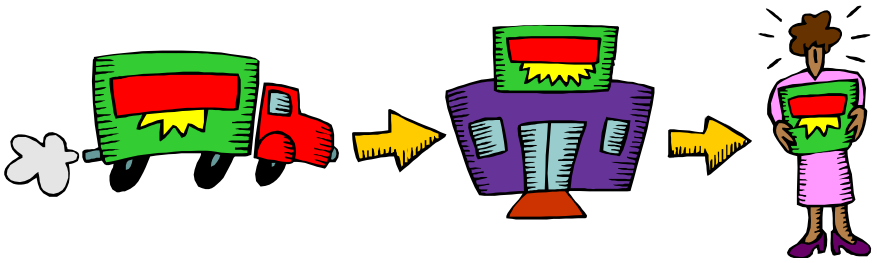
Công nghệ phần mềm

Cài đặt phần mềm
(Implementation)



Nội dung

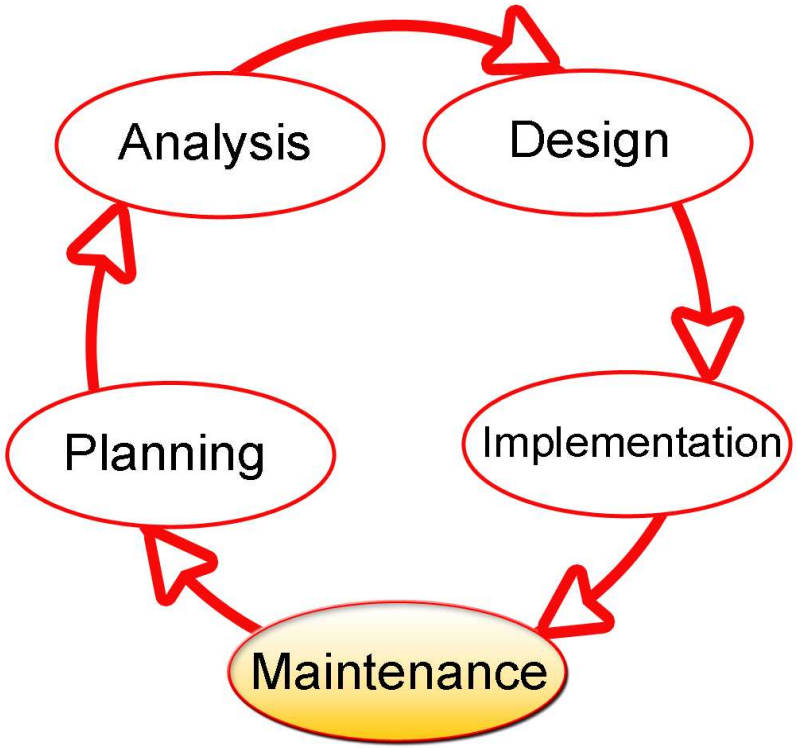
- Giới thiệu
- Các yêu cầu viết mã nguồn chương trình
- Phong cách lập trình
- Chú thích
- Tái sử dụng mã nguồn
- Gỡ lỗi

Giới thiệu

Implementation = triển khai thiết kế chi tiết thành chương trình

➔Sản phẩm phần mềm tốt, hiệu quả kinh tế cao

- Hạn chế tối đa xảy ra lỗi
- Mã nguồn dễ bảo trì: dễ hiểu, dễ sửa lỗi được, nâng cấp – thay đổi dễ dàng.
- Khả năng tái sự dụng cao



Kỹ thuật lập trình tốt, hiệu quả

Các yêu cầu viết mã nguồn chương trình

- Kỹ thuật lập trình chuyên nghiệp
  - Tuân theo các chuẩn viết mã nguồn (coding styles, coding convention, programming styles)
  - Các chuẩn quy định do Ngôn ngữ lập trình, do Công ty
- Kỹ thuật lập trình hiệu quả
  - Dễ dàng bảo trì: dễ hiểu, dễ sửa lỗi
  - Khả năng tái sử dụng cao: nâng cấp, thay đổi

    ➔ Chú thích rõ ràng, đầy đủ
  - Sử dụng các cấu trúc an toàn
    - bắt lỗi, xử lý ngoại lệ
    - mẫu thiết kế

Phong cách lập trình

Đặt tên biến, tên hàm

static boolean isUppercase(char ch);

static boolean isLowercase(char ch)

static boolean isDigit(char ch);

static boolean isLetter(char ch)

static boolean isLetterOrDigit(char ch);

static char toUpperCase(char ch)

static char toLowerCase(char ch);

boolean equals(String)
boolean equalsIgnoreCase(String)
boolean startWith(String)
boolean endWith(String)
int compareTo(String)

Integer k = Integer.valueOf("12"); int i = k.intValue();

Hằng số: Integer.MAX_VALUE, Integer.MIN_VALUE

Phong cách lập trình

Đặt tên biến, tên hàm

- Tên biến, tên hàm có nghĩa, gợi nhớ
  - Sử dụng các ký hiệu, từ Tiếng anh có nghĩa
  - Làm cho dễ đọc, dễ hiểu
    - Thí dụ: dateOfBirth hoặc date_of_birth
    - Không viết dateofbirth
- Tránh đặt tên quá dài
  - Tránh đặt tên dài với biến cục bộ
- Thống nhất cách dùng
  - Tên lớp bắt đầu bằng chữ hoa
  - Tên hằng số toàn chữ hoa
  - Biến vòng lặp, chỉ số: **i** (**i**teration, **i**ndex)
  - Dùng tiền tố để chỉ kiểu dữ liệu: i.e., sName (string), iCount (int), dTotal (double)

Phong cách lập trình

Câu lệnh

- Câu lệnh phải mô tả cấu trúc
  - Tụt lề, dễ đọc, dễ hiểu
- Làm đơn giản các câu lệnh
  - Mỗi lệnh trên 1 dòng
  - Triển khai các biểu thức phức tạp
  - Hạn chế truyền tham số là kết quả của hàm, biểu thức:
- Tránh các cấu trúc phức tạp
  - Các lệnh if lồng nhau
  - Điều kiện phủ định if not

Ví dụ - Viết câu lệnh

Trên 1 dòng

```
if (g < 17 && h < 22 || i < 60) { return true; } else {System.out.println (“incorrect”) ; return false; }
```

Thay bằng

```
if ((g < 17 && h < 22) ||  i < 60)) {
        return true;
} else {
        System.out.println(“incorrect”);
        return false;
}
```

- Dễ đọc
- Dễ hiểu
- **Dễ dàng bảo trì**

Ví dụ - Viết câu lệnh

```
//Arithmetic (Expression e1, Expression e2, Operand o)
Arithmetic exp1 = new Arithmetic ( //3x^2y - xy
      new Arithmetic (
            new Const(3),
            new Arithmetic (
                  new Exp(x, 2),
                  y,
                  '*'
            ),
            '*'
      ),
      new Arithmetic (x, y, '*'),
      '-'
);
```

- Dễ đọc
- Dễ hiểu
- **Dễ dàng bảo trì**

Ví dụ - Viết câu lệnh

Dùng ký tự space

Không cách

```
for(int i=0;i<40;i++)
    {system.out.println(i);}
```

Lập trình viên thường không để ý

Thay bằng

```
For (int i = 0; i < 40; i++ )
{
    system.out.println(i);
    }
```

- Dễ đọc, dễ hiểu
- **Dễ dàng sửa lỗi**

Chú thích

- Chú thích rất quan trọng: hỗ trợ đáng kể tính dễ đọc và dễ bảo trì của mã nguồn
- Cách viết chú thích
  - Mục đích sử dụng của các biến
  - Các câu lệnh phức tạp: i.e., gọi đến hàm khác
  - Chú thích các mô đun
    - Mục đích, chức năng của mô đun
    - Tham số, giá trị trả lại
    - Cấu trúc, thuật toán
    - Ý nghĩa của các biến cục bộ
    - Người viết, thời gian sửa đổi mô đun

Chú thích – Lưu ý

- Không cần chú thích cho những câu lệnh đã "rõ ràng"

```
// get the country code
$country_code = get_country_code($_SERVER['REMOTE_ADDR']);

// if country code is US
if ($country_code == 'US') {

   // display the form input for state
   echo form_input_state();
}
```

```
// display state selection for US users
$country_code = get_country_code($_SERVER['REMOTE_ADDR']);
if ($country_code == 'US') {
   echo form_input_state();
}
```

Cách viết chú thích

- Viết chú thích cho File

Mô tả ở đầu File

descriptive file header

separator ->

Exceptions.php

```
1  <?php  if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2  /**
3   * CodeIgniter
4   *
5   * An open source application development framework for PHP 4.3.2 or newer
6   *
7   * @package     CodeIgniter
8   * @author      ExpressionEngine Dev Team
9   * @copyright   Copyright (c) 2008 - 2009, EllisLab, Inc.
10  * @license     http://codeigniter.com/user_guide/license.html
11  * @link        http://codeigniter.com
12  * @since       Version 1.0
13  * @filesource
14  */
15
16 // ------------------------------------------------------------------------
17
```

(Source from https://code.tutsplus.com/tutorials/top-15-best-practices-for-writing-super-readable-code--net-8118)

Cách viết chú thích

- Viết chú thích cho Lớp

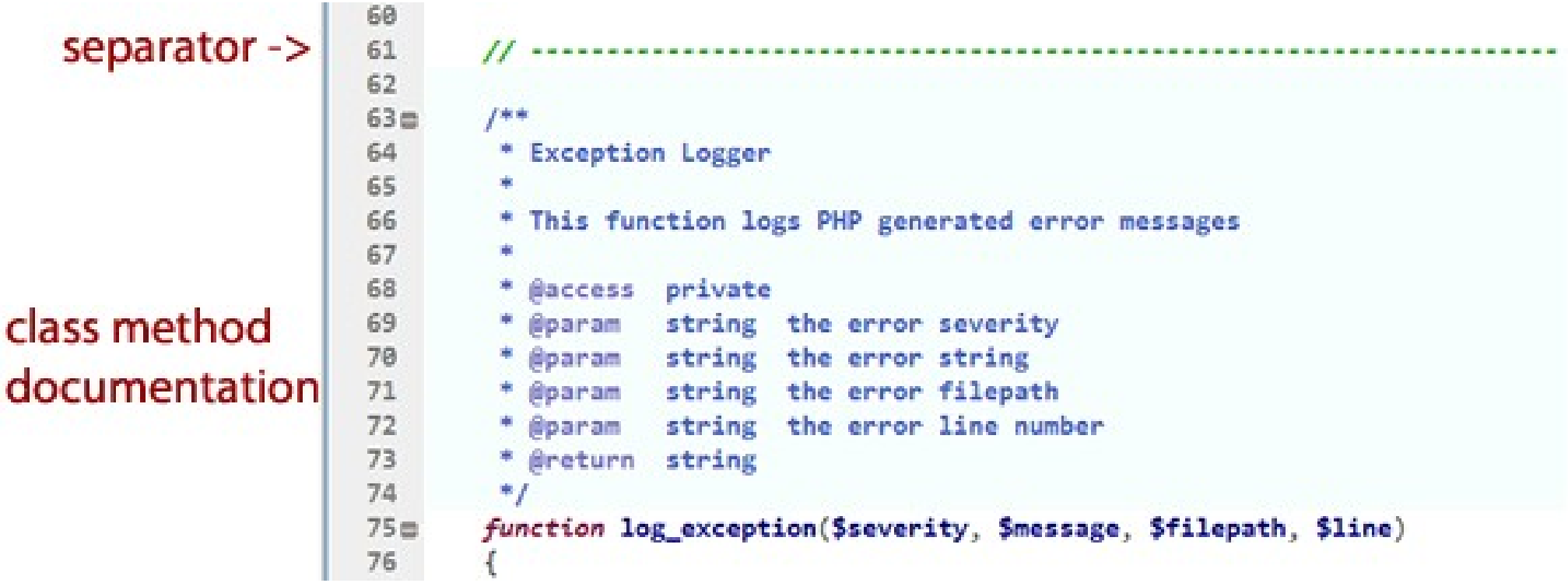
class documentation

```
/**
 * Exceptions Class
 *
 * @package     CodeIgniter
 * @subpackage  Libraries
 * @category    Exceptions
 * @author      ExpressionEngine Dev Team
 * @link        http://codeigniter.com/user_guide/libraries/exceptions.html
 */
class CI_Exceptions {
    var $action;
    var $severity;
    var $message;
    var $filename;
    var $line;
    var $ob_level;
```

(Source from https://code.tutsplus.com/tutorials/top-15-best-practices-for-writing-super-readable-code--net-8118)

Cách viết chú thích

- Viết chú thích cho Hàm



(Source from https://code.tutsplus.com/tutorials/top-15-best-practices-for-writing-super-readable-code--net-8118)

Tái sử dụng mã nguồn

- Tái sử dụng các thành phần phần mềm (components) định nghĩa trước
  - Tránh trường hợp “Phát minh lại bánh xe”
  - Sử dụng những đoạn mã đã được thẩm định chất lượng

Viết mã nguồn để tái sử dụng

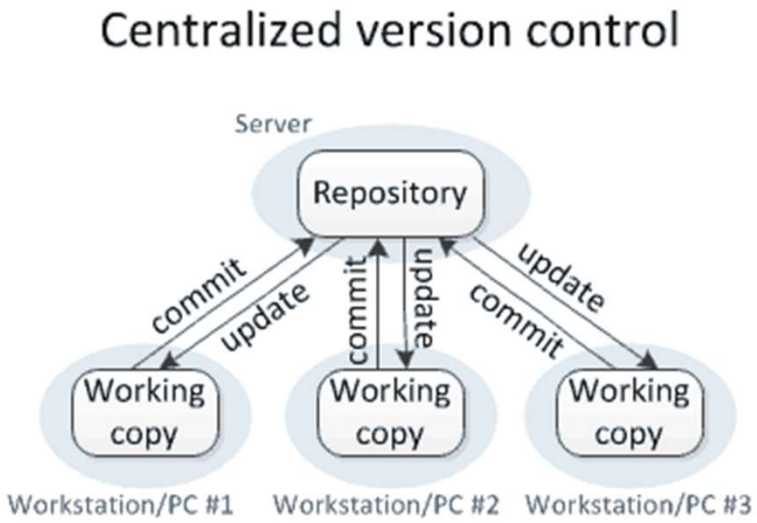
- Đặc điểm của phần mềm để tái sử dụng
  - Phân chia mô-đun
  - Có tính đóng gói
- Viết mã nguồn để tái sử dụng:
  - Tạo thư viện phần mềm
  - Lập trình tổng quát (generic programming)
  - Phần mềm sinh mã tự động (generators)
  - Kế thừa, đa hình, giao diện thống nhất,…

Quản lý các phiên bản mã nguồn

- Quản lý quá trình chỉnh sửa mã nguồn của một nhóm phát triển phần mềm
- Các khái niệm cơ bản
  - Repository (kho lưu trữ)
    - Lưu trữ dự án chung của đội phát triển phần mềm
  - Working copies
    - Check out: Lấy dự án từ server về máy cục bộ
    - Commit: Đưa các thay đổi lên server
    - Update: Cập nhật thay đổi từ thành viên khác về máy cục bộ
    - Merge: Nhiều thành viên cập nhật trên một tệp tin
    - V..v.

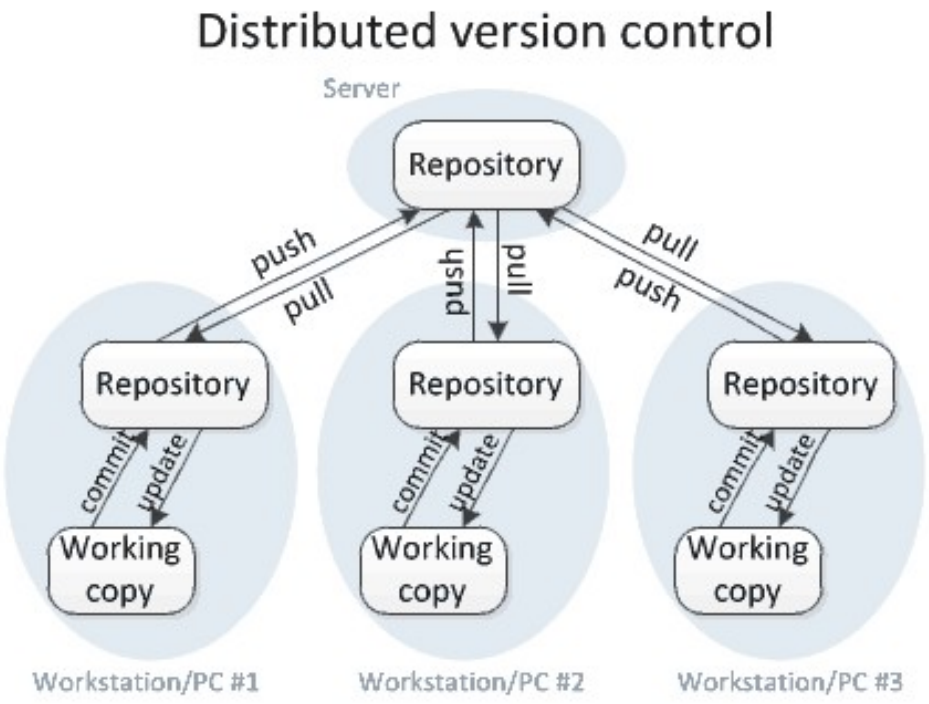
Các loại công cụ quản lý phiên bản

- Tập trung:
  - Mỗi người dùng lấy bản sao làm việc của riêng mình, nhưng chỉ có một kho lưu trữ trung tâm.
  - CVS (Concurrent Versions System) và SVN (SubVersioN)



Các loại công cụ quản lý phiên bản

- Phân tán
  - Mỗi người dùng có kho lưu trữ của riêng mình và bản sao làm việc.
  - Git, Mercurial



Các loại công cụ quản lý phiên bản

- Quản lý tập trung
  - Tránh conflic hoặc out of update
- Quản lý phân tán
  - An toàn hơn so với quản lý tập trung

Gỡ lỗi

- Debug (gỡ lỗi) là một kĩ năng nền tảng của lập trình viên
  - Phân tích lỗi của chương trình xuất phát từ nguyên do gì.
  - Loại bỏ lỗi (errors) khỏi chương trình
  - Hiểu rõ hơn sự thực thi của chương trình

Các phương pháp gỡ lỗi

- Debugging Tool
  - Dùng công cụ để debug
  - Microsoft Visual Studio Debugger , GNU Debugger
- Printlining
  - Thêm dòng lệnh để in ra những thông tin cần theo dõi
- Logging
  - Ghi lại những thông tin sau khi chương trình thực thi

Tổng kết

- Quy trình xây dựng mã nguồn
- Phong cách lập trình
- Tái sử dụng mã nguồn
- Quản lý các phiên bản mã nguồn
- Các phương pháp gỡ lỗi chương trình