**CÁC THÀNH PHẦN CƠ BẢN CỦA JAVA**

1. **Giới thiệu Java**

Java được phát triển bởi James Goshing, Sun Micro Systems

Phiên bản Java: 19 (9/2022) 🡪 Open JDK và Oracle Java

Đặc điểm:

* Java là ngôn ngữ thuần LTHĐT
* Đa nền tảng (multi-platform)
* Đơn giản, an toàn, linh hoạt
* Các ứng dụng
  + Desktop, Web, Mobile
  + Tính toán hiệu năng cao (HPC)
  + Hệ thống phân tán
  + Big data
  + IoT

1. **Từ khóa**

if, else, for, while, continue, break, import, class, try, catch …

1. **Kiểu dữ liệu**

***\*Kiểu dl cơ bản:***

* Số nguyên: byte, short, int, long
* Số thực: float, double
* Ký tự: char
* Luận lý: boolean

***\*Kiểu dl tham chiếu:***

* String
* List, Vector….
* Các lớp do người dùng định nghĩa
* Wrapper classes: …

1. **Biến – hằng**
2. **Biến**

Khai báo: <kiểu> <tên biến> [=<giá trị>]

1. **Hằng**

Khai báo: ***final*** <kiểu> <tên biến> = <giá trị>;

1. **Biểu thức – Khối lệnh**
2. **Biểu thức**

***\*Các phép toán:***

* Gán: =
* Phép toán số học: +, -, \*, /, %
* Phép toán viết tắt: +=, -=, \*=, /=, %=
* Phép toán tăng giảm: ++, --
* Phép toán quan hệ: <, >, <=, >=, ==, !=
* Phép toán luận lý: &&, ||, !

1. **Khối lệnh**

{ }

1. **Xuất nhập trong Java**
2. **Xuất**

* System.out.print( biểu thức );
* System.out.println( biểu thức );

1. **Nhập**

* import java.util.Scanner;
* Scanner sc = new Scanner(System.in);
* Nhập liệu từ bàn phím

1. **Viết 1 chương trình Java**
2. **Các cấu trúc điều khiển**
3. **Cấu trúc điều kiện**

**a)if**

if (<biểu thức điều kiện>){

//khối lệnh 1;

}

else{

//khối lệnh 2;

}

**b)switch…case**

1. **Cấu trúc lặp**

**a)for**

for(<biểu thức khởi tạo>; <biểu thức điều kiện>; <biểu thức lặp>){

//khối lệnh;

}

**b)while**

<biểu thức khởi tạo>

while(<biểu thức điều kiện>){

//khối lệnh;

<biểu thức lặp>;

}

**c)do…while**

do{

//khối lệnh;

<biểu thức lặp>;

}while (<biểu thức điều kiện>);

1. **Phương thức** (method)
2. **Khai báo**

<phạm vi> <kiểu> <tên phương thức> (danh sách đối số){

//khối lệnh;

[return <biểu thức>;]

}

***\*Lưu ý:***

* Để gọi 1 phương thức hay hàm thì phải được gọi từ 1 đối tượng
* 1 phương thức tĩnh có thể đc gọi từ tên lớp (1 phương thức bình thường thì ko đc gọi từ tên lớp)
* 1 hàm tĩnh chỉ có thể đc gọi tên trong 1 hàm tĩnh và chỉ có thể truy cập các thuộc tính tĩnh

1. **Tái định nghĩa/ Nạp chồng**

Tái định nghĩa cho phép người dùng định nghĩa nhiều hàm trùng tên nhưng khác đối số

Có 3 cách để thực hiện tái định nghĩa 1 phương thức:

* Thay đổi kiểu đối số
* Thay đổi số đối số
* Thay đổi thứ tự đối số

***\*Lưu ý:***

* Lỗi mơ hồ: khi 1 lời gọi có hơn 1 định nghĩa phương thức thỏa mãn lời gọi

🡪Khắc phục lỗi mơ hồ: Định nghĩa thêm 1 phương thức có kiểu đổi số tương thích kiểu của lời gọi

1. **Mảng – chuỗi**
2. **Mảng**

Khai báo:

<kiểu> <tên biến>[ ];

<kiểu> [ ]<tên biến>;

1. **Chuỗi**

Khai báo: String <tên biến> [= “<giá trị>”];

***\*1 số phương thức xử lý chuỗi***

* int length(): chiều dài của chuỗi
* String trim(): chuỗi mà bỏ khoảng trắng đầu và cuối chuỗi
* int indexOf (String s2): vị trí xuất hiện của chuỗi s2 trong s1
* int lastIndexOf (String s2): vị trí xuất hiện của chuỗi s2 trong s1 từ cuối ra đầu chuỗi
* String subString (int b [, int e]): trả về chuỗi con từ vị trí b [đến e] hoặc đến cuối
* boolean equals (String s2): so sánh nội dung 2 chuỗi có giống ko?
* boolean equalsIgnoreCase (String s2): so sánh nội dung 2 chuỗi (ko phân biệt hoa thường) có giống ko?
* int compareTo (String s2): so sánh giá trị của 2 chuỗi
  + <0: s1<s2
  + =0: s1=s2
  + >0: s1>s2
* int compareToIgnoreCase (String s2): so sánh giá trị của 2 chuỗi (ko phân biệt hoa thường)
* String []split (String s2):

1. **Ngoại lệ**

Cú pháp:

try { khối lệnh 1;}

catch (Exception x2) { khối lệnh 2;}

catch (Exception x3) { khối lệnh 3;}

**LÝ THUYẾT LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

1. **Giới thiệu**

* OOP: Object – Oriented Programming
* Định nghĩa: LTHĐT là 1 phương pháp lập trình dựa trên việc mô hình hóa bài toán thành các đối tượng. Sau đó, ánh xạ các đối tượng vào chương trình máy tính thông qua 1 ngôn ngữ lập trình

1. **Một số khái niệm**
2. **Lớp** (Class)

ĐN: Lớp là sự tổng quát hóa của nhiều đối tượng có cùng đặc trưng (gồm: thuộc tính và phương thức). Mỗi lớp đc xem như là 1 kiểu dl

1. **Đối tượng**

ĐN: Đối tượng là sự cụ thể hóa (thể hiện) của 1 lớp. 1 đối tượng đc xem như là 1 biến đc tạo ra từ 1 kiểu dl lớp nào đó

1. **Thuộc tính** (Attribute/ Data)

ĐN: Thuộc tính đc xem như là thành phần dl đc sử dụng để lưu trữ trạng thái tại 1 thời điểm của 1 đối tượng

1. **Phương thức** (Method/ Function)

ĐN: Phương thức là thành phần xử lý trên các thuộc tính của lớp hay đối tượng. Nó được sử dụng để truy cập hay cập nhật các giá trị trên các thuộc tính

1. **Phạm vi truy cập** (Access modifier)

Phạm vi truy cập xác định đường truy xuất đến các thành phần của 1 lớp hay đối tượng

Có 3 kiểu PVTC:

* - private
* # protected
* + public

1. **Thông điệp – truyền thông điệp** (Message passing)

Thông điệp là 1 yêu cầu để thực hiện 1 hành vi nào đó

Truyền thông điệp: là báo cho đối tượng đó biết phải làm gì. Khi 1 đối tượng nhận đc 1 thông điệp, nó sẽ kích hoạt 1 phương thức tương ứng

1. **Các tính chất của LTHĐT** (Characteristics)
2. **Tính bao gói** (Encapsulation)

Tính bao gói thể hiện ở 2 khía cạnh:

* Gom nhóm các TT và PT vào 1 lớp hay ĐT
* Phạm vi truy cập: xác định đường truy xuất đến các thành phần của 1 lớp hay đối tượng

***\*Phạm vi truy cập:***

***- private:*** thành phần riêng, chỉ đc truy cập trong cùng lớp

***# protected:*** thành phần riêng đc bảo vệ, đc truy cập trong cùng lớp, cùng gói và các lớp con cháu

***+ public:*** thành phần chung, đc truy cập trong và ngoài lớp (ko giới hạn)

1. **Tính đa hình** (Polymorphism)

Là 1 thành phần sẽ có đc ý nghĩa hoặc xử lý khác nhau trong môi trường khác nhau

Có 2 cách thực hiện tính đa hình:

* Nạp chồng/ Tái định nghĩa (Overloading): là cơ chế cho phép chúng ta định nghĩa nhiều hàm trùng tên nhưng khác đối số
* Nạp đè (Overriding): là cơ chế cho phép ta định nghĩa nhiều hàm trùng tên và trùng đối số nhưng đc định nghĩa trong lớp con cháu

1. **Tính thừa kế** (Inheritance)

1 lớp con C thừa kế từ lớp cha A thì nó sẽ có đc tất cả các thành phần của lớp cha A nhưng chỉ truy cập đc các thành phần đc bảo vệ và chung của lớp cha A

***\*Có 2 dạng thừa kế:***

* Đơn thừa kế: thừa kế từ 1 lớp (JAVA)
* Đa thừa kế: thừa kế từ nhiều lớp

***\*Chú ý:*** Các thành phần của lớp con chỉ truy cập đc thành phần đc bảo vệ và thành phần chung của lớp cha. Tuy nhiên, các thành phần lớp cha ko truy cập đc các thành phần lớp con, ngoại trừ thành phần bị nạp đè

1. **Tính trừu tượng hóa** (Abstraction)

Là việc mô hình hóa bài toán thành các đối tượng. Quá trình xác định thuộc tính, phương thức và mối quan hệ giữa các đối tượng đgl quá trình trừu tượng hóa

**LTHĐT VỚI JAVA**

**CHƯƠNG 1: LỚP – ĐỐI TƯỢNG – PHƯƠNG THỨC XÂY DỰNG**

1. **Lớp**

***\*ĐN:*** Lớp là sự tổng quát hóa của nhiều đối tượng có cùng đặc trưng (gồm: thuộc tính và phương thức). Mỗi lớp đc xem như là 1 kiểu dl

***\*Khai báo:***

<phạm vi> class <tên lớp>{

//thuộc tính

<phạm vi> <kiểu> <tên TT>;

//phương thức

<phạm vi> <kiểu> <tên PT>{

//định nghĩa PT

}

}

***\*Phạm vi truy cập:***

* -private: đc truy cập trong cùng lớp
* ●default: đc truy cập trong cùng lớp, cùng gói
* #protected: đc truy cập trong cùng lớp, cùng gói, lớp con cháu
* +public: ko giới hạn

1. **Đối tượng**

\*ĐN: Đối tượng là sự cụ thể hóa (thể hiện) của 1 lớp. 1 đối tượng đc xem như là 1 biến đc tạo ra từ 1 kiểu dl lớp nào đó

***\*Khai báo:***

<tên lớp> <tên đối tượng>;

***\*Truy cập các thành phần ĐT:***

<tên ĐT> . <thành phần> [(đối số)];

***\*Mảng ĐT:***

1. **Phương thức xây dựng** (Constructor)

***\*ĐN:***

* PTXD là PT thành viên của 1 lớp đc gọi tự động mỗi khi tạo ra ĐT
* PTXD có tên trùng tên lớp, có thể có đối số, nhưng ko có kiểu trả về
* Nếu ko định nghĩa hàm xây dựng, Java tự tạo hàm default ko đối số

***\*Khai báo:***

public class A{

…

public A (đối số){

//cấp phát bộ nhớ cho các TT đối tượng

//đặt giá trị khởi tạo cho các TT

}

…

}

**CHƯƠNG 2: PHƯƠNG THỨC – TÁI ĐỊNH NGHĨA**

1. **Phương thức** (method)

***\*ĐN:*** Phương thức là thành phần xử lý trên các thuộc tính của lớp hay đối tượng. Nó được sử dụng để truy cập hay cập nhật các giá trị trên các thuộc tính

***\*Khai báo:***

<fvi> class <tên lớp>{

…

<fvi> <kiểu> <tên PT> (đối số){

//định nghĩa PT

}

…

}

1. **Tái định nghĩa** (Overloading)

***\*ĐN:*** Tái định nghĩa là cơ chế cho phép chúng ta định nghĩa nhiều hàm trùng tên nhưng khác đối số

* Thay đổi kiểu đối số
* Thay đổi số đối số
* Thay đổi thứ tự đối số

***\*Khai báo:***

<fvi> class <tên lớp>{

…

<fvi> <kiểu> f (đối số 1){…}

<fvi> <kiểu> f (đối số 2){…}

…

}

1. **Phương thức xây dựng sao chép** (Copy construction)

***\*ĐN:*** Phương thức xây dựng sao chéo là PTXD đc gọi từ động khi tạo 1 ĐT từ 1 ĐT đã có

***\*Khai báo:***

<fvi> class <tên lớp>{

…

<fvi> A (A a){

//sao chép giá trị của các TT từ a

}

…

}

1. **Phương thức tĩnh** (static method)

* PT tĩnh có thể đc gọi từ tên lớp hoặc ĐT
* PT tĩnh chỉ có thể gọi PT tĩnh và chỉ có thể truy cập các TT tĩnh

**CHƯƠNG 3: THUỘC TÍNH ĐỐI TƯỢNG – SAO CHÉP ĐỐI TƯỢNG**

1. **Thuộc tính đối tượng** (Quan hệ bao gồm / thành phần)

Lớp mới có bao gồm lớp cũ?

VD: DoanThang bao gồm Diem

DuongTron bao gom Tam, BanKinh

1. **Sao chép đối tượng**

***Sao chép cạn*** (Shallow copy) là thực hiện phép toán gán bằng cho từng thuộc tính của 2 ĐT

***Sao chép sâu*** (Deep copy) là thực hiện sao chép giá trị của từng thuộc tính của 2 ĐT

1. **Thuộc tính tĩnh**

***\*ĐN:*** Thuộc tính tĩnh là TT dùng chung cho tất cả các ĐT của lớp đó

**CHƯƠNG 4: THỪA KẾ - ĐA HÌNH – TRỪU TƯỢNG**

1. **Thừa kế**

***\*ĐN:*** 1 lớp con C thừa kế từ lớp cha A thì nó sẽ có đc tất cả các thành phần của lớp cha A nhưng chỉ truy cập đc các thành phần đc bảo vệ và chung của lớp cha A

***\*Khai báo:***

<fvi> class A { *//superclass*

//các thành phần lớp A

}

<fvi> class B ***extends*** A { *//subclass*

//bổ sung thêm các thành phần của A

}

***\*Có 3 cách để tạo lớp con:***

* Thêm thuộc tính (đặc trưng hóa)
* Thêm phương thức (thêm tùy chọn)
* Nạp chồng/ nạp đè phương thức

1. **Đa hình**

***\*ĐN:*** Là 1 thành phần sẽ có đc ý nghĩa hoặc xử lý khác nhau trong môi trường khác nhau

***\*Có 2 cách:***

* Nạp chồng (Overloading)
* Nạp đè (Overriding)

***\*Chú ý:***

* 1 tham chiếu cha có thể tham chiếu đến các ĐT của lớp cha và các ĐT của lớp con. Tuy nhiên, tham chiếu cha ko thể truy cập đc các thành phần của lớp con ngoại trừ thành phần bị nạp đè
* 1 tham chiếu con chỉ có thể tham chiếu đến các ĐT của lớp con mà ko thể tham chiếu tới các ĐT của lớp cha. Tuy nhiên, nó có thể truy cập đc các thành phần của lớp cha (tp chung, tp đc bảo vệ và mặc nhiên nếu trong cùng gói)

1. **Trừu tượng**

***\*ĐN:*** Là việc mô hình hóa bài toán thành các đối tượng. Quá trình xác định thuộc tính, phương thức và mối quan hệ giữa các đối tượng đgl quá trình trừu tượng hóa

1. **Lớp trừu tượng của phương thức trừu tượng**

***\*Khai báo:***

<fvi> ***abstract*** class A {

//các tp của A

<fvi> ***abstract*** <kiểu> f (đối số);

}

public class C ***extends*** A {

//các tp mở rộng của A

<fvi> <kiểu> f (đối số){

//định nghĩa hàm trừu tượng

}

}

***\*Chú ý:***

* Lớp trừu tượng là 1 lớp tổng quát cho các lớp khác. Nó ko đc dùng để tạo ra các ĐT mà đc dùng để tạo ra lớp con và tham chiếu
* 1 PT trừu tượng là 1 PT chỉ có phần khai báo mà ko có phần định nghĩa. Nó phải đc định nghĩa trong lớp con cháu
* 1 lớp trừu tượng có thể ko chứa hàm trừu tượng nào. Tuy nhiên, 1 lớp mà có chứa hàm trừu tượng thì lớp đó phải là lớp trừu tượng

1. **Giao diện**

***\*ĐN:***

* Nó đc xem như là 1 cam kết cung cấp các tính năng hoặc tùy chọn đến người dùng. Nó đc dùng để tạo ra các lớp và các tham chiếu mà ko đc dùng để tạo ra các ĐT
* Giao diện bao gồm tập các khai báo PT chung
* *Ko đòi hỏi quan hệ cha con, chỉ cần có cam kết sử dụng chung interface*

***\*Khai báo:***

<fvi> ***interface*** I1 {

<fvi> <kiểu> <tên PT> (đối số);

…

}

public class A ***implements*** I1, I2, I3… {

//thuộc tính và phương thức

//định nghĩa các PT giao diện

}

***\*Chú ý:***

* 1 tham chiếu giao diện có thể tham chiếu đến nhiều ĐT có kiểu khác nhau miễn sao các kiểu lớp đó phải thực thi cùng giao diện
* 1 tham chiếu giao diện chỉ có thể truy cập các PT của giao diện
* 1 lớp có thể thực thi 1 hoặc nhiều giao diện

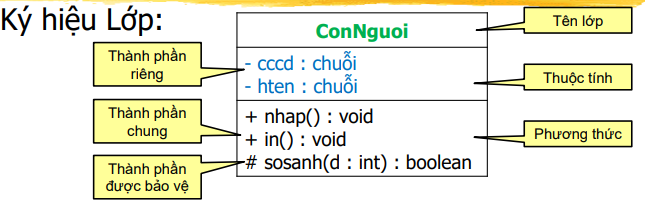
**XÂY DỰNG ỨNG DỤNG HƯỚNG ĐỐI TƯỢNG VỚI JAVA**

1. **Giới thiệu**

OOP là:

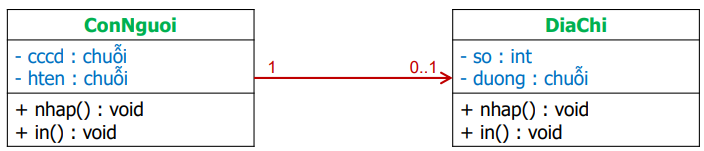
* Mô hình hóa bài toán 🡪 lớp / ĐT
* Chuyển mô hình 🡪 chương trình bởi OOP

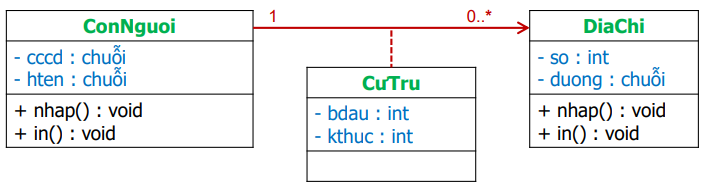
1. **Mô hình hóa – UML class Diagram**
2. **Ký hiệu lớp**



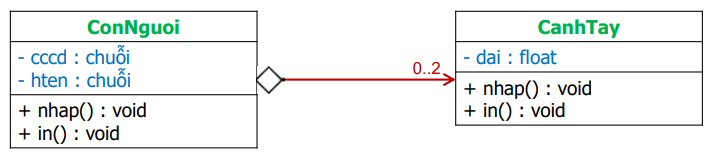
1. **Các kiểu quan hệ**

**a) Association** (Gắn kết)

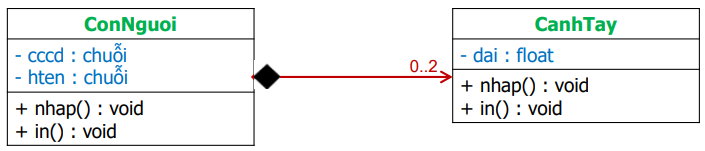


****

**b) Aggregation** (Bao gồm) : *có thể thay thế đc*



**c) Composition** (Thành phần) : ***final*** *(bắt buộc phải có; Là trường hợp đặc biệt của bao gồm nhưng có sự ràng buộc)*



**d) Generalization** (Thừa kế)



**e) Realization** (Thực thi)

