NGÔN NGỮ LẬP TRÌNH JAVA

Giảng viên: TS. Vũ Hữu Tiến
Email: tienvh@ptit.edu.vn
Mobile: 0939919396

GIỚI THIỆU MÔN HỌC

- Nội dung môn học:
  - **Chương 1:** Tổng quan về lập trình hướng đối tượng và ngôn ngữ lập trình Java
  - **Chương 2:** Lớp và đối tượng trong Java
  - **Chương 3:** Tính kế thừa và đa hình trong Java
  - **Chương 4:** Xử lý nhập/ xuất trong Java
  - **Chương 5:** Xử lý ngoại lệ trong Java
  - **Chương 6:** Lập trình đa luồng trong Java
  - **Chương 7:** Lập trình giao diện trong Java

LỚP VÀ ĐỐI TƯỢNG

- Lớp & đối tượng
  - Lớp là một "bản thiết kế" của một đối tượng nào đó. Lớp mô tả khái niệm về các đối tượng và bất cứ đối tượng cụ thể nào được tạo ra từ lớp đều là một hiện thực hóa khái niệm đó.
  - Ví dụ: Lớp *Sinhvien* mô tả về sinh viên của một trường đại học. Đối tượng *Sinhvien1* được tạo ra từ lớp *Sinhvien* mô tả về một sinh viên cụ thể nào đó của trường.

LỚP VÀ ĐỐI TƯỢNG

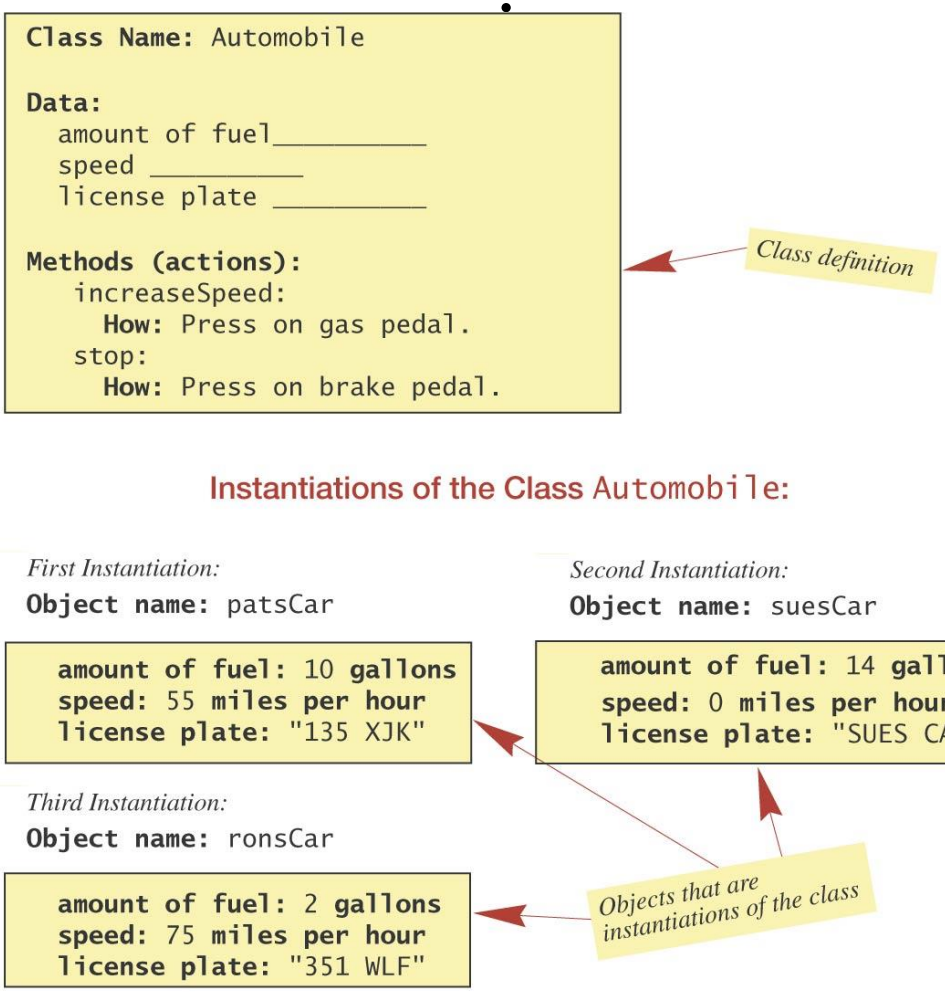
- Lớp & đối tượng
  - Mỗi đối tượng đều có trạng thái *(state)* được định nghĩa bởi các giá trị thuộc tính *(attribute)* đi kèm với đối tượng đó.
  - Ví dụ: đối tượng *Sinhvien* có thuộc tính là *tên, tuổi, chuyên ngành, điểm trung bình.*
  - Trong Java, các thuộc tính của đối tượng được định nghĩa bằng các biến trong một lớp.
  - Mỗi đối tượng đều có các hoạt động đi kèm với nó.
  - Ví dụ: đối tượng *Sinhvien* có các hoạt động là *Cập nhật địa chỉ, Tính điểm trung bình.*
  - Trong Java, hoạt động của đối tượng được định nghĩa bằng các phương thức *(method)* được khai báo trong một lớp.

LỚP VÀ ĐỐI TƯỢNG

- Lớp bao gồm:
  - Thuộc tính
  - Phương thức
- Ví dụ:
  - Lớp *Person* gồm:
    - Attributes: *name*, *age*, *address*, *phoneNumber*
    - Methods: *changeAddress*, *changePhoneNumber*

LỚP VÀ ĐỐI TƯỢNG

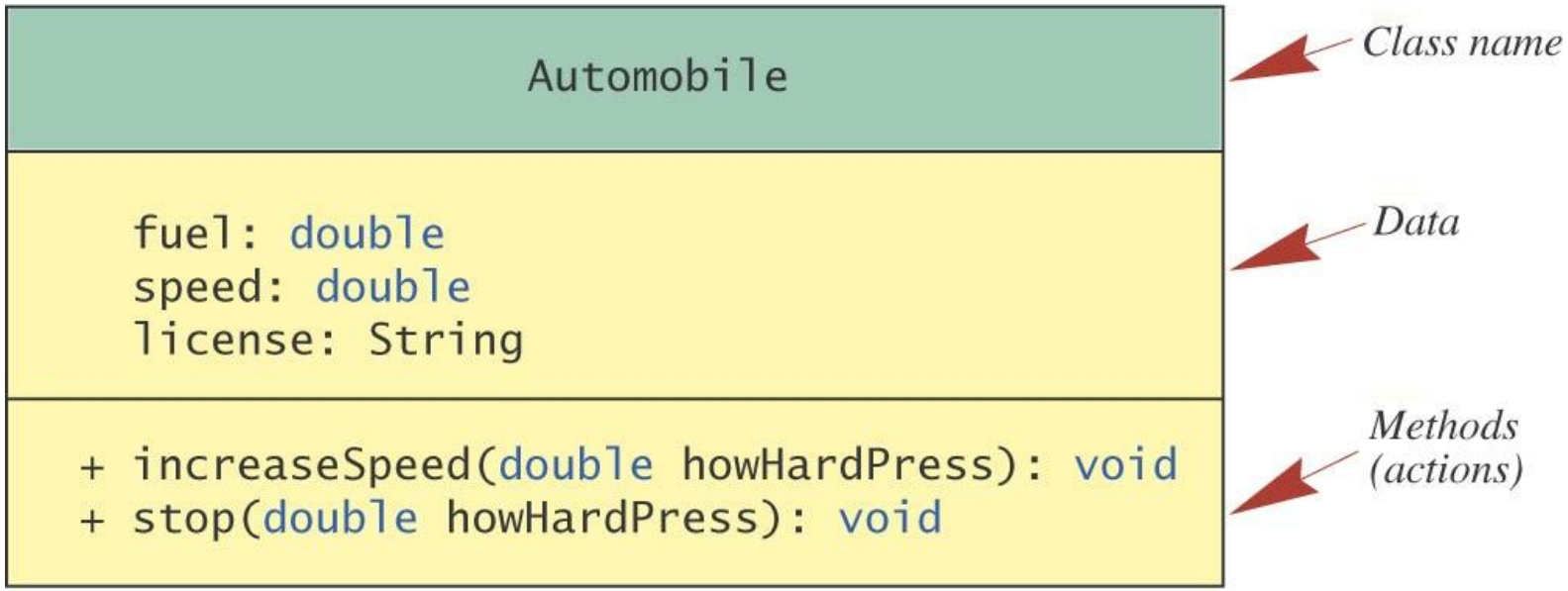
- Lớp được hiểu như một bản thiết kế của đối tượng:



Display 4.1

LỚP VÀ ĐỐI TƯỢNG

- Mô tả lớp bằng ngôn ngữ UML:



Display 4.2

A Class Outline as a UML Class Diagram

LỚP VÀ ĐỐI TƯỢNG

- Khai báo lớp và khởi tạo đối tượng
  - Khai báo:

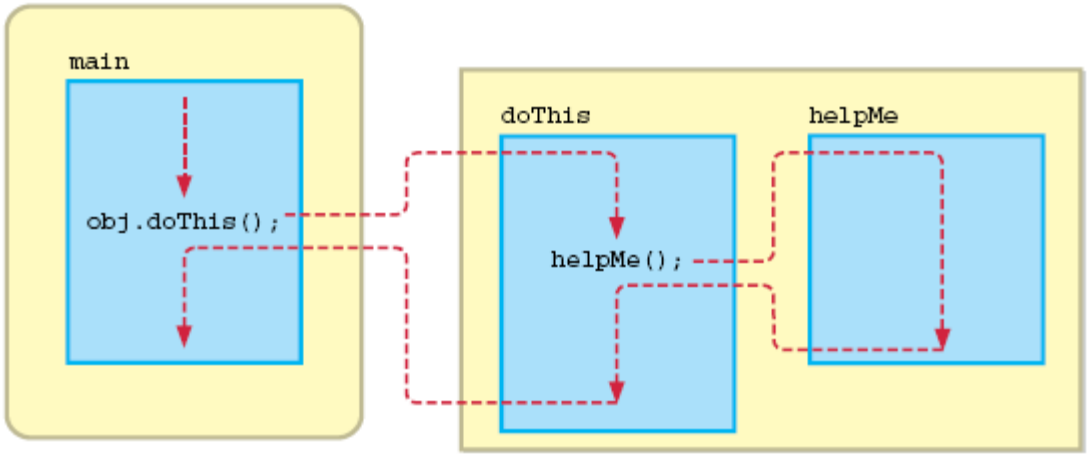
    Modifier class ten_lop{

    }
  - Nếu một chương trình gồm 2 lớp trở lên thì có thể
    - Đặt tất cả lớp vào cùng một file. Lưu ý là lớp đầu tiên trong file phải được khai báo là public, các lớp sau không được là public.
    - Đặt mỗi lớp vào một file riêng của nó. Lưu ý là tên lớp trùng với tên file và tất cả các lớp đều phải được khai báo là public.
  - Khởi tạo:

    ten_lop *ten_doi_tuong* = new ten_lop();

LỚP VÀ ĐỐI TƯỢNG

- Phương thức:
  - Phương thức mô tả hoạt động của đối tượng khi phương thức được gọi.
  - Phương thức được khai báo và định nghĩa trong phạm vi của một lớp.
  - Nếu phương thức được gọi trong phạm vi của lớp thì chỉ cần gọi tên của phương thức. Nếu phương thức thuộc lớp khác thì khi gọi cần gọi cả tên của lớp đó.



LỚP VÀ ĐỐI TƯỢNG

- Kiểu trả về của phương thức
  - Kiểu trả về của phương thức có thể là kiểu giá trị nguyên thủy (int, float, double,…) hoặc kiểu void. Mặc định kiểu của phương thức là kiểu void.
  - Câu lệnh trả về giá trị cho phương thức nằm ở cuối phương thức và có cú pháp:

    return *biểu_thức;*
  - Giá trị của biểu_thức phải cùng kiểu với phương thức.
  - Ví dụ:

```
return;
return distance*4;
```

LỚP VÀ ĐỐI TƯỢNG

- Tham số của phương thức
  - Tham số là giá trị được truyền tới phương thức khi phương thức được gọi.
  - Tên của các tham số khi khai báo phương thức gọi là *tham_biến*. Giá trị truyền vào cho các tham số khi gọi phương thức gọi là *tham_trị*. Tham trị còn được gọi là *arguments*.
  - *Ví dụ*

```
Student st1= new Student();
st1.setStudent();
st1.getName();
```

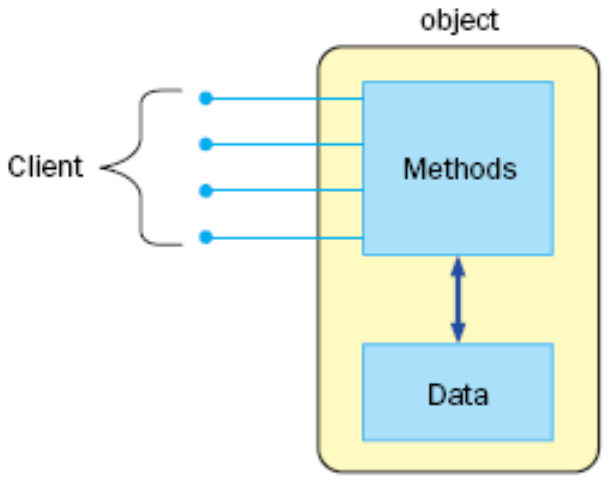
```
Student st1= new Student("Vũ Hữu Tiến",18,2.5f);
st1.getName();
```

LỚP VÀ ĐỐI TƯỢNG

- Phương thức khởi tạo (Constructor)
  - Constructor trong Java là một kiểu phương thức đặc biệt mà được sử dụng để khởi tạo đối tượng.
  - Constructor được triệu hồi tại thời gian tạo đối tượng. Nó xây dựng giá trị, cung cấp dữ liệu cho đối tượng, đó là lý do nó được gọi là Constructor.
  - Contructor khác với phương thức thông thường ở 2 điểm:
    - Tên của constructor cùng tên với lớp
    - Contructor không trả về giá trị khi khai báo

LỚP VÀ ĐỐI TƯỢNG

- Tính bao đóng (Encapsulation)
  - Dữ liệu của một đối tượng chỉ được thay đổi bởi chính đối tượng đó. Đặc tính này gọi là *tính bao đóng*.
  - Chúng ta có thể tạo một lớp được bao đóng hoàn toàn trong Java bằng việc tạo tất cả thành viên dữ liệu của lớp là private
  - Đối tượng có thể giao tiếp với các phần khác của chương trình thông qua các phương thức cụ thể gọi là *giao diện (interface)*.



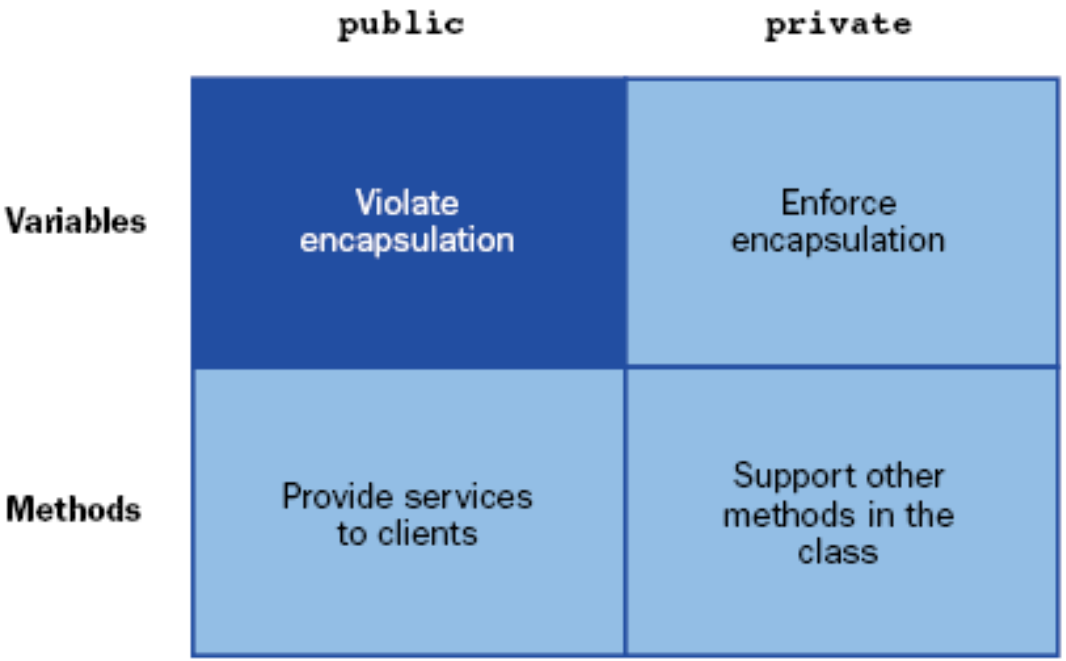
LỚP VÀ ĐỐI TƯỢNG

- Tính bao đóng (Encapsulation)
  - Có hai loại phương thức giao diện: getter & setter
  - Tính bao đóng là kỹ thuật tạo một trường của lớp private và cung cấp khả năng truy cập trường này qua các phương thức pullic
  - Ví dụ:

```
public String getName(){
        return name;
}
public void setName(String name){
        this.name=name;
}
```

LỚP VÀ ĐỐI TƯỢNG

- Phạm vi truy cập của phương thức (Modifier)



LỚP VÀ ĐỐI TƯỢNG

- Từ khóa This
  - Sử dụng trong hàm khởi tạo

```
Student(String name,int age,float point){
        this.name=name;
        this.age=age;
        this.point=point;
}
```

  - Sử dụng để phân biệt biến của các đối tượng

```
public float addPoint(float pointStd){
            return this.point + pointStd;
}
```

LỚP VÀ ĐỐI TƯỢNG

- Từ khóa This
  - Sử dụng trong hàm khởi tạo

```
Student(String name,int age,float point){
        this.name=name;
        this.age=age;
        this.point=point;
}
```

  - Sử dụng để phân biệt biến của các đối tượng

```
public float addPoint(float pointStd){
            return this.point + pointStd;
}
```

LỚP VÀ ĐỐI TƯỢNG

- Static
  - Kiểu dữ liệu *static* thường được sử dụng cho biến và phương thức trong một lớp.
  - Biến static:
    - Trong một lớp có 2 loại biến: *local variable* được khai báo trong mỗi phương thức và *instance variable* được khai báo ngoài phương thức.
    - Biến static hay còn gọi là class variable được chia sẻ cho tất cả các đối tượng của cùng một lớp.
    - Biến *local* trong một phương thức không được khai báo kiểu *static*
    - Khi thay đổi giá trị biến static trong một đối tượng thì giá trị trong tất cả các đối tượng cũng thay đổi theo.

LỚP VÀ ĐỐI TƯỢNG

- Static
  - Phương thức static:
    - Phương thức kiểu *static* có thể gọi thông qua tên của lớp mà không cần phải khởi tạo một đối tượng cụ thể.
    - Vì phương thức *static* không thuộc một đối tượng cụ thể nên phương thức *static* không thể thao tác các biến *instance* mà chỉ thao tác với các biến *static*.
    - Phương thức *main* có kiểu static là vì phương thức *main* được thực thi bởi trình biên dịch mà không cần phải khởi tạo một đối tượng của lớp chính.
    - Phương thức *main* chỉ có thể truy cập biến static hoặc biến nội bộ của phương thức main.

LỚP VÀ ĐỐI TƯỢNG

- Nạp chồng phương thức (method overloading)
  - Trong một lớp có thể có nhiều phương thức trùng tên nhưng khác tham số truyền vào. Kỹ thuật này gọi là nạp chồng phương thức.
  - Các tham số của các phương thức cùng tên có thể khác nhau về số lượng, kiểu dữ liệu hoặc thứ tự các tham số.
  - Ví dụ:
    - Sum(2,3);
    - Sum(2,3,4);

LỚP VÀ ĐỐI TƯỢNG

- Bài tập 1: Viết chương trình chơi xúc sắc. Hai người chơi lần lượt tung xúc sắc. Nếu ai tung được mặt nhiều điểm thì người đó thắng. Chương trình sử dụng lớp xúc sắc Dice gồm
  - Thuộc tính:
    - Điểm trên mặt ngửa của xúc sắc: faceValue
  - Phương thức:
    - Tung xúc sắc: Roll();
    - Lấy giá trị của các mặt xúc sắc: getFaceValue

LỚP VÀ ĐỐI TƯỢNG

- Bài tập 2 : Viết chương trình quản lý sinh viên
  - Thuộc tính:

```
private String name;
private int age;
private int id;
private float[] point;
```

  - Phương thức:
    - Phương thức khởi tạo: Student(String name,int age, int id)
    - Phương thức nhập điểm: public void setPoint()
    - Phương thức tính điểm trung bình: public float getAvrPoint()

LỚP VÀ ĐỐI TƯỢNG

- Bài tập: Viết chương trình "Quản lý sinh viên" với các lớp sau:
  SinhVien.java
  - Gồm các instance variable masv, hoten, gioitinh, diemtoan, diemly, diemhoa với các kiểu dữ liệu phù hợp.
  - Xây dựng các constructor không tham số (mặc định) và có tham số.
  - Xây dựng các phương thức set/get cho từng biến.
  - Xây dựng các phương thức inputSinhVien/displaySinhVien để nhập và hiển thị thông tin về sinh viên
  SinhVienTest.java
  - Đây là một main class.
  - Chương trình được tổ chức theo hệ thống menu chức năng như sau:
  1. Nhập một sinh viên mới.
  2. Xem danh sách sinh viên.
  3. Sắp xếp danh sách theo thứ tự tăng dần theo tongdiem

LỚP VÀ ĐỐI TƯỢNG

- Gói (Package):
  - Một số gói có sẵn của Java:

package	Mô tả
java.lang	Đây là gói mặc định của Java, người lập trình không cần phải import.
java.io	Hỗ trợ các lớp liên quan đến việc xuất/nhập dữ liệu
java.applet	Hỗ trợ các lớp xử lý applet trong trình duyệt
java.awt	Hỗ trợ các lớp tạo giao diện cho ứng dụng
java.util	Cung cấp các lớp và giao diện cho việc tạo các thành phần trong một ứng dụng như lịch, ngày/giờ,…
java.net	Cung cấp các lớp và giao diện dùng cho lập trình mạng.

LỚP VÀ ĐỐI TƯỢNG

- Gói (Package):
  - Gói là thư mục dùng để lưu trữ các lớp.
  - Chương trình Java có thể gồm nhiều gói, mỗi gói gồm nhiều lớp. Các lớp trong cùng một gói có mối liên quan nhất định đến nhau.
  - Khai báo:

```
package package1;
import package2.*;
public class mypackage1 {
}
```

LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - String Class
    - charAt(): trả về ký tự ở vị trí đầu tiên của chuỗi
    - startsWith(): trả về giá trị Boolean phụ thuộc vào phần đầu của chuỗi

```
String strname="Java language";
boolean flag = strname.startsWith("Java"); //true value
```

    - endsWith()
    - copyValueOf()
    - toUpperCase()
    - toLowerCase()
    - equals()

```
char[] name = {'P','T','I','T'};
String subname = String.copyValueOf(name,1,3);
//subname =“TIT”
```

LỚP VÀ ĐỐI TƯỢNG

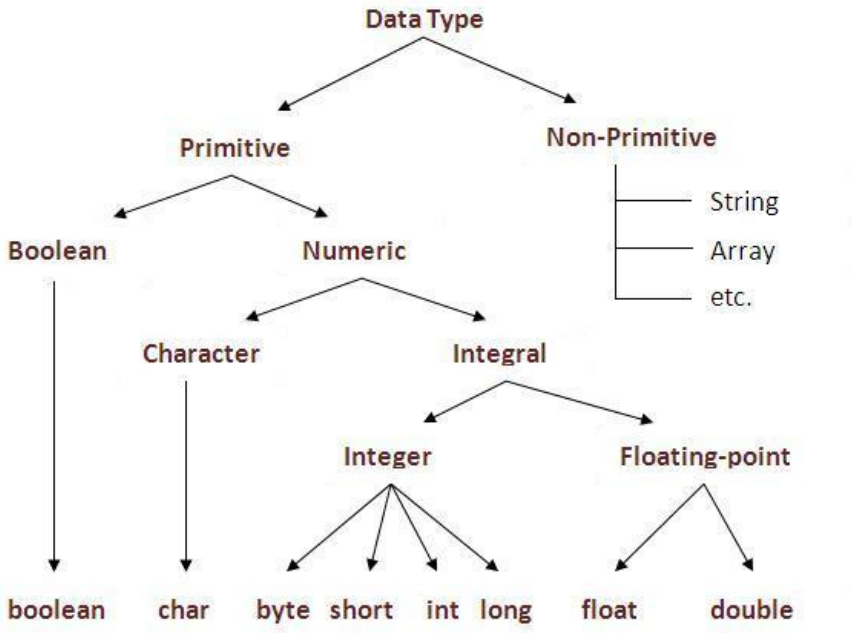
- Gói Java.lang
  - Dữ liệu gốc (primitive) và dữ liệu tham chiếu (objective data)

Dữ liệu gốc: là những dữ liệu đã cố định kích thước ô nhớ để lưu

Dữ liệu tham chiếu:

Dữ liệu đối tượng được tạo bởi sử dụng các constructor đã được định nghĩa của các lớp. Chúng được sử dụng để truy cập các đối tượng. Ví dụ: Employee, Student, …

Giá trị mặc định của bất kỳ biến đối tượng nào đều là null.



LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - Wrapper Class:
    - Các biến kiểu primitive không phải là đối tượng nên không thể truy cập các phương thức.
    - Để sử dụng được các phương thức có sẵn để xử lý dữ liệu primitive, cần phải sử dụng lớp bao.
    - Ví dụ:
      - Nếu khai báo *int num;* thì biến *num* không thể truy cập được phương thức *num.InitValue();*
      - Khi đó phải khai báo: *Integer num = new Integer(0);*

LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - String Class
    - Một số hàm khởi tạo:

```
//Chuỗi rỗng
String str1 = new String();
//Chuỗi "Hello world"
String str2 = new String ("Hello world");
//Chuỗi: "PTIT"
char[] ch = {'P','T','I','T'};
String str3=new String(ch);
//Chuỗi: "PI"
String str4= new String(ch,0,2);
```

LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - StringBuffer Class
    - Phương thức capacity() của lớp StringBuffer trả về dung lượng của bộ nhớ đệm. Dung lượng mặc định của bộ nhớ đệm là 16. Nếu số lượng ký tự của chuỗi tăng lên thì dung lượng được tính theo công thức (dung lượng cũ*2)+2.

```
StringBuffer sb = new StringBuffer();
System.out.println(sb.capacity());//mặc định là 16
 sb.append("1234567812345678");
System.out.println(sb.capacity());//đến đây vẫn là 16
sb.append("java is my favourite language");
System.out.println(sb.capacity());//đến đây là (16*2)+2=34 i.e (dung lượng cũ*2)+2
sb.append("PTIT");
System.out.println(sb.capacity());//đến đây là (34*2)+2=70 i.e (dung lượng cũ*2)+2
```

LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - StringBuffer Class

```
StringBuffer s1 = new StringBuffer();
StringBuffer s2 = new StringBuffer(20);
StringBuffer s3 = new StringBuffer("StringBuffer");
```

System.***out.println("s3 = " + s3);***	s3 = StringBuffer
System.***out.println(s1.length());***	0
System.***out.println(s2.length());***	0
System.***out.println(s3.length());***	12
System.***out.println(s1.capacity());***	16
System.***out.println(s2.capacity());***	20
System.***out.println(s3.capacity());***	28 //16+12

LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - StringBuffer Class
    - insert()
    - setCharAt()
    - getChar()

```
StringBuffer str = new StringBuffer("Java");
str.insert(1,'b'); //Jbava
```

```
StringBuffer str = new StringBuffer("Java");
str.setCharAt(1,'b'); //Jbva
```

```
StringBuffer str = new StringBuffer("Java");
char ch[] = new char[10];
str.getChars(1,3,ch,0); //ch = {a,v}
```

LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - Bài tập: Viết chương trình nhập họ tên, tuổi, email trong đó có các hàm hỗ trợ kiểm tra email có hợp lệ hay không.

LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - Math Class
    - abs()
    - ceil()
    - floor()
    - max()
    - min()
    - round()
    - random()
    - sqrt()

LỚP VÀ ĐỐI TƯỢNG

- Gói Java.lang
  - Bài tập: Viết phương thức hỗ trợ giải phương trình bậc 2.

LỚP VÀ ĐỐI TƯỢNG

- Giao diện (Interface)