Bài 2
Lớp và Đối tượng

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài “Ngôn ngữ lập trình Java"

Tóm tắt lại các phần đã học từ bài “Ngôn ngữ lập trình Java”

Mục tiêu

- Trình bày được mô hình lập trình hướng đối tượng
- Trình bày được các khái niệm lớp, đối tượng, phương thức, thuộc tính, hàm tạo
- Trình bày được cú pháp khái báo lớp
- Trình bày được cú pháp khởi tạo đối tượng
- Trình bày được cách truy xuất thuộc tính, phương thức của lớp
- Tạo và sử dụng được các đối tượng đơn giản
- Mô tả được lớp bằng biểu đồ

Thảo luận

Đối tượng

Lớp

Khai báo lớp

- **Lớp** là đơn vị thực thi cơ bản trong ngôn ngữ Java
- Lớp quy định hình thức và các khả năng của các đối tượng
- Khai báo lớp đồng thời cũng là khai báo một kiểu dữ liệu mới để có thể khởi tạo các đối tượng thuộc kiểu dữ liệu đó

Cú pháp khai báo lớp

- Cú pháp:

```
class <class_name> {
    // class body
}
```

  Trong đó:

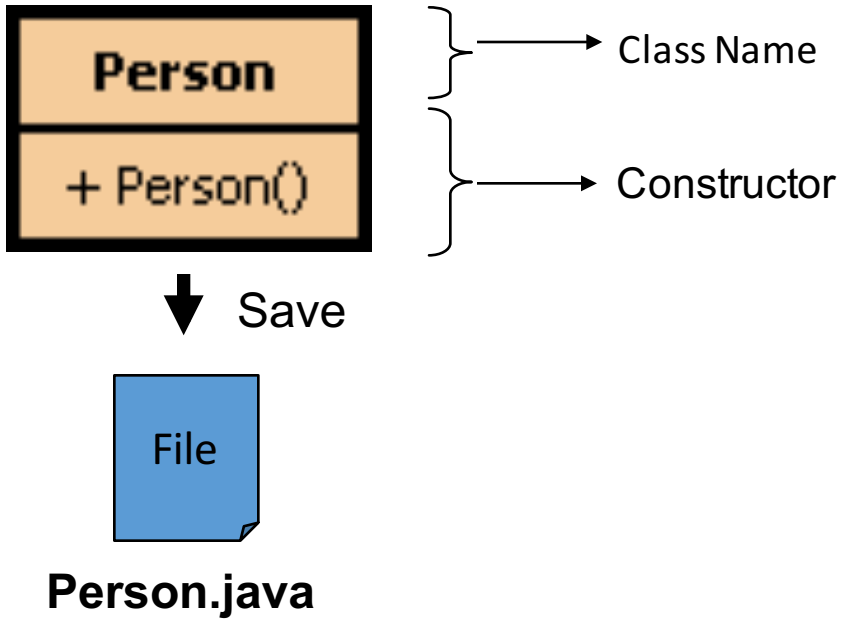
  - *class* là từ khoá được dùng để khai báo biến
  - *class_name* là tên của lớp
  - *class body* là phần thân của lớp: nơi khai báo các thành phần của lớp như các trường (field), các phương thức (method) và các phương thức khởi tạo (constructor)

- Constructor – phương thức khởi tạo: là một phương thức đặc biệt được sử dụng để khởi tạo các đối tượng của một lớp

Đặt tên lớp

- Một số quy ước khi đặt tên lớp:
  - Tên lớp nên là một danh từ
  - Tên lớp nên tuân theo quy tắc Camel
  - Tên lớp nên đơn giản, có nghĩa
  - Tên lớp không thể trùng với các từ khoá trong Java
  - Tên lớp không được bắt đầu bằng chữ số. Có thể bắt đầu bằng ký tự dollar ($) hoặc dấu gạch dưới (_)

Khai báo lớp: Ví dụ




```
/**
 * Write a description of class Person
   here.
 *
 * @author CodeGym
 * @version 1/1/2018
 *
 */
public class Person {
    /**
     * Hàm tạo đối tượng của lớp
     */
    public Person() {
        // Khởi tạo đối tượng:
    }
}
```

Khởi tạo đối tượng

- Có thể khởi tạo đối tượng của một lớp sau khi lớp đó được khai báo
- Sử dụng từ khoá *new* để khởi tạo đối tượng
- Cú pháp:

```
<class_name> <object_name> = new <class_name> ();
```

Trong đó:

- class_name là tên của lớp
- *new* là từ khoá để khởi tạo đối tượng
- *object_name* là tên biến chứa tham chiếu trỏ đến đối tượng

Khởi tạo đối tượng: Ví dụ

- Ví dụ:

```
Person personObj = new Person();
```

Trong đó:

  - Biểu thức new Person() ở phía bên phải cấp phát bộ nhớ tại thời điểm thực thi
  - Sau khi vùng nhớ đã được cấp phát thì một tham chiếu đến vùng nhớ đó được trả về và gán cho biến personObj

- Có thể tách rời việc khai báo biến và khởi tạo đối tượng, ví dụ:

```
Person personObj;
personObj = new Person();
```

Demo

Tạo lớp và đối tượng

Quá trình khởi tạo đối tượng - 1

- Bước 1: Khai báo biến:

```
Person personObj;
```

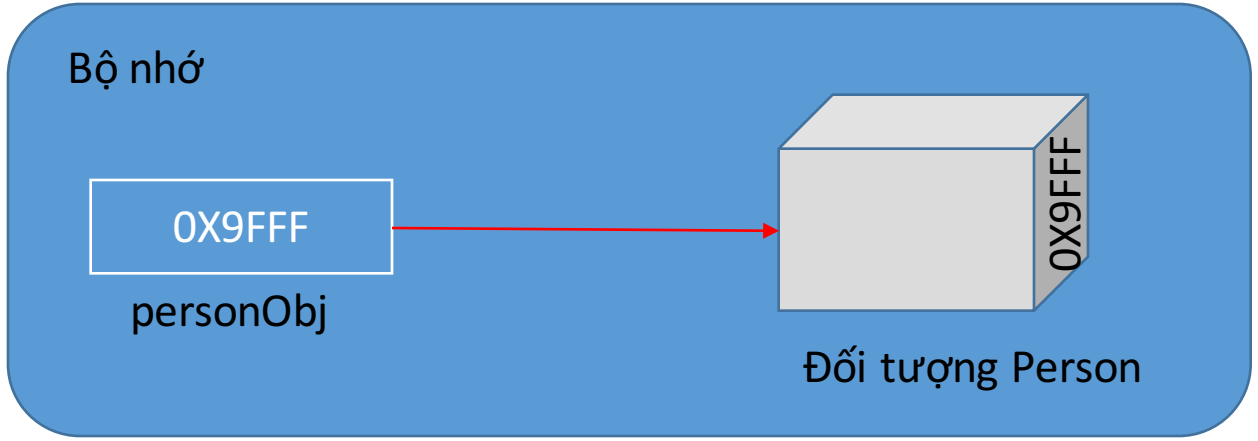
  - Biến personObj được khai báo và không trỏ đến bất kỳ đối tượng nào
  - Biến personObj có giá trị null
  - Nếu sử dụng biến personObj để truy cập các phương thức hoặc thuộc tính của lớp Person tại thời điểm này, sẽ có lỗi xảy ra

Quá trình khởi tạo đối tượng - 2

- Bước 2: Khởi tạo đối tượng:

```
personObj = new Person();
```

  - Một đối tượng của lớp Person được khởi tạo và lưu vào một vùng nhớ (chẳng hạn có địa chỉ là 0X9FFF)
  - Địa chỉ vùng nhớ được gán cho biến personObj

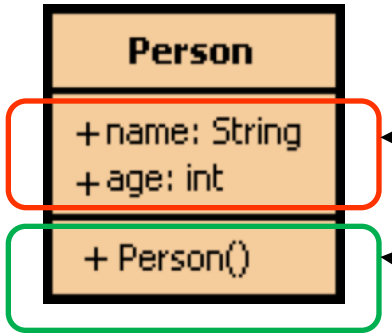


Khai báo thuộc tính

- Các thuộc tính mô tả các đặc điểm của đối tượng
- Thuộc tính còn được gọi là instance variable (biến của đối tượng)
- Cú pháp:

```
access_modifier  data_type  property_name
```

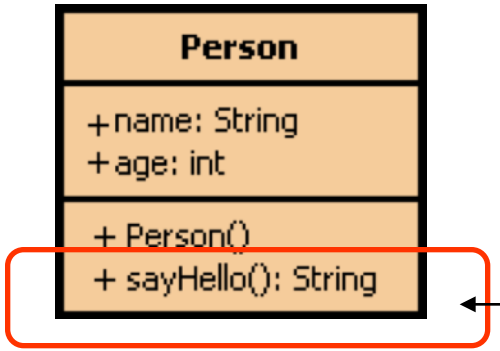
- Ví dụ:




```
public class Person {
    /**
     * Fields
     */
    public String name;
    public int age;
    /**
     * Constructors
     */
    public Person()  {
    }
}
```

Khai báo phương thức

- Phương thức mô tả các hành vi mà đối tượng có thể thực hiện
- Phương thức còn được gọi là instance method (phương thức của đối tượng)
- Ví dụ:




```
public class Person {
    /**
     * Fields
     */
    public String name;
    public int age;
    /**
     * Constructors
     */
    public Person() {
    }
    /**
     * Methods
     */
    public String sayHello() {
        return "Hello World";
    }
}
```

Constructor

- Constructor là một phương thức đặc biệt giúp khởi tạo đối tượng
- Constructor có tên trùng với tên của lớp
- Một lớp có thể có nhiều constructor
- Nếu không khai báo constructor cho lớp thì mặc định lớp đó có một constructor không có tham số
- Ví dụ:

```
public Person() {
    name = "No name";
    age = 10;
}
```

```
public Person(String s, int n) {
    name = s;
    age = n;
}
```

Sử dụng constructor

- Có thể lựa chọn sử dụng các constructor khác nhau bằng cách truyền vào tham số khác nhau
- Ví dụ:

```
public Person() {
      name = "No name";
      age = 10;
}
```

→ personObj = **new** Person();

```
public Person(String s, int n) {
      name = s;
      age = n;
}
```

→ personObj = **new** Person("John", 20);

Demo

Tạo lớp với các loại constructor

Tạo đối tượng

Thảo luận

Truy xuất các thuộc tính

Gọi các phương thức

Truy xuất thuộc tính của đối tượng

- Có thể truy xuất các thành phần của đối tượng thông qua biến trỏ đến đối tượng
- Sử dụng dấu chấm (.) để truy xuất thuộc tính của đối tượng
- Ví dụ:

```
Person personObj;
personObj = new Person("John", 20);

System.out.println("My name is: " + personObj.name);
System.out.println("My age is: " + personObj.age);
```

Lưu ý: Quyền truy xuất đến các thành phần của đối tượng được quy định bởi access modifier (public/private/protected/default), sẽ được đề cập đến sau.

Gọi phương thức

- Sử dụng dấu chấm (.) để gọi phương thức của đối tượng
- Ví dụ:

```
Person personObj;
personObj = new Person();
String greeting = personObj.sayHello();
```

Thảo luận

Getter và Setter

Truy cập trực tiếp vào các trường dữ liệu

- Sử dụng từ khoá public khi khai báo thuộc tính sẽ cho phép truy cập trực tiếp vào các thuộc tính đó
- Ví dụ:

  Khai báo lớp Person sau cho phép truy cập trực tiếp vào trường name

```
class Person{
    public String name;
}
```

```
Person person = new Person();
person.name = "John";
```

- Nhược điểm:
  - Không kiểm soát được truy cập vào thuộc tính
  - Gây khó khăn cho việc duy trì, dễ phát sinh bug

Data field encapsulation

- Data field encapsulation (bao gói trường dữ liệu) là hình thức hạn chế quyền truy cập trực tiếp vào các thuộc tính của đối tượng bằng cách sử dụng từ khoá private
- Khai báo các phương thức để kiểm soát việc truy cập vào các thuộc tính của đối tượng
- Các phương thức cho phép thay đổi giá trị của thuộc tính được gọi là setter, các phương thức cho phép lấy về giá trị của thuộc tính được gọi là getter
- Ví dụ getter: getName(), getAge(), getDate(), isAvailable()…
- Ví dụ setter: setName(), setAge(), setAddress()…

Khai báo getter/setter

- Cú pháp khai báo getter:

```
public returnType getPropertyName()
```

  - Đối với các thuộc tính kiểu *boolean* thì tên getter bắt đầu bằng chữ *is*:

```
public boolean isPropertyName()
```

- Cú pháp khai báo setter:

```
public void setPropertyName(dataType propertyValue)
```

Getter/setter: Ví dụ

```
class Person{
    private String name;

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

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

```
public static void main(String[] args) {
    Person person = new Person();
    person.setName("John");
    System.out.println("My name is: " + person.getName());
}
```

Từ khoá this

- Từ khoá *this* được sử dụng để đại diện cho đối tượng hiện tại
- Có thể sử dụng từ khoá this để truy cập đến các thành phần của đối tượng hiện tại
- Ví dụ, sử dụng từ khoá *this* để phân biệt 2 biến có cùng tên:

```
class Person{
    private String name;

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

Biến name của lớp Person

Tham số name được truyền vào

Demo

Sử dụng getter/setter

Sử dụng this

Tóm tắt bài học

- Lập trình hướng đối tượng (OOP) là mô hình lập trình phổ biến hiện nay
- OOP mô phỏng các đối tượng trong thế giới thực vào trong thế giới lập trình
- Từ khoá *class* được sử dụng để khai báo lớp
- Từ khoá *new* được sử dụng để khởi tạo đối tượng
- Thuộc tính mô tả các đặc điểm của đối tượng
- Phương thức mô tả các hành vi của đối tượng
- Phương thức khởi tạo (constructor) là phương thức giúp khởi tạo các đối tượng
- Nếu không khai báo constructor thì mặc định các lớp đều có một constructor không tham số
- Có thể mô tả lớp bằng các ký hiệu UML
- Có thể truy xuất các thành phần của lớp thông qua dấu chấm (.)

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: Access modifier, static method, static property