Lập trình hướng đối tượng

TS H.Q.Thắng, TS C.T.Dũng
Bộ môn công nghệ phần mềm
Đại học bách khoa Hà Nội

Môn học: Lập trình hướng đối tượng

- Giới thiệu về môn học
- Đề cương chi tiết môn học
- Tài liệu tham khảo môn học
- Bài tập lớn môn học
- Thi và đánh giá môn học

Giới thiệu về môn học

- **Mục đích môn học:** Môn học này được giới thiệu cho sinh viên các kiến thức và kỹ thuật trong LTHDT. Sinh viên sẽ tiếp cận với phương pháp luận, các kỹ năng, kỹ thuật trong thiết kế và lập trình hướng đối tượng
- **Vị trí môn học:** áp dụng cho sinh viên năm thứ 4 (học kỳ 7), CNTT. Trong môn học có sử dụng kiến thức của các môn học: Cấu trúc dữ liệu và giải thuật, hệ điều hành, lý thuyết ngôn ngữ

Giới thiệu về môn học (tiếp)

- Thời lượng môn học: 45 tiết lý thuyết được phân bổ trong 12 tuần. Sẽ có các bài tập tuần, sinh viên tự tham khảo tài liệu được giao để thực hiện bài tập lớn (đồ án môn học) ngay từ tuần 2 của cuối học kỳ
- Bài tập lớn môn (đồ án môn học): làm theo nhóm, có bảo vệ làm theo nhóm

Đề cương chi tiết môn học

- **Chương 1.** Tổng quan về lập trình hướng đối tượng
- **Chương 2.** Các kỹ thuật làm việc với hằng, biến, xây dựng và sử dụng hàm trong LTHĐT
- **Chương 3.** Các kỹ thuật cơ bản trong xây dựng lớp
- **Chương 4.** Kỹ thuật thừa kế, kết tập và đa hình trong LTHĐT

Đề cương chi tiết môn học

- **Chương 5.** Ngôn ngữ mô hình hóa UML và sử dụng trong LTHDT
- **Chương 6.** Thiết kế khuôn mẫu
- **Chương 7.** Các phép đo đánh giá phần mềm hướng đối tượng

Tài liệu tham khảo môn học

- **[1].** Peter Coad, Jill Nicola; Object-Oriented Programming
- **[2].** Harvey M. Deitel, Paul J. Deitel C++ How to Programing (5th Edition)
- **[3].** Harvey M. Deitel, Paul J. Deitel How to Java Programing (5th Edition)
- **[4].** Bruce Eckel - Thinking in C++ . Second Edition. MindView Inc., 2000. Có thể tải về từ web site: www.bruceeckel.com
- [5] **Martin Fowler UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition**

Bài tập lớn (đồ án môn học)

- Đồ án môn học: Sinh viên được chia làm các nhóm, mỗi nhóm chọn một cấu trúc dữ liệu hoặc giải thuật tiêu biểu để xây dựng phần mềm theo nguyên lý hướng đối tượng (công cụ tuỳ chọn).
- Yêu cầu của đồ án môn học: các nhóm tiến hành mô tả và đánh giá phần mềm hướng đối tượng theo các tiêu chí

Đánh giá môn học

- Cuối kỳ thi hết môn, thời gian 60-90 phút
- Điểm đánh giá môn học:
  - Điểm kiểm tra điều kiện: 20%
  - Điểm thi: trọng số 40%
  - Điểm đồ án môn học: trọng số 40%
- Điều kiện miễn thi: thực hiện tốt đồ án môn học và có những nghiên cứu sâu về các lĩnh vực trong CNTT. Đồ án môn học thực hiện đúng theo các quy định chuẩn của môn học

Thảo luận

- Các câu hỏi của sinh viên liên quan đến nội dung môn học "Lập trình hướng đối tượng"
- Các câu hỏi của sinh viên liên quan đến nội dung của bài tập lớn và phương pháp thực hiện và bảo vệ
- Các câu hỏi của sinh viên liên quan đến nội dung thi

Chương 1. Các khái niệm cơ bản trong Lập trình hướng đối tượng (LTHDT)

1. Lịch sử phát triển của các ngôn ngữ lập trình
2. Đối tượng và các khái niệm liên quan
3. Khái niệm thực hiện ẩn (hidden implementation) trong LTHDT
4. Khái niệm tái sử dụng trong LTHDT
5. Khái niệm kế thừa trong LTHDT
6. Khái niệm đa hình trong lập trình hướng đối tượng

Chương 1. Các khái niệm cơ bản trong Lập trình hướng đối tượng (LTHDT) tiếp

7. Khởi tạo và giải phóng đối tượng trong LTHDT
8. Bắt lỗi và xử lý lỗi trong LTHDT
9. Phân tích và thiết kế hướng đối tượng
10. Extreme programming
11. Tại sao ngôn ngữ C++ thông dụng
12. Tại sao ngôn ngữ Java thông dụng
13. Quá trình dịch một phần mềm
14. Các đặc điểm của công cụ biên dich độc lập
15. Câu hỏi và bài tập tuần 1

1. Lịch sử phát triển của các ngôn ngữ lập trình

- Tất cả các ngôn ngữ lập trình cho phép và yêu cầu chúng ta phải trừu tượng hóa (abstraction).
- Trong tất cả các ngôn ngữ lập trình chúng ta đã trực tiếp hoặc gián tiếp thực hiện trừu tượng hóa.
  - Hợp ngữ (Assembly language) là một ngôn ngữ lập trình tuần tự, gần với tập các lệnh mã máy của CPU vì thế khả năng trừu tượng hóa là rất nhỏ.
  - Các ngôn ngữ lập trình cấu trúc khả năng trừu tượng đã tăng lên rất nhiều so với hợp ngữ. Tư tưởng chính: tìm ra cách thể hiện bài toán cần giải quyết bằng những cấu trúc lập trình có trong các ngôn ngữ tương ứng.
  - Phương pháp tiếp cận hướng logic (logic-oriented)

1. Lịch sử phát triển của các ngôn ngữ lập trình

- Mong muốn tìm ra môt cách tiếp cận cho phép giải quyết chung được số lượng lớn các bài toán.
- Phương pháp tiếp cận hướng đối tượng được coi là một phương pháp tốt để phục vụ cho mục đích ấy.
- Chúng ta tiếp cận bài toán bằng cách thể hiện các thành phần của bài toán là các “đối tượng” (object).
- Mỗi đối tượng có thể coi như một “thành phần sống” - có nghĩa là nó có trạng thái, có các hoạt động - thực hiện các thao tác nào đó. Các thao tác này thực hiện các chức năng của hệ thống.

Trừu tượng hóa

- Trừu tượng hóa điều khiển (control abstraction): Một trong những đặc tính quan trọng của các ngôn ngữ lập trình.
  - a = (1 + 2) * 5;
  - Bao gồm trong đó khái niệm, trừu tượng hóa chức năng
- Trừu tượng hóa dữ liệu: Chỉ quan tâm dữ liệu được sử dụng như thế nào, không quan tâm nó được biểu diễn cụ thể ra sao.

1. **Lịch sử phát triển của các ngôn ngữ lập trình**

Alan Kay đã tổng hợp các đặc tính của LTHDT:

1. Tất cả đều là đối tượng.
2. Chương trình phần mềm có thể coi là một tập hợp các đối tượng tương tác với nhau
3. Mỗi đối tượng trong chương trình có các dữ liệu độc lập của mình và chiếm bộ nhớ riêng của mình.
4. Mỗi đối tượng đều có dạng đặc trưng của lớp các đối tượng đó.
5. Tất cả các đối tượng thuộc về cùng một lớp đều có các hành vi giống nhau

2. Đối tượng và các khái niệm liên quan

- Đối tượng
- Giao diện của đối tượng
- Lớp đối tượng
- Thuộc tính đối tượng
- Hành vi đối tượng
- Gửi thông điệp
- Biểu điễn đối tượng - biểu đồ đối tượng và biểu đồ lớp trong UML

Đối tượng (object)

- **Đối tượng** là chìa khóa để hiểu được kỹ thuật hướng đối tượng
- Trong hệ thống hướng đối tượng, mọi thứ đều là đối tượng

Viết một chương trình hướng đối tượng nghĩa là đang xây dựng một mô hình của một vài bộ phận trong thế giới thực

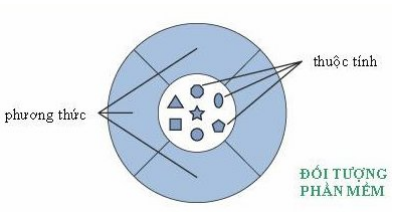
Đối Tượng Thế Giới Thực (Real Object)

- Một **đối tượng thế giới thực** là một thực thể cụ thể mà thông thường chúng ta có thể *sờ, nhìn thấy* hay *cảm nhận* được.
- Tất cả có trạng thái (state) và hành động (behaviour)

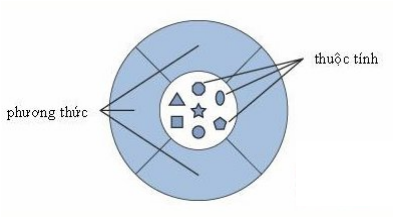
	Trạng thái	Hành động	
Con chó	Tên Màu Giống Vui sướng	Sủa Vẫy tai Chạy Ăn	
Xe đạp	Bánh răng Bàn đạp Dây xích Bánh xe	Tăng tốc Giảm tốc Chuyển bánh răng ...	

Đối Tượng Phần Mềm (Software Object)

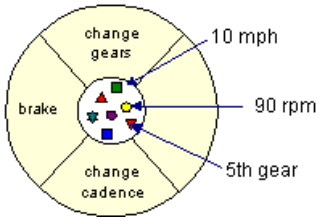
- Các **đối tượng phần mềm** có thể được dùng để *biểu diễn* các đối tượng thế giới thực.
- Cũng có trạng thái và hành động
  - Trạng thái: thuộc tính (attribute; property)
  - Hành động: phương thức (method)



Đối tượng



Đối tượng phần mềm



Đối tượng phần mềm **Xe Đạp**

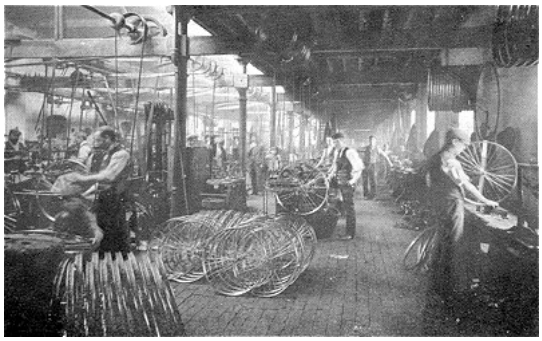
Đối tượng (object) là một thực thể phần mềm bao bọc các **thuộc tính** và các **phương thức** liên quan.

Thuộc tính được xác định bởi giá trị cụ thể gọi là **thuộc tính thể hiện**.

Một đối tượng cụ thể được gọi là một **thể hiện**.

Lớp đối tượng

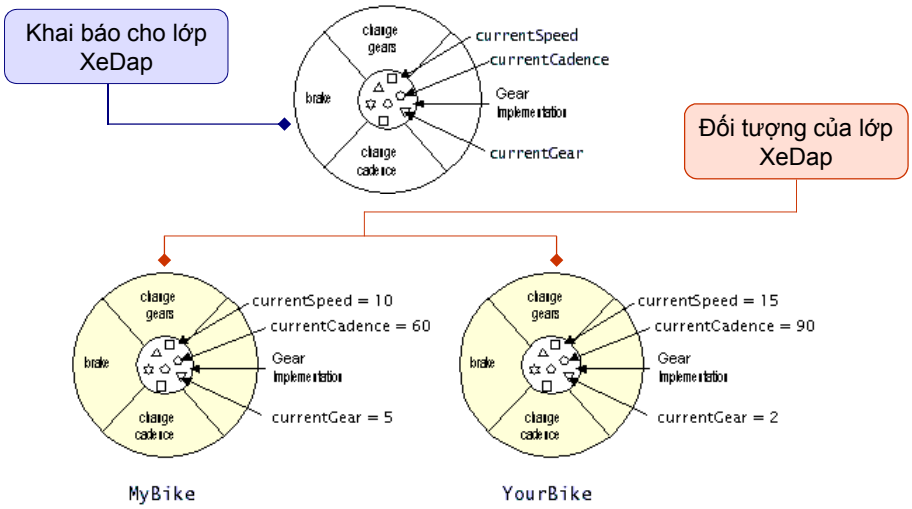
- Trong thế giới thực có nhiều đối tượng cùng loại.
- Chương trình hướng đối tượng có nhiều đối tượng cùng loại chia sẻ những đặc điểm chung.
- Ví dụ



Lớp

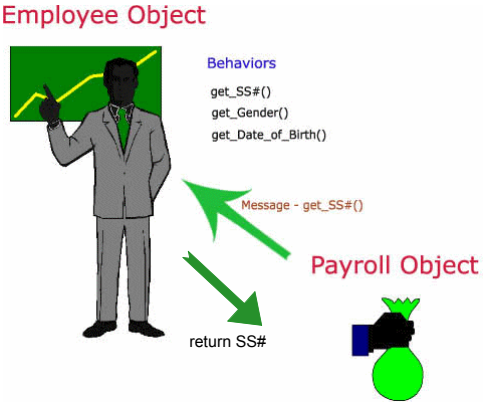
- Một **lớp** là một thiết kế (blueprint) hay mẫu (prototype) cho các đối tượng cùng kiểu
  - Ví dụ: lớp XeDap là một thiết kế chung cho nhiều đối tượng xe đạp được tạo ra
- Lớp định nghĩa các thuộc tính và các phương thức chung cho tất cả các đối tượng của cùng một loại nào đó
- Một đối tượng là một thể hiện cụ thể của một lớp.
  - Ví dụ: mỗi đối tượng xe đạp là một thể hiện của lớp XeDap
- Mỗi thể hiện có thể có những thuộc tính thể hiện khác nhau
  - Ví dụ: một xe đạp có thể đang ở bánh răng thứ 5 trong khi một xe khác có thể là đang ở bánh răng thứ 3.

Ví dụ Lớp Xe đạp



Trao đổi thông điệp

- Một chương trình (xây dựng theo tiếp cận HĐT) là tập các đối tượng trao đổi thông điệp với nhau

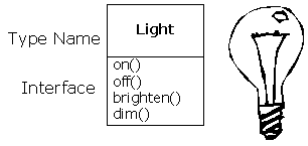


Giao diện của đối tượng

- Thử thách của LTHDT là có thể ánh xạ một phần tử (thực thể) trong không gian bài toán về một đối tượng trong không gian lời giải.
- Một đối tượng có thể được sử dụng khi nó có thể đáp ứng được một số "yêu cầu" nào đó từ bên ngoài. Giao diện của đối tượng định nghĩa các dịch vụ mà đối tượng cung cấp

C++

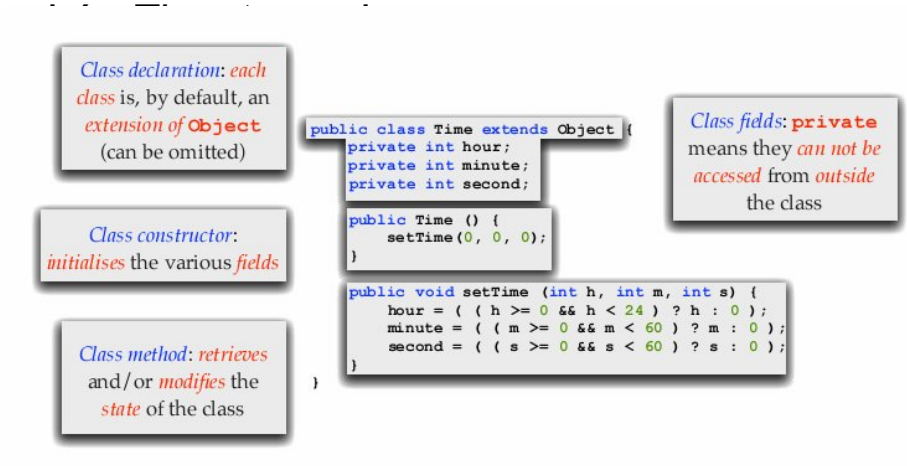
```
Light
lt;
lt.on();
```



Java

```
Light lt = new Light();
lt.on();
```

Ví dụ về lớp và đối tượng trong một số NNLT



Java: Chương trình và các đối tượng

```
public class Test {

    public static void main (String args[]) {
        Time time = new Time();

        time.hour = 7;
        time.minute = 15;
        time.second = 30;
    }
}
```

```
Test.java:6: hour has private access in Time
      time.hour = 7;
           ^
Time.java:7: minute has private access in Time
      time.minute = 15;
           ^
Time.java:8: second has private access in Time
      time.second = 30;
           ^
3 errors
```

Lớp Time trong C++

Class definition bắt đầu bằng từ khoá **class**.

Class body bắt đầu bằng ngoặc mở.

Function prototype cho các **public** member function.

Class **Time** definition (1 of 1)

Constructor: thành viên trùng tên với tên class, **Time**, và không có giá trị trả về.

Nhãn quyền truy nhập

private data member chỉ có thể được truy nhập từ các member function.

```
1  class Time {
2
3  public:
4     Time();                        // constructor
5     void setTime( int, int, int ); // set hour, minute, second
6     void printUniversal();         // print universal-time format
7     void printStandard();          // print standard-time format
8
9  private:
10    int hour;      // 0 - 23 (24-hour clock format)
11    int minute;    // 0 - 59
12    int second;    // 0 - 59
13
14 }; // end class Time
```

Chương trình và các đối tượng: C++

fig06_07.cpp (1 of 2)

Include **time1.h** để đảm bảo tạo đúng và để tính kích thước đối tượng thuộc lớp **Time**.

```
1  // Fig. 6.7: fig06_07.cpp
2  // Program to test class Time.
3  // NOTE: This file must be compiled with time1.cpp.
4  #include <iostream>
5
6  using std::cout;
7  using std::endl;
8
9  // include definition of class Time from time1.h
10 #include "time1.h"
11
12 int main()
13 {
14    Time t;  // instantiate object t of class Time
15
16    // output Time object t's initial values
17    cout << "The initial universal time is ";
18    t.printUniversal();   // 00:00:00
19    cout << "\nThe initial standard time is ";
20    t.printStandard();    // 12:00:00 AM
21
22    t.setTime( 13, 27, 6 );   // change time
23
```

Chương trình và các đối tượng: C++

```
24    // output Time object t's new values
25    cout << "\n\nUniversal time after setTime is ";
26    t.printUniversal();   // 13:27:06
27    cout << "\nStandard time after setTime is ";
28    t.printStandard();    // 1:27:06 PM
29
30    t.setTime( 99, 99, 99 );  // attempt invalid settings
31
32    // output t's values after specifying invalid values
33    cout << "\n\nAfter attempting invalid settings:"
34         << "\nUniversal time: ";
35    t.printUniversal();   // 00:00:00
36    cout << "\nStandard time: ";
37    t.printStandard();    // 12:00:00 AM
38    cout << endl;
39
40    return 0;
41
42 } // end main
```

fig06_07.cpp
(2 of 2)

fig06_07.cpp
output (1 of 1)

```
The initial universal time is 00:00:00
The initial standard time is 12:00:00 AM

Universal time after setTime is 13:27:06
Standard time after setTime is 1:27:06 PM
```

Đóng gói (Encapsulation)

- Kết quả của quá trình trừu tượng hóa:
  - Đối tượng = Thuộc tính + phương thức
- Đóng gói: Thông tin và hoạt động bên trong của một đối tượng có thể được che khỏi thế giới bên ngoài.
  - Che dấu thông tin
  - Thực hiện ẩn

A Television Set:-
A simple interface (Abstraction)
Hidden complexity (Encapsulation)

3. Khái niệm thực hiện ẩn trong LTHDT

- Thông thường trong lập trình người ta phân biệt hai công việc: thứ nhất là công việc tạo ra các lớp đối tượng (class creators) và công việc thứ hai là sử dụng các lớp đối tượng này (client programmers).
- Có hai nguyên nhân chỉ ra sự cần thiết phải có khả năng thực hiện ẩn trong LTHDT:
  - Khả năng này cho phép quy định những người sử dụng (client) chỉ được phép truy nhập và sử dụng những gì đã quy định cho họ. Một phần class được che dấu và không cho người sử dụng được quyền truy nhập.
  - Khả năng này cho phép những người thiết kế các class có khả năng thay đổi hay định nghĩa lại class mà vẫn chắc chắn rằng không ảnh hưởng tới chương trình của những người sử dụng class này.

3. Khái niệm thực hiện ẩn trong LTHDT

- C++/ Java sử dụng các từ khóa để xác định khả năng truy nhập các thông tin dữ liệu từ bên ngoài lớp: public, private, và protected.
- Public
- private
- protected

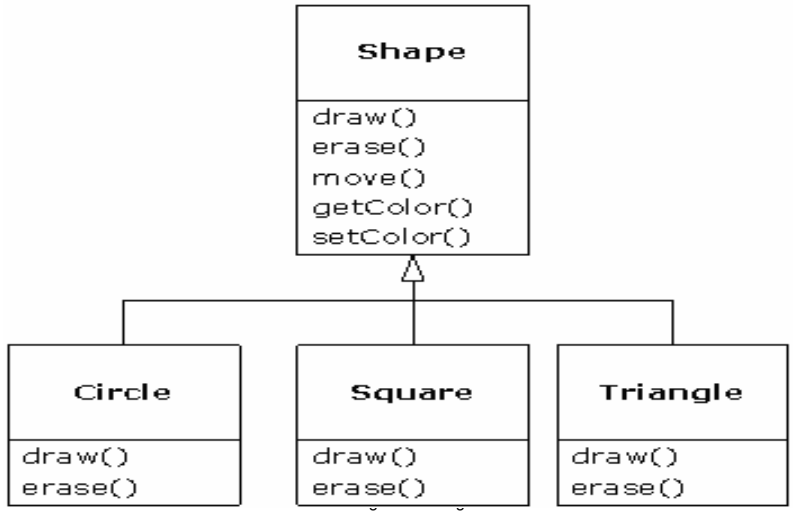
4. Khái niệm tái sử dụng trong LTHDT

- Khi một lớp được tạo ra, chúng ta đã mất công sức để xây dựng và viết mã cho các lớp này. Khả năng sử dụng lại các lớp là một trong những tính chất quan trọng của LTHDT.
- Cách dễ nhất để sử dụng lại lớp là sử dụng trực tiếp các đối tượng của lớp này như là các biến cần thiết để giải quyết bài toán
- Cách thứ hai là sử dụng các biến - đối tượng thuộc một lớp để xây dựng nên các lớp mới (tức là đối tượng nằm bên trong lớp mới. Nguyên lý hợp thành (*composition)* còn goi là nguyên lý kết tập (*aggregation*). Nguyên lý hợp thành thể hiện quan hệ có *(has-a relationship)*.
- Cách thứ ba: thể hiện ở nguyên lý kế thừa

5. Khái niệm kế thừa trong LTHDT

- Khả năng cho phép chúng ta có thể xây dựng được các lớp mới dựa trên các lớp đã có sẵn, chúng ta chỉ cần thêm vào đó những gì chưa có, những gì chưa đủ, được gọi là tính kế thừa trong LTHDT.
- Lớp cơ sở
- Lớp kế thừa
- Tính chất của kế thừa
- Mối quan hệ “Là”: các đối tượng thuộc lớp kế thừa có thể được coi như các đối tượng thuộc các lớp cơ sở và quan hệ “Như là”: trong các lớp kế thừa định nghĩa lại các hành vi của lớp cơ sở
- Ký pháp UML

5. Khái niệm kế thừa trong LTHDT

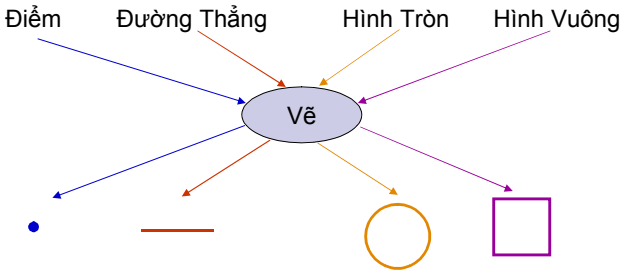


6. Khái niệm đa hình trong LTHDT

- "polymorphism" có nghia "nhiều hình thức", hay "nhiều dạng sống"
- Định nghĩa: Đa hình là hiện tượng các đối tượng thuộc các lớp *khác nhau* có khả năng hiểu *cùng một* thông điệp theo các cách *khác nhau*
- Ví dụ: nhận được cùng một thông điệp "nhảy", một con kangaroo và một con cóc nhảy theo hai kiểu khác nhau: chúng cùng có hành vi "nhảy" nhưng các hành vi này có nội dung khác nhau.

Đa hình

- Ngữ cảnh khác → kết quả khác



6. Khái niệm đa hình trong LTHDT

- Khi xây dựng các lớp kế thừa trong LTHDT có thể xảy ra trường hợp trong lớp kế thừa và lớp cơ sở cùng định nghĩa một hành vi (hàm) nào đó có giao diện giống hệt nhau. Khi đó chúng ta gọi hàm này, chương trình dịch không thể xác định được sẽ gọi hành vi nào, hành vi được định nghĩa trong lớp cơ sở hay hành vi trong lớp kế thừa.
  - Cơ chế liên kết sớm (early binding function)
  - Cơ chế liên kết muộn (late binding function)

6. Khái niệm đa hình trong LTHDT

- Các thể hiện khác của tính đa hình trong LTHDT có thể được thực hiện ở các nguyên tắc sau:
  - Nguyên lý chồng hàm trong LTHDT: LTV có khả năng xây dựng các hàm có tên giống hệt nhau nhưng có các đối số khác nhau: khác nhau về số lượng các đối số có trong hàm, khác nhau về kiểu dữ liệu của các đối số.

 C++  Java

  - Nguyên lý chồng toán tử trong LTHDT: LTV có khả năng định nghĩa những toán tử đã tồn tại trong các ngôn ngữ lập trình tương ứng trên các dữ liệu mới - các đối tượng thuộc các lớp mà người lập trình muốn xây dựng.

 C++  Java

7. Khởi tạo và giải phóng đối tượng trong LTHDT

Tồn tại hai phương pháp chia sẻ tài nguyên cho các đối tượng và các biến nói chung trong kỹ thuật lập trình:

- Phương pháp thứ nhất: Để tạo điều kiện cho chương trình có thể thực hiện với tốc độ cao nhất, tất cả các biến và các đối tượng đã được chia sẻ trước bộ nhớ, người ta gọi phương pháp này là nguyên lý lưu trữ tĩnh các biến và đối tượng (static storage). Các biến và các đối tượng luôn có sẵn và tồn tại trong suốt thời gian thực hiện chương trình.
- Phương pháp thứ hai: Sử dụng nguyên lý vun đống (heap) của bộ nhớ. Toàn bộ công việc này: tạo ra đối tượng, sử dụng chúng, giải phóng chúng được thực hiện trong lúc thực hiện chương trình.

8. Bắt lỗi và xử lý lỗi trong LTHDT

- Trong kỹ thuật lập trình, bắt lỗi và xử lý lỗi là một trong những công việc nặng nhọc và vất vả nhất.
- *Bắt lỗi (Exception handling)* hay dịch sát nghĩa theo từ tiếng Anh "xử lý ngoại lệ" có thể thực hiện trực tiếp ngay trong môi trường, hoặc sử dụng các khả năng của hệ điều hành. Khi có lỗi xuất hiện trong chương trình, lỗi này cần phải được xử lý: nhận dạng lỗi, phân loại lỗi và xử lý lỗi.
- Các đặc điểm của công việc xử lý lỗi:
  - Phần việc này phải được tách riêng và sẽ không được thực hiện nếu chương trình thực hiện bình thường, không có lỗi
  - Thông thường phần công việc xử lý lỗi là một phần việc song song với phần việc chính của chương trình.

9. Phân tích và thiết kế hướng đối tượng

- Phân tích và thiết kế hướng đối tượng là một trong các phương pháp hiệu quả nhất để phát triển phần mềm.
- Nếu như quan trong nhất trong LTHDT là lớp và đối tượng thì từ đó chúng ta cũng cần có một phướng pháp hướng đối tượng cho phép chúng ta phát triển các phần mềm tương ứng.
- Phương pháp luận (*methodology*) trong PT&TK phần mềm thông thường được định nghĩa như là một tập các quá trình và thao tác để tìm và khám phá cách có thể giải quyết được bài toán phần mềm.

9. Phân tích và thiết kế hướng đối tượng

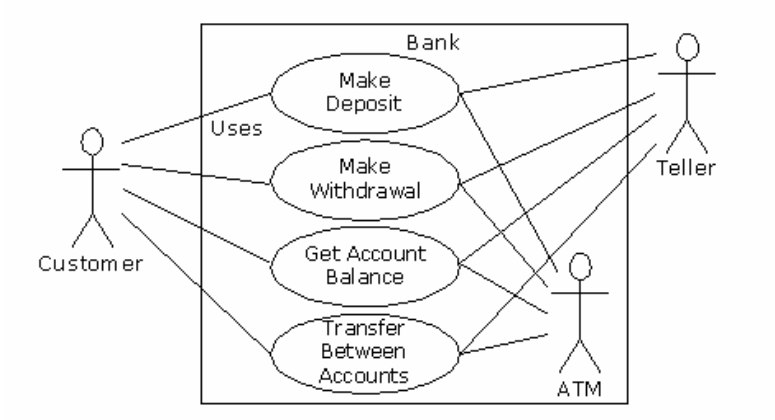
- Thông thường quá trình phân tích và thiết kế một phần mềm theo hướng đối tượng được chia làm 6 giai đoạn, trong đó giai đoạn số 0 được coi như là giai đoạn chuẩn bị.
  - Giai đoạn 0: Lập kế hoạch (make a plan)
  - Giai đoạn 1: Xác định mục tiêu - làm gì (what are we making)
  - Giai đoạn 2: Xác định cách làm thế nào (how to build it)
  - Giai đoạn 3: Xây dựng phần lõi - Building the core
  - Giai đoạn 4: Lặp lại các trường hợp sử dụng
  - Giai đoạn 5: Phát triển (evolution)

9. Phân tích và thiết kế hướng đối tượng

Giai đoạn 1: Xác định mục tiêu - làm gì (what are we making)

- Trong giai đoạn này chúng ta có nhiệm vụ xác định cụ thể các mục tiêu, chức năng và nhiệm vụ mà phần mềm chúng ta cần xây dựng phải đáp ứng.
- Trong phương pháp lập trình cổ điển hướng thủ tục người ta gọi giai đoạn này là giai đoạn tạo ra "phân tích yêu cầu và mô tả hệ thống" (*requirements analysis* and *system specification)*.
- Trong PT&TK hướng đối tượng người ta sử dụng các ký pháp và kỹ thuật User case để mô tả các công việc này.

Biểu đồ Use case



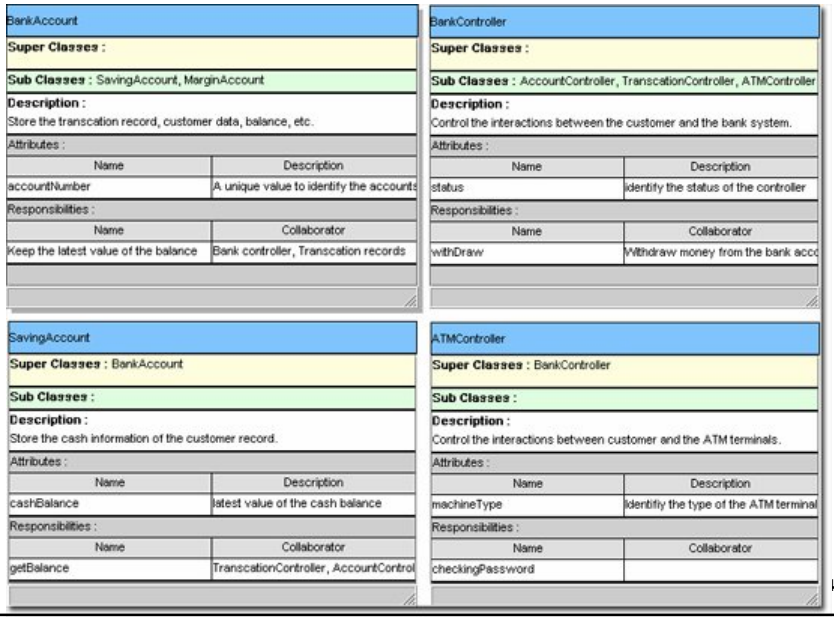
9. Phân tích và thiết kế hướng đối tượng

Giai đoạn 2: Xác định cách làm - như thế nào

Trong giai đoạn này chúng ta cần phải xác định các đặc tính của các lớp. Một trong các kỹ thuật được ứng dụng nhiều nhất đó là Thẻ tương tác - tính chất-lớp *Class-Responsibility-Collaboration* (CRC) card. Mỗi thẻ thể hiện một lớp, trên thẻ chúng ta lưu lại các thông tin sau về các lớp:

1. Tên của lớp. Thông thường người ta đặt tên lớp liên quan đến vai trò của lớp, chúng ta sẽ sử dụng lớp để làm gì.
2. Trách nhiệm của lớp: lớp có thể làm gì. Thông thường các thông tin ở đây bao gồm tên của các hàm thành phần
3. Tương tác của lớp: lớp này có thể tương tác được với những lớp nào khác.

CRC card



BankAccount	
Super Classes :	
Sub Classes : SavingAccount, MarginAccount	
Description : Store the transcation record, customer data, balance, etc.	
Attributes :	
Name	Description
accountNumber	A unique value to identify the accounts
Responsibilities :	
Name	Collaborator
Keep the latest value of the balance	Bank controller, Transcation records

BankController	
Super Classes :	
Sub Classes : AccountController, TranscationController, ATMController	
Description : Control the interactions between the customer and the bank system.	
Attributes :	
Name	Description
status	identify the status of the controller
Responsibilities :	
Name	Collaborator
withDraw	Withdraw money from the bank acc

SavingAccount	
Super Classes : BankAccount	
Sub Classes :	
Description : Store the cash information of the customer record.	
Attributes :	
Name	Description
cashBalance	latest value of the cash balance
Responsibilities :	
Name	Collaborator
getBalance	TranscationController, AccountControl

ATMController	
Super Classes : BankController	
Sub Classes :	
Description : Control the interactions between customer and the ATM terminals.	
Attributes :	
Name	Description
machineType	Identify the type of the ATM terminal
Responsibilities :	
Name	Collaborator
checkingPassword	

9. Phân tích và thiết kế hướng đối tượng

- Trong giai đoạn này , một trong những yêu cầu quan trong đó là thiết kế các đối tượng (object design). Trong PT&TK hướng đối tượng người ta đã tổng kết 5 bước để thiết kế đối tượng:

  Bước 1. Phát hiện đối tượng (Object discovery). Bước này được thực hiện ở giai đoạn phân tích chương trình. Chúng ta phát hiện các đối tượng nhờ các yếu tố bên ngoài và các đặc điểm bên ngoài

  Bước 2. Lắp ráp đối tượng (Object assembly). Bước tìm kiếm các đặc điểm của đối tượng để thêm vào các thuộc tính, các hàm thành phần cho đối tượng.

9. Phân tích và thiết kế hướng đối tượng

Bước 3. Xây dựng hệ thống (System construction). Trong giai đoạn này chúng ta phát triển các đối tượng, xem xét các tương tác giữa các đối tượng để hình thành hệ thống hoạt động.

Bước 4. Mở rộng hệ thống (System extension). Khi chúng ta thêm vào các tính năng của hệ thống, chúng ta cần thêm các lớp mới, các đối tượng mới và các tương tác giữa các đối tượng này với các đối tượng đã có trong hệ thống.

Bước 5. Tái sử dụng đối tượng (Object reuse). Đây là một trong những thử nghiệm quan trọng của các đối tượng và lớp trong thiết kế phần mềm. Chúng ta cần phải sử dụng lại các lớp và các đối tượng trong phần mềm (thông qua tính kế thừa và tương tác giữa các đối tượng)

9. Phân tích và thiết kế hướng đối tượng

Một số đặc điểm cần lưu ý khi phát triển các lớp

1. Chúng ta cần phân biệt rõ việc tạo ra lớp, sau đó mới nghĩa tới việc phát triển và hoàn thiện lớp trong quá trình giải quyết bài toán
2. Việc phát hiện ra các lớp cần thiết cho chương trình là một trong những nhiệm vụ chính của thiết kế hệ thống, nếu chúng ta đã có những lớp này (trong một thư viện lớp nào đó chẳng hạn) thì công việc sẽ dễ dàng hơn
3. Khi phân tích hay phát triển các lớp không nên tập trung vào để biết tất cả về một lớp, chúng ta sẽ biết tất cả các thuộc tính này khi phát triển hệ thống (learns as you go)

9. Phân tích và thiết kế hướng đối tượng

Một số đặc điểm cần lưu ý khi phát triển các lớp (tiếp)

4. Khi tiến hành lập trình cần tuân thủ theo các thiết kế đã làm. Không nên băn khoăn rằng chúng ta sẽ không sử dụng phương pháp lập trình truyền thống và cảm thấy choáng ngợp trước số lượng lớn các đối tượng.
5. Luôn luôn giữ nguyên tắc: mọi vấn đề cần giải quyết theo phương án đơn giản nhất, không nên phức tạp hóa. Sử dụng nguyên lý của Occam Razor: *Lớp đơn giản nhất bao giờ cũng là lớp tốt nhất, hãy bắt đầu bằng những cái đơn giản và chúng ta sẽ kết thúc bằng những hệ thống phức tạp*

10. Extreme programming

- Các tư tưởng của *Extreme Programming* (XP) rất thú vị và rất cơ bản. XP chứa đựng cả những quan điểm về phương pháp và kỹ thuật lập trình và có rất nhiều những chỉ dẫn ở trong đó. Một số những chỉ dẫn mang tính chất xuyên suốt và có thể trở thành các phương pháp lập trình. Trong XP hai tư tưởng quan trọng nhất là:
  - Viết kiểm thử trước (write tests first)
  - Lập trình theo cặp (pair programming)

10. Extreme programming

- Viết kiểm thử trước
  - Kiểm thử (testing) thông thường luôn được thực hiện ở giai đoạn cuối cùng trong quá trình phát triển phần mềm.
  - Trong XP quan điểm này được thay đổi hoàn toàn. Kiểm thử được coi trọng là một công việc hàng đầu, có trọng số quan trong ít nhất là bằng với lập trình và trên thực tế trong XP quan niệm rằng chúng ta phải viết kiểm thử trước khi chúng ta tiến hành lập trình. Theo quan điểm này chúng ta phải đảm bảo kiểm thử phải được thực hiện thành công tại mọi thời điểm trong khi tích hợp chương trình.

10. Extreme programming

Viết kiểm thử trước có hai hiệu ứng quan trọng:

- Thứ nhất: thúc đẩy việc xác định rõ ràng hơn giao diện của các lớp. Quan điểm này của XP đã chỉ ra khi viết các kiểm thử chúng ta đã chỉ ra rõ ràng hơn, đầy đủ hơn giao diện của các lớp (ở đấy có thể sử dụng các công cụ như các biểu đồ UML, các thẻ CRC, ...
- Thứ hai: Chúng ta có khả năng thực hiện kiểm thử mỗi khi chúng ta compile phần mềm. Lịch sử phát triển của các NNLT đã qua triển như thế: Assembly kiểm tra cú pháp (syntax), trong lập trình hướng thủ tục đã thực hiện kiểm tra các ngữ nghĩa (semantic) điều đó giúp chúng ta loại bỏ lỗi. Các ngôn ngữ LTHDT kiểm tra ngữ nghĩa của các phần mềm ở mức độ cao hơn nũa và tiến tới có thể định nghĩa các trường hợp kiểm thử để kiểm tra khi compile phần mềm.

10. Extreme programming

- Lập trình theo cặp Pair programming
  - Tư tưởng lập trình theo cặp phủ nhận các tư tưởng cá nhân chủ nghĩa (individualism).
  - Trong tư tưởng lập trình theo cặp này công việc viết mã được tổ chức theo hai người: một người thực hiện lập trình và người kia thì suy nghĩ về nó, cách tiến hành nó. Người suy nghĩ về chương trình có thể nghĩ được rộng hơn và nhìn thấy xa hơn, chỉ không chỉ gói gọn trong vấn đề hiện tại, còn người viết mã thì quan tâm thực hiện những vấn đề hiện tại để viết mã tốt.
  - Khi mã code bị tắc đơn giản là họ có thể đổi chỗ cho nhau và như thế có thể là một cách giải quyết vấn đề tắc nghẽn của viết mã.

11. Một số lý do để ngôn ngữ C++ thông dụng

- Một trong những nguyên nhân C++ trở thành phổ biên là nó đã chuyển ngôn ngữ lập trình C thành ngôn ngữ lập trình hướng đối tượng.
- C++ là ngôn ngữ kế thừa và mở rộng từ ngôn ngữ C, trên thực tế không đưa vào các mô hình lập trình mới. Tất cả các chương trình đã có sẵn của C sẽ chạy được trong C++
- C++ có những đặc tính phát triển tốt hơn ngôn ngữ C:
  - TRong C++ có đặc tính tham chiếu của các biến (*references)* cho phép quản lý địa chỉ của các biến hay lấy ra giá trị của các biến ở các địa chỉ tương ứng.
  - Quản lý tên hàm và biển đã được mở rông thông qua cơ chế chồng hàm *function overloading*,

11. Một số lý do để ngôn ngữ C++ thông dụng

- Tư tưởng phân vùng các biến *namespaces* cho phép quản lý các biến được tốt hơn.
- Tính hiệu quả
- Các phần mềm xây dựng trở nên dễ hiểu hơn
- Hiệu quả sử dụng của các thư viện
- Khả năng sử dụng lại mã thông qua templates
- Quản lý lỗi
- Cho phép xây dựng các phần mềm lớn hơn

JAVA

- Tính khả chuyển cao: "Viết một lần, chạy bất cứ đâu"
- Cơ bản cú pháp khá giống C++
- Độ an toàn cao:
  - Trong Java, khái niệm con trỏ (pointer) được thay bằng tham chiếu đối tượng. Người lập trình không có quyền can thiệp trực tiếp lên các địa chỉ phần cứng.
  - Hỗ trợ quản lý bộ nhớ. (Garbage collector)
- Bộ thư viện chất lượng tốt, đầy đủ và liên tục cập nhật
- Miễn phí - Ngôn ngữ của giới "nghiên cứu"
- Tốc độ chấp nhận được (80% so với C++)
- Cho phép xây dựng các ứng dụng ở quy mô lớn (Enterprise features) – J2SEE

13. Quá trình dịch một mã nguồn phần mềm

- Mọi ngôn ngữ lập trình đều được xây dựng theo một cú pháp với các từ khoá dễ hiểu cho người sử dụng. Vì thế để chuyển từ những mã nguồn của ngôn ngữ lập trình sang ngôn ngữ mã máy (machine instructions) cần tới chương trình dịch (translators). Thông thường chương trình dịch được chia làm hai loại: thông dịch (interpreters) và biên dịch (compilers).
- Interpreters: Thông dịch dịch các mã nguồn thành các công việc và máy tính (CPU) sẽ thực hiện ngay những lệnh này.
  - Ưu điểm:
    - Khả năng sửa lỗi tức thì (vì mã nguồn có sẵn khi gặp lỗi)
    - Thích hợp cho những công cụ phát triển phần mềm nhanh
  - Nhược điểm: Cần nhiều tài nguyên của máy tính hơn

13. Quá trình dịch một mã nguồn phần mềm

- Biên dịch dịch trực tiếp các mã nguồn thành các ngôn ngữ mã máy. Kết quả là chúng ta thu được một tệp chứa các mã máy. Thông thường quá trình biên dịch thực hiện qua một số bước và thường phức tạp hơn so với thông dịch. Một số biên dịch dịch các thư viện thành các chương trình riêng biệt sau đó có một công cụ liên kết (linker) liên kết những chương trình riêng biệt này thành chương trình thực hiện được. Quá trình này được gọi là biên dịch độc lập. Biên dịch độc lập thể hiện nhiều ưu điểm:
  - Cho phép khả năng tạo các thư viện
  - Trợ giúp cho việc sửa và gỡ rối các lỗi (khi thực hiện biên dịch từng chương trình nhỏ)

14. Các đặc điểm của quá trình biên dịch độc lập

- Các công cụ biên dịch tự động đóng vai trò rất quan trọng trong quá trình phát triển các phần mềm lớn.
- Thông thường chúng ta thường chia các phần mềm lớn thành các thành phần nhỏ hơn. Trong C, C++, Java đấy là các hàm (*function).*
- Khi xây dựng hàm chúng ta chú ý tới các yếu tố sau:
  - Tên hàm
  - Đối số của hàm
  - Kiểu dữ liệu trả về của hàm

14. Các đặc điểm của quá trình biên dịch độc lập

- Như thế việc đầu tiên là chúng ta phải khai báo và định nghĩa các biến và các hàm.
  - Khai bao hàm (Function declaration syntax)

    Trong C/C++ chúng ta có thể khai báo hàm như sau:

    int func1(int,int);
  - Định nghĩa hàm (Function definitions

    int func1(int length, int width) { …….. }
  - Khai báo biến (Variable declaration syntax): int a;
  - Khai báo các thư viện sử dụng (Including headers)

14. **Các đặc điểm của quá trình biên dich độc lập**

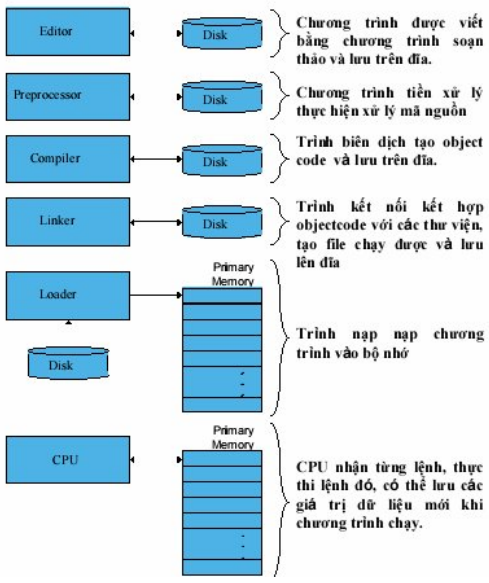
- C/C++ sử dụng các *header file*. Header file chứa các giao diện (khai báo về các biến các hằng số và các hàm sử dụng trong thự viện tương ứng. Thông thường các header file có phần mở rộng của tên là .h (**headerfile.h)**
- Khi chúng ta muốn sử dụng thư viện nào thì gọi tên thư viên đó bằng cách trong phần mềm của mình chúng ta sẽ đưa vào từ khóa: **#include**
- **Một số thư viện chuẩn thông dụng của C++**

  iostream.h cstdlib.h conio.h

Với C++

Các giai đoạn của chương trình C++:

1. Soạn thảo - Edit
2. Tiền xử lý - Preprocess
3. Biên dịch - Compile
4. Liên kết - Link
5. Nạp - Load
6. Chạy - Execute



Hello World (C++)

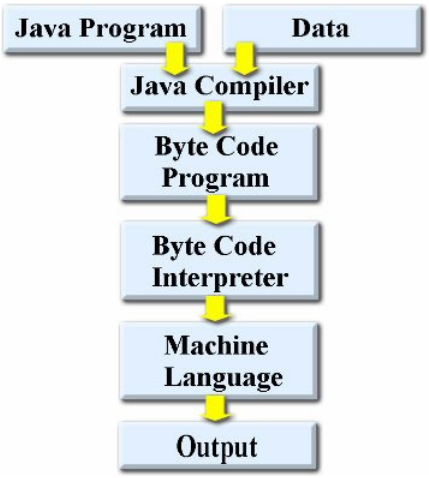
```
#include <iostream>
#include <string>

int main (int argc, char *argv [])
{
    std::cout << "Who are you? ";
    std::string name;
    std::cin >> name;
    if (name. length () > 10)
        std::cout << "You have a long name, " << name << ".";
    else
        std::cout << "Hello, " << name <<  ".";

    return 0;
}
```

#include = declarations of data structures and methods to be inlined, parsed, compiled, and linked with the rest of the code

main routine is typed int so that it can return a status code
0=success
non-zero = failed

Với Java

- Vừa biên dịch, vừa thông dịch
- Biên dịch từ mã nguồn → mã Bytecode (mã máy của máy ảo Virtual Machine)
- Thông dịch từng lệnh trong mã ByteCode → mã lệnh máy trong máy tính chương trình được dịch.



Hello World (Java)

import = names from io library can be used without qualification

In Java, all functions must be bound to a class

Complicated I/O because Java doesn't privilege stream-based IO.

Absence of input data during a read operations is considered exceptional

```
import java. io. *;
class Name {
    public static void main (String args []) {
        InputStreamReader inStream = new InputStreamReader (System.in);
        BufferedReader input = new BufferedReader (inStream);
        PrintStream output = System. out;

        try {
            output. print ("Who are you? ");
            String name = input. readLine ();
            if (name. length () > 10)
                output. println ("You have a long name, " + name + ".");
            else
                output. println ("Hello, " + name + ".");
        }
        catch (Exception e) {
            output. println ("Error.");
        }
    }
}
```


15. Ví dụ, câu hỏi, bài tập

Các câu hỏi tuần 1:

1. Khái niệm đối tượng và các tính chất của đối tượng
2. Các khái niệm liên quan: thực hiện ẩn, khởi tạo, giải phóng đối tượng
3. Phân tích thiết kế hướng đối tượng
4. Khái niệm UML, CRC
5. Khái niệm **Extreme programming**
6. Quy trình dịch một mã nguồn phần mềm và các khái niệm liên quan

15. Ví dụ, câu hỏi, bài tập

Bài tập tuần 1:

(1) Làm quen với một công cụ lập trình (C++, VC, Java, C#)

(2) Viết chương trình nhập mảng các số nguyên, sắp xếp (theo các giải thuật khác nhau) và tìm kiếm (theo các giải thuật khác nhau) một số nhập vào từ bàn phím và đếm xem số này xuất hiện bao nhiêu lần trong mảng