BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC GIA ĐỊNH

****KHOA CÔNG NGHỆ THÔNG TIN



**TIỂU LUẬN MÔN HỌC**

**LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG NÂNG CAO**

GVHD: ThS.LÊ HUỲNH PHƯỚC.

NĂM HỌC: 2022.

NHÓM: 11.

THÀNH VIÊN: NGUYỄN ĐỨC TÀI – 2008110304.

PHAN HOÀNG NAM -.2004110054.

NGUYỄN MINH TUẤN – 2008110220.

NGUYỄN HƯỞNG – 2004110008.

Tp. HỒ CHÍ MINH, 08/2022

Mục lục

[***LỜI NÓI ĐẦU.*** 2](#_Toc110498686)

[**Đề Tài: *Hướng Dẫn Và Ví Dụ Java Reflection.*** 3](#_Toc110498687)

[**I.** **Khái Niệm:** 3](#_Toc110498688)

[**II.** **Cú Pháp:** 4](#_Toc110498689)

[**Cách lấy đối tượng của lớp Class của Reflection trong Java** 5](#_Toc110498690)

[**Phương thức của lớp Object getClass()** 5](#_Toc110498691)

[**Cú pháp .class** 6](#_Toc110498692)

[A. **Lấy thông tin bằng cách sử dụng Reflection API trong Java** 6](#_Toc110498693)

[**III.** **Ý Nghĩa:** 6](#_Toc110498694)

[IV. Cấu trúc của 1 Class trong Java Reflection: 7](#_Toc110498695)

[1. Lớp (Classes): 8](#_Toc110498696)

[2. Tạo đối tượng Class<>: 8](#_Toc110498697)

[3. Cấu tử (Constructor): 8](#_Toc110498698)

[4. Trường (Field): 9](#_Toc110498699)

[5. Phương thức (method): 10](#_Toc110498700)

[***LỜI CÁM ƠN*** 11](#_Toc110498701)

# ***LỜI NÓI ĐẦU.***

**Lập trình hướng đối tượng (Object Oriented Programming – OOP)** là một trong những kỹ thuật lập trình rất quan trọng và sử dụng nhiều hiện nay. Hầu hết các ngôn ngữ lập trình hiện nay như Java, PHP, .NET, Ruby, Python... Qua môn học sinh viên có thể hiểu rõ và nắm những kiến thức cơ bản nhất về hướng đối tượng cho riêng mình. Ngoài ra, sinh viên còn được trang bị những kiến thức từ cơ bản đến nâng cao nhằm phục vụ cho quá trình học và nghiên cứu thêm về **Lập trình hướng đối tượng** trong tương lai.

Sau quá trình học tập, trau dồi kiến thức lập trình trong học kỳ vừa qua, nhóm tôi đã ôn tập và tiến hành thực hiện bài tiểu luận này để kết thúc học phần.

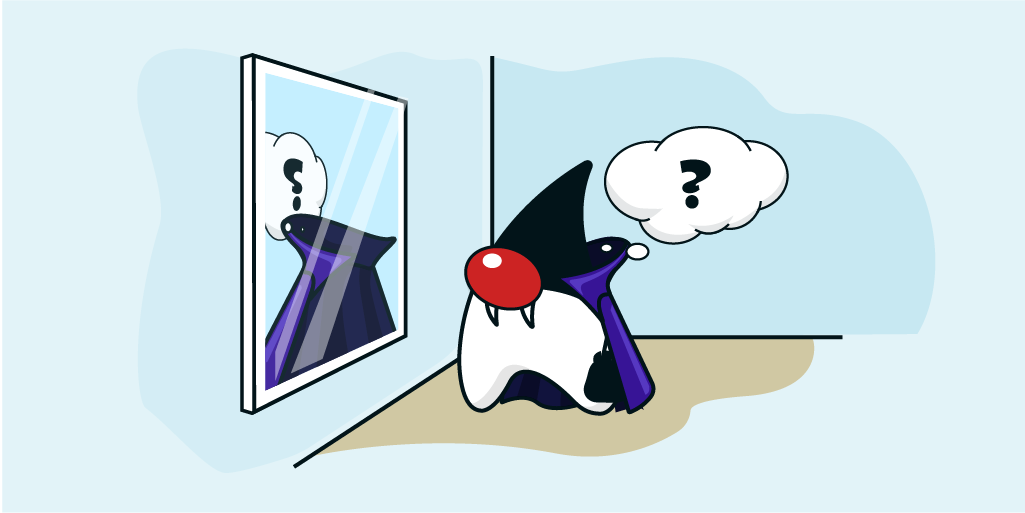
Do hạn chế về mặt thời gian và kiến thức nên bài tiểu luận còn nhiều hạn chế và sai sót. Rất mong nhận được sự đóng góp ý kiến của các thầy cô và bạn đọc để bài tiểu luận được hoàn thiện hơn.

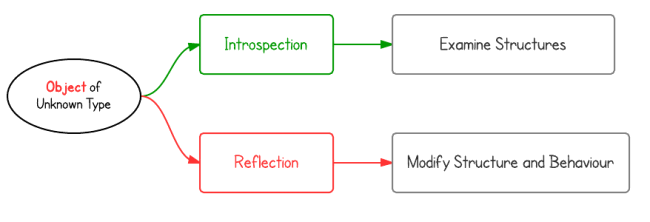
# **Đề Tài: *Hướng Dẫn Và Ví Dụ Java Reflection.***

1. **Khái Niệm:**

* **Java**sử dụng từ **"Java Reflection"**để đặt tên cho một **API** quan trọng trong thư viện chuẩn của **Java**. Tại sao **API**này lại được đặt tên như vậy? Chúng hãy cùng phân tích ý nghĩa của việc này.
* **Reflection**chính là một hình ảnh phản chiếu của một vật thể. Chẳng hạn hình ảnh của bạn trong một tấm gương, hoặc ảnh phản xạ của một cái cây trên mặt hồ. Từ **"Java Reflection"** đơn giản là đang ám chỉ một hình ảnh khác, một cách tiếp cận khác của **Java**.



* **Java**là một ngôn ngữ hướng đối tượng (Object-oriented), thông thường bạn cần tạo ra một đối tượng và bạn có thể truy cập vào các trường (field), hoặc gọi phương thức (method) của đối tượng này thông qua toán tử dấu chấm ( . )
* **Java Reflection** giới thiệu một cách tiếp cận khác, bạn có thể truy cập vào một trường của một đối tượng nếu bạn biết tên của trường đó. Hoặc bạn có thể gọi một phương thức của đối tượng nếu bạn biết tên phương thức, các kiểu tham số của phương thức, và các giá trị tham số để truyền vào ...  
    
   
* **Java Reflecion** cho phép bạn đánh giá, sửa đổi cấu trúc và hành vi của một đối tượng tại thời gian chạy (runtime) của chương trình. Đồng thời nó cho phép bạn truy cập vào các thành viên **private** (private member) tại mọi nơi trong ứng dụng, điều này không được phép với cách tiếp cận truyền thống.



* **Java**thông thường có thể được gọi là **Java Introspection**(Nội quan), chương trình có khả năng đánh giá cấu trúc của một đối tượng tại thời gian chạy (Runtime).
* Với **Java Reflection**, chương trình có khả năng đánh giá cấu trúc của một đối tượng tại thời gian chạy, sửa đổi cấu trúc và hành vi của đối tượng.

1. **Cú Pháp:**

Reflection API của Java chủ yếu được sử dụng trong:

* IDE như Eclipse, MyEclipse, NetBean,....
* Trình gỡ lỗi
* Công cụ kiểm tra

Lớp java.lang.Class

Lớp  java.lang.Class thực hiện hai nhiệm vụ như sau:

* Nó cung cấp các phương thức để lấy các siêu dữ liệu của một lớp tại thời gian chạy
* Nó cung cấp các phương thức để kiểm tra và thay đổi hành vi của lớp tại thời gian chạy

Graphical user interface, text, email

Description automatically generated

**Cách lấy đối tượng của lớp Class của Reflection trong Java**

Có ba cách lấy đối tượng của lớp Class như sau:

*Phương thức của lớp Class forName()*

Phương thức forName() tải lớp động hoặc trong thời gian chạy. Phương thức này trả về biến của lớp Class. Chúng ta chỉ nên sử dụng phương thức này nếu chúng ta biết tên đạt tiêu chuẩn của lớp đó. Chúng ta không thể sử dụng tên này cho các loại nguyên thủy.

**Phương thức của lớp Object getClass()**

Phương thức getClass() thuộc về lớp Object và trả về biến trong lớp Class. Chúng ta nên sử dụng nó khi chúng ta biết kiểu của nó. Chúng ta cũng có thể sử dụng nó với các biến nguyên thủy.

**Cú pháp .class**

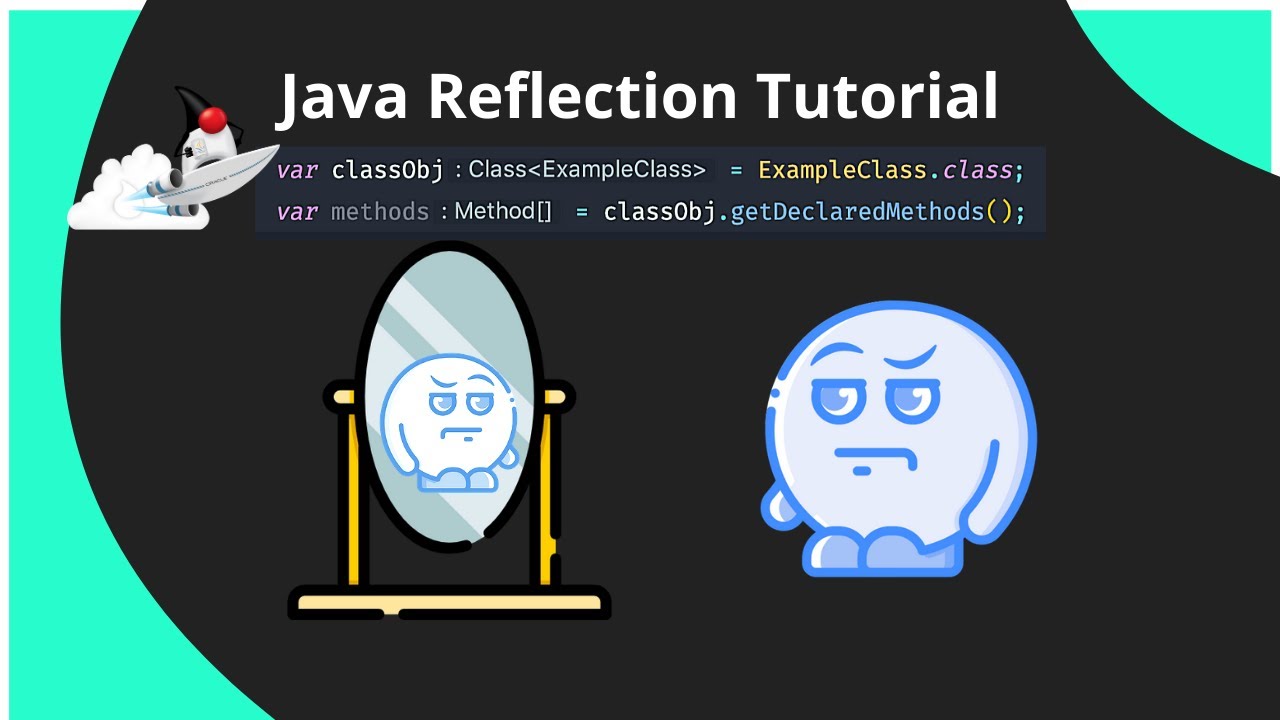
Đôi khi, có tình huống khi một kiểu có sẵn nhưng không có thể hiện của lớp. Trong những trường hợp như thế, chúng ta có thể lấy lớp bằng cách thêm cú pháp .class vào tên của kiểu. Chúng ta cũng có thể sử dụng các cú pháp này với các nguyên mẫu.

## **Lấy thông tin bằng cách sử dụng Reflection API trong Java**

Chúng ta có thể sử dụng Reflection để lấy các thông tin về:

* Class: Phương thức getClass() cho biết tên của lớp mà đối tượng thuộc về
* Constructors: Phương thức getConstructors() trả về tất cả các hàm tạo công hai của lớp mà đối tượng thuộc về
* Methods: Phương thức getMethods () đưa ra tất cả các phương thức chung của lớp mà một đối tượng thuộc về.

1. **Ý Nghĩa:**



* **java Reflection còn những hạn chế:**
* **Hiệu năng thấp:**

Ví dụ phải quét classpath để tìm class.

* **Các vấn đề bảo mật:**

Việc chỉnh sửa class/object trong quá trình runtime có thể ảnh hưởng tới các thread … khiến cho ứng dụng bị fail.

* **Khó bảo trì:**

Việc Reflection khá khó hiểu với người mới và không dễ để debug, nên sẽ rất khó để có thể tìm ra lỗi. Ngoài ra chúng ta cũng không thể check được một số lỗi trong quá trình compile.

* **Một số điểm lưu ý về java reflection:**

+Các lớp cần thiết để phản chiếu trong Java có trong gói java.lang.reflect.

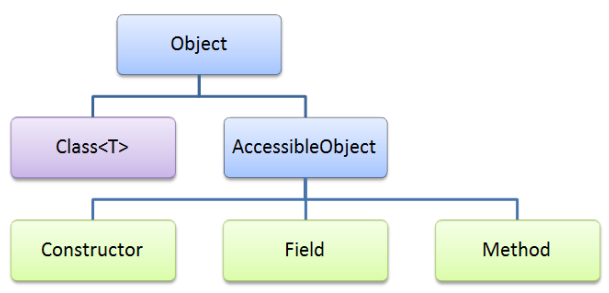
+Reflection cung cấp cho chúng ta các dữ liệu về lớp với các đối tượng liên kết và phương thức cho lớp đó.

+Thông qua sự phản chiếu, chúng ta có thể gọi một phương thức tại thời gian chạy độc lập với trình xác định truy cập của chúng.

1. Cấu trúc của 1 Class trong Java Reflection:

Các lớp được dùng trong reflection nằm trong hai package là java.lang và java.lang.reflect. Package java.lang.reflect bao gồm ba lớp chính mà bạn cần biết là Constructor, Field và Method:

* **Class<T>**: lớp này đại diện cho các lớp, interface và chứa các phương thức dùng để lấy các đối tượng kiểu Constructor, Field, Method,…
* **AccessibleObject**: các kiểm tra về phạm vi truy xuất (public, private, protected) của field, method, constructor sẽ được bỏ qua. Nhờ đó bạn có thể dùng reflection để thay đổi, thực thi các thành phần này mà không cần quan tâm đến phạm vi truy xuất của nó.
* **Constructor**: chứa các thông tin về một constructor của lớp.
* **Field**: chứa các thông tin về một field của lớp, interface.
* **Method**: chứa các thông tin về một phương thức của lớp, interface.



### Lớp (Classes):

Khi sử dụng Java Reflection để duyệt qua một class thì việc đầu tiên thường phải làm đó là có được một đối tượng kiểu Class, từ các đối tượng kiểu Class chúng ta có thể lấy được các thông tin về:

* Class Name
* Class Modifies (public, private, synchronized etc.)
* Package Info
* Superclass
* Implemented Interfaces
* Constructors
* Methods
* Fields
* Annotations

### Tạo đối tượng Class<>:

Đối tượng kiểu **Class** được tạo ra bằng cách sử dụng phương thức static **Class.forName()**. Cách này thường được sử dụng khi chỉ biên được tên lớp lúc thực thi (runtime):

Trong trường hợp không tìm thấy lớp tương ứng, phương thức trên sẽ ném ra ngoại lệ **ClassNotFoundException**. Điều này có thể bất tiện vì bạn phải sử dụng try catch hoặc ném ngoại lệ này khỏi phương thức.

Khi bạn biết chính xác tên Lớp tại thời điểm biên dịch (combine), có thể sử dụng TenLop.class để tạo đối tượng kiểu Class. Cách này đảm bảo rằng lớp được sử dụng luôn luôn tồn tại và không có ngoại lệ nào xảy ra. Đối với các kiểu dữ liệu nguyên thủy như void, int, boolean, char,… bạn có thể dùng field **TYPE** để lấy được đối tượng Class tương ứng.

Nếu bạn có một đối tượng, bạng cũng có thể lấy được đối tượng Class.

 ví dụ đơn giản dùng reflection để in ra các thông tin của lớp:

### Cấu tử (Constructor):

#### Lấy tất cả Constructor của một Class:

Các đối tượng lớp Contructor là những phuơng thức khởi tạo của một lớp. Reflection cho phép lấy ra những Contructor từ Class Object:

#### Lấy một Constructor cụ thể:

Nếu như bạn biết chính xác các kiểu parameter của constructor mà bạn muốn access đến thì bạn có thể lấy về đối tượng Constructor mà mình mong muốn thay vì phải lấy tất cả (một mảng).

**Lưu ý:** Khi bạn thực hiện lấy 1 đối tượng Constructor của một lớp bất kỳ, nhưng Constructor không tồn tại thì nó sẽ quăng ra **NoSuchMethodException**.

#### Lấy danh sách tham số của một Constructor

Bạn cũng có thể truy cập được đến tham số của các Contructor, các tham số này đều được đua về kiểu Class.

#### Lấy danh sách tham số của một Constructor

Bạn cũng có thể truy cập được đến tham số của các Contructor, các tham số này đều được đua về kiểu Class.

### Trường (Field):

#### Lấy các đối tượng field được khai báo là public

Bạn có thể lấy được đối tượng field được khai báo là **public** của một Class bằng 2 cách là chỉ lấy một field duy nhất nếu bạn biết chính xác tên của 1 field, hoặc lấy nguyên 1 mảng danh sách các field của từ một đối tượng Class.

#### Lấy các đối tượng field khai báo bất kỳ

Phương thức **getField()**và **getFields()** chỉ có thể lấy các field được khai báo là **public**. Vậy làm sao để access được những field được khai báo là private, protected…? Khá đơn giản, trong Java bạn có thể lấy được chúng thông qua 2 methods là **getDeclaredField()**và **getDeclaredFields()**.

**Lưu ý:** Các Exception có thể được ném ra khi gọi phương thức getDeclaredField():

* **NoSuchFieldException**: Nếu tên field bạn nhập vào không tìm thấy trong Class Cat.
* **NullPointerException**: Nếu tham số truyền vào là null.
* **SecurityException**: Nếu có Security manager được áp dụng ở lớp này

#### Lấy tên field, kiểu dữ liệu kiểu field:

Sau khi bạn đã có được đối tượng **Field** hoặc mảng **Field[]**bạn muốn biết tên của Field chỉ cần gọi phương thức **getName()** để lấy tên field, hoặc **getType()** để lấy kiểu dữ liệu của field.

#### Gán giá trị cho Field:

Gọi phương thức **Field.set(Object obj, Object value)**: gán value cho field tương ứng của đối tượng obj.

### Phương thức (method):

Như đã nói ở trước đó, khi bạn sử dụng java reflection bạn có thể truy cập vào các phuơng thức của một lớp và gọi những phuơng thức này.

#### Lấy tất cả Method của một Class

Phương thức **Class.getMethods()**: trả về danh sách đối tượng Method của một lớp.

1. *Lấy một Method cụ thể:*

Phương thức **Class.getMethod(String name, Class[] parameterTypes):** trả về đối tượng Method đại diện cho một phương thức của lớp. Phương thức này được xác định qua tên và các kiểu tham số.

#### Thực thi một Method:

Phương thức **Method.invoke(Object obj, Object[] args)** thực thi phương thức tương ứng của đối tượng obj với các tham số args.

1. *Truy cập vào các private method, field:*

Bạn không thể truy cập vào các method hay field mà nó là private theo cách thông thường, quá trình biên dịch java cũng không cho phép điều đó. Nhưng với Java Reflection điều đó hoàn toàn có thể

***LỜI CÁM ƠN***

Trước tiên, tôi xin bày tỏ lòng biết ơn sâu sắc đến thầy ThS. LÊ HUỲNH PHƯỚC, người đã truyền cho tôi niềm đam mê khoa học, người đã luôn tận tình bảo ban và đồng hành cùng tôi trong suốt quá trình học tập và nghiên cứu.

Tôi xin gửi lời cám ơn đến quý thầy cô khoa Công Nghệ Thông Tin, đã dạy bảo và truyền đạt kiến thức cho tôi trong suốt quá trình học tập.

Cuối cùng, tôi xin chân thành cám ơn Ban giám hiệu Trường Đại Học Gia Định đã luôn tạo điều kiện cho tôi trong suốt học kỳ vừa qua.

|  |  |
| --- | --- |
|  | *Tp.Hồ Chí Minh, ngày 27 tháng 07 năm 2022.*  Sinh viên thực hiện |
|  |  |