Data class
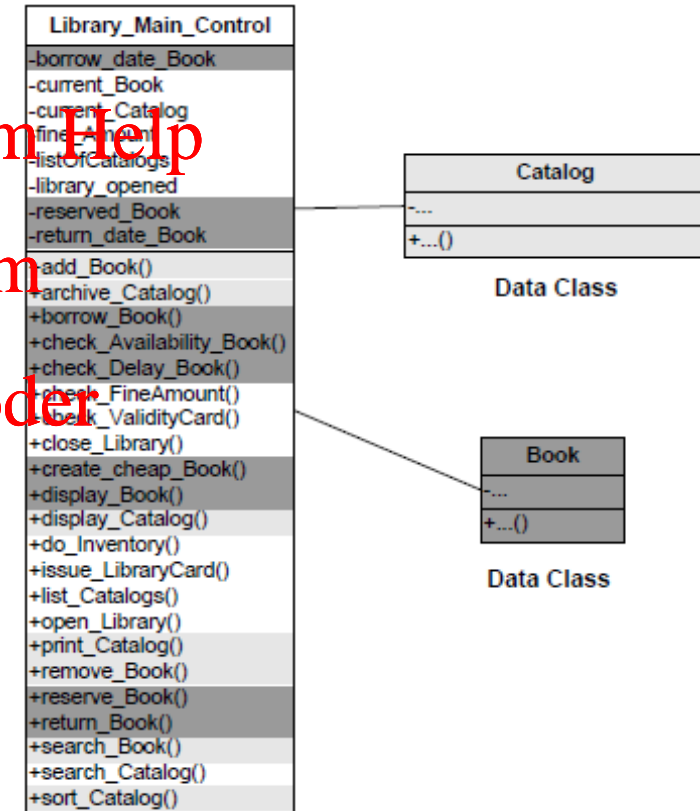- Functional decomposition
- ...

The Blob (God Class)

- ## Definition

  - Procedural-style design leads to one object with numerous responsibilities and most other objects only holding data or executing simple processes.

- ## Symptoms

  - A Blob is a *controller* class, *abnormally large*, with *almost* no parents and no children. It *mainly uses* data classes, i.e. *very small* classes with *almost* no parents and no children (Brown et al. '98).

Library_Main_Control
-borrow_date_Book
-current_Book
-current_Catalog
-fine_Amount
-listOfCatalogs
-library_opened
-reserved_Book
-return_date_Book
+add_Book()
+archive_Catalog()
+borrow_Book()
+check_Availability_Book()
+check_Delay_Book()
+check_FineAmount()
+check_ValidityCard()
+close_Library()
+create_cheap_Book()
+display_Book()
+display_Catalog()
+do_Inventory()
+issue_LibraryCard()
+list_Catalogs()
+open_Library()
+print_Catalog()
+remove_Book()
+reserve_Book()
+return_Book()
+search_Book()
+search_Catalog()
+sort_Catalog()

Catalog
-...
+...()
Data Class

Book
-...
+...()
Data Class

- Causes
  - *Lack of an object-oriented architecture*
    - *Inadequate understanding of OO principles*
  - *Lack of (any) architecture*
    - *Design, interaction between object etc.*
  - *Too limited intervention*

Duplicate Code or
Cut and Paste Programming

- Code reused by copying source statements leads to significant maintenance problems.
  - Duplicate methods in subclasses
  - Duplicate expressions in same class
  - Duplicate expressions in different classes

```ruby
def to_time(value)
  if value.respond_to?(:to_time)
    value.to_time
  else
    Coercion::String.to_time(to_string(value))
  end
end


def to_datetime(value)
  if value.respond_to?(:to_datetime)
    value.to_datetime
  else
    Coercion::String.to_datetime(to_string(value))
  end
end
```

```php
public function add()
{
    if($this->input->post("departmentName"))
    {
        $departmentName = $this->input->post("departmentName");
        $departmentBudget = $this->input->post("departmentBudget");

        $this->departments_m->addDepartment($departmentName, $departmentBudget);
        redirect("/departments");
    }

    $data = array('title' => 'Add Department - DB Hotel Management System', 'page' => 'departments');
    $this->load->view('header', $data);
    $departments = $this->departments_m->get_departments();
    $viewdata = array('departments' => $departments);
    $this->load->view('departments/add', $viewdata);
    $this->load->view('footer');
}
```

```php
public function edit($department_id)
{
    if($this->input->post("departmentName"))
    {
        $departmenName = $this->input->post("departmentName");
        $departmentBudget = $this->input->post("departmentBudget");

        $this->departments_m->editEmployee($department_id, $departmenName, $departmentBudget);
        redirect("/departments");
    }

    $data = array('title' => 'Edit Department - DB Hotel Management System', 'page' => 'departments');
    $this->load->view('header', $data);
    $department = $this->departments_m->getDepartment($department_id);
    $viewdata = array('department' => $department[0]);
    $this->load->view('departments/edit', $viewdata);
    $this->load->view('footer');
}
```

21

Large method

- A method that does more than one thing
  - Many things, sometimes unconnected things
- Problems
  - Could indicates low levels of abstraction, low level of class design, reduced re-usability
  - Harder to test, poor readability

Long Parameter List

- Introduce parameter object
- Only worthwhile if there are several methods with same parameter list, and they call each other

user= userManager.create(USER_NAME,group, USER_NAME, "test", Language, false, false, new Date(), "blah", "new Date())

Message Chain

- Long list of method calls:
  - customer.getAddress().getState()
  - window.getBoundingbox().getOrigin().getX()

Message Chain

```
Class Employee{
    public function getConfiguration(){
        $this->employeeConfig->getConfiguration();
    }
}

Class EmployeeConfig{
    public function getConfiguration(){
        $this->config->getConfiguration();
    }
}

Class Config{
    public function getConfiguration(){
        $this->loadConfiguration();
    }
}
```

Data Class/Lazy Class

- Class has no methods except for getter and setters

```
public class Example implements InterfaceExample {
    @Override
    public void doSomething() {
        // Input text is put here.
    }
}
```

Switch statements

- Switch statements are very rare in properly designed object-oriented code
  - Therefore, a switch statement is a simple and easily detected "bad smell"
  - Of course, not all uses of switch are bad
  - A switch statement should *not* be used to distinguish between various kinds of object

Example 1, continued

- class Animal {
    final int MAMMAL = 0, BIRD = 1, REPTILE = 2;
    int myKind;

    ...
    String getSkin() {
        switch (myKind) {
            case MAMMAL: return "hair";
            case BIRD: return "feathers";
            case REPTILE: return "scales";
            default: return "skin";
        }
    }
}

Example 1, improved

```
class Animal {
    String getSkin() { return "skin"; }
}
class Mammal extends Animal {
    String getSkin() { return "hair"; }
}
class Bird extends Animal {
    String getSkin() { return "feathers"; }
}
class Reptile extends Animal {
    String getSkin() { return "scales"; }
}
```

Dead Code/
Old Baggage

- Description
  - System contains many classes whose purpose is not known
    - Lava Flow, **Dead Code**
  - Much of the code is left over from previous ideas and no longer has a purpose
    - was once fluid and useful, now is solid lava that you are afraid to remove

- Consequences
  - difficult to maintain, just gets worse

Comments?

- The purpose of comments should be only "why you are doing something (to help future modifiers   rather than "what code is doing

- Whenever possible   express the intent of the comment and remove the comment.

- Comments are to provide intent that is not expressible in code

- Any comment that duplicates what the code says should be deleted

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

Assignment Project Exam Help

https://powcoder.com

```
public void add(Object element) {
    if (!readOnly) {
        int newSize = size + 1;
        if (newSize > elements.length) {
            grow();
        }
        elements[size++] = element;
    }
}
```
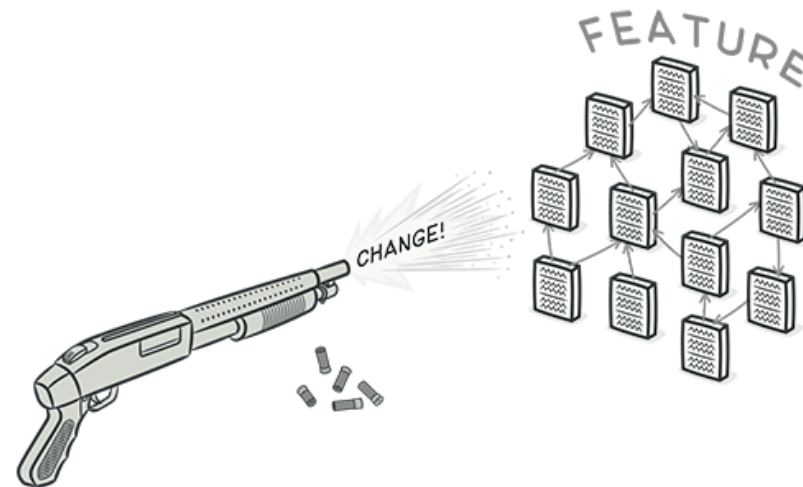
Add WeChat powcoder

- This happens when, you want to make some kind of change, your are forced to figure out lot of changes to a lot of different classes

- And when changes are all over the place, they are hard to find, and it's easy to miss an important change

Functional Decomposition

- Description
  - Classes with names ''functions''.
  - All class attributes are private and are used only inside the class
  - Classes with a single action similar to a procedural function.

- Consequences
  - No O/O benefits such as inheritance and polymorphism
    - Expensive to maintain
    - Complexity of testing software
    - Complexity of reuse of the code

Spaghetti code

- Description
  - System hard to debug, modify
  - Bunch of code similar in structure to a bowl of spaghetti.
    - Bad coding practices
- Consequences
  - Low readability
  - Impossible to understand how it exactly works

- When a method seems more interesting in a class, other than the one in actually it is.

- Example : a method that invokes half a dozen getting methods on another object to calculate some value.

```php
class User
{
    private $contactInfo;

    public function __construct()
    {
        $this->contactInfo = new ContactInfo();
    }

    public function getFullAddress()
    {
        $address = $this->contactInfo->getStreetName();
        $address .= ' ' . $this->contactInfo->getStreetNumber() . ', ';
        $address .= $this->contactInfo->getZipCode() . ', ';
        $address .= $this->contactInfo->getCity() . ', ';
        $address .= $this->contactInfo->getCountry();

        return $address;
    }
}
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Class Activity

Assignment Project Exam Help

See additional file added on d2l

https://powcoder.com

Add WeChat powcoder