Bài 3
Access modifier, static method, static property

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài “Lớp và đối tượng"

Tóm tắt lại các phần đã học từ bài “Lớp và đối tượng”

Mục tiêu

- Sử dụng được access modifier
- Sử dụng được static method
- Trình bày được cơ chế kế thừa
- Triển khai được cơ chế kế thừa giữa các lớp
- Trình bày được cơ chế ghi đè phương thức (method overriding)
- Biểu diễn được mối quan hệ kế thừa bằng các ký hiệu
- Trình bày được ý nghĩa của từ khoá final
- Trình bày được khái niệm Polymophism
- Trình bày được phương thức toString() của lớp Object
- Trình bày được cơ chế ép kiểu (casting)

Thảo luận

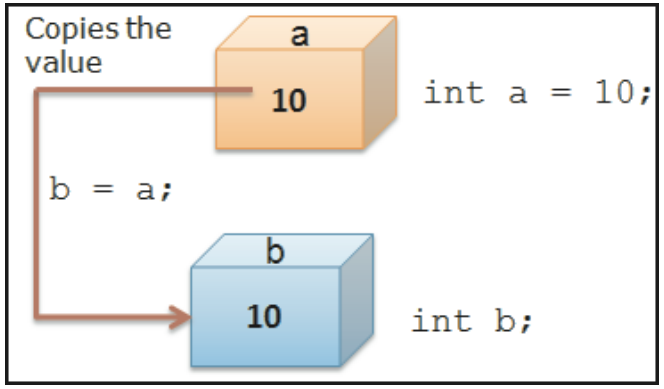
Biến kiểu dữ liệu nguyên thuỷ

Biến tham chiếu

Primitive data types

- Các biến thuộc kiểu dữ liệu nguyên thuỷ (như int, long, float…) lưu trữ **giá trị** của chúng trong vùng nhớ được cấp
- Giá trị của một biến có thể được gán cho một biến khác
- Ví dụ:

```
int a = 10;
int b = a;
```



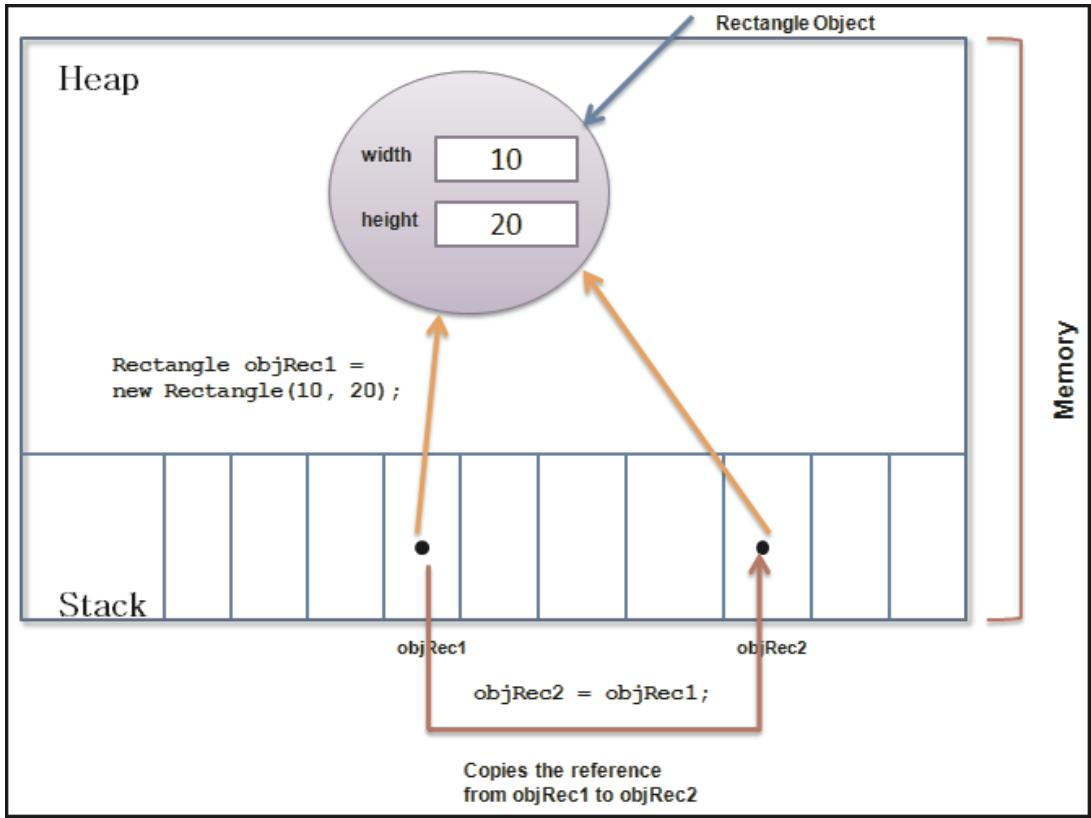
  - Thao tác này sao chép giá trị của biến a (được lưu trong vùng nhớ được cấp cho a) cho biến b (lưu vào vùng nhớ được cấp cho b)

Reference data types

- Các biến thuộc kiểu dữ liệu tham chiếu (chẳng hạn như Scanner, Person, Customer…) lưu trữ **tham chiếu** của đối tượng ở trong vùng nhớ được cấp
- Có thể gán giá trị tham chiếu của một biến cho một biến khác
- Ví dụ:

```
Rectangle rectangleObj1 = new Rectangle(10, 20);
Rectangle rectangleObj2 = rectangleObj1;
```

  - Thao tác này sao chép **địa chỉ** được lưu trong biến rectangleObj1 sang biến rectangleObj2
  - Không có ảnh hưởng nào xảy ra đối với đối tượng thực tế trong bộ nhớ



Primitive data type: Ví dụ

```
public static void swap(int first, int second){
    int temp = first;
    first = second;
    second = temp;
}

public static void main(String[] args) {
    int a = 5;
    int b = 10;

    swap(a, b);

    System.out.println("a = " + a);
    System.out.println("b = " + b);
}
```

Kết quả:

```
a = 5
b = 10
```

Reference data type: Ví dụ

```
class Person{
    public String name;

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

```
public static void swap(Person first, Person second){
    String temp = first.name;
    first.name = second.name;
    second.name = temp;
}

public static void main(String[] args) {
    Person a = new Person("John");
    Person b = new Person("Bill");

    swap(a, b);

    System.out.println("a.name = " + a.name);
    System.out.println("b.name = " + b.name);
}
```

Kết quả:

```
a.name = Bill
b.name = John
```

Demo

Biến kiểu dữ liệu nguyên thuỷ

Biến tham chiếu

Thảo luận

Access modifier

Access modifier

- Access modifier là các từ khoá được sử dụng để quy định mức độ truy cập đến lớp và các thành phần của lớp
- Các mức truy cập:
  - *public*: có thể truy cập từ bất cứ đâu
  - *private*: các phương thức và thuộc tính chỉ được phép truy xuất trong cùng một lớp
  - *protected*: các phương thức và thuộc tính được phép truy xuất trong cùng một lớp và ở các lớp con (kế thừa)
  - *default*: Nếu không có access modifier thì mức default sẽ được áp dụng. Lớp và các thành phần của lớp được truy xuất ở những nơi trong cùng một package

Access modifier: Ví dụ

```java
package p1;

public class C1 {
  public int x;
  int y;
  private int z;

  public void m1() {
  }
  void m2() {
  }
  private void m3() {
  }
}
```

```java
package p1;

public class C2 {
  void aMethod() {
    C1 o = new C1();
    can access o.x;
    can access o.y;
    cannot access o.z;

    can invoke o.m1();
    can invoke o.m2();
    cannot invoke o.m3();
  }
}
```

```java
package p2;

public class C3 {
  void aMethod() {
    C1 o = new C1();
    can access o.x;
    cannot access o.y;
    cannot access o.z;

    can invoke o.m1();
    cannot invoke o.m2();
    cannot invoke o.m3();
  }
}
```

Tổng hợp các mức truy cập

Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

Demo

Access modifier

Thảo luận

Package

Package

- Package (gói) là cách để phân loại các lớp và interface thành các nhóm có liên quan đến nhau và tổ chức chúng thành các đơn vị để quản lý
- Ví dụ, Java cung cấp sẵn các gói:
  - *java.io*: Thực hiện các thao tác nhập xuất dữ liệu
  - *java.net*: Thực hiện các thao tác qua mạng lưới
  - *java.security*: Thực hiện các thao tác liên quan đến bảo mật
  - *java.util*: Cung cấp các lớp và phương thức hỗ trợ
  - …
- Lập trình viên có thể tự định nghĩa các gói mới để tổ chức mã nguồn hợp lý

Tính chất của package

- Có thể khai báo các gói con - subpackage (gói ở bên trong gói)
- Không thể có 2 lớp có cùng tên trong cùng 1 gói
- Khi một lớp được khai báo bên trong một gói thì cần phải sử dụng tên của gói nếu muốn truy cập đến lớp đó
- Tên của gói được viết bằng chữ thường
- Các gói được cung cấp sẵn của Java được bắt đầu bằng từ *java* hoặc *javax*

Khai báo package

- Cú pháp:

```
package package_name;
```

- Ví dụ:

```
package codegym;
```

- Tên của package phải trùng với tên của thư mục chứa mã nguồn
- Tên của subpackage phải lần lượt trùng với tên của các thư mục tương ứng, ví dụ:

```
package com.codegym.ui;
```

Từ khoá import

- Cần sử dụng từ khoá import để có thể sử dụng các lớp được định nghĩa trong các package khác
- Ví dụ:

```
package model;

public class Customer {
}
```

```
package controller;

import model.Customer;

public class CustomerController {
    public void index(){
        Customer customer = new Customer();
    }
}
```

Demo

Package

Thảo luận

Static property

Static method

Từ khoá static

- Từ khoá *static* được sử dụng để khai báo các thuộc tính và phương thức của lớp (khác với thuộc tính và phương thức của đối tượng)
- Các thành phần static trực thuộc lớp, thay vì trực thuộc đối tượng
- Biến static còn được gọi là biến của lớp (class variable)
- Phương thức static còn được gọi là phương thức của lớp (class method)
- Có thể truy xuất các thành phần static bằng cách sử dụng lớp hoặc đối tượng
- Không cần khởi tạo đối tượng vẫn có thể sử dụng các thành phần static

Static property

- Cú pháp khai báo *static property*:

```
modifier static data_type variable_name;
```

- Ví dụ:

  Khai báo biến static:

```
class Application{
    public static String language = "english";
}
```

  Truy xuất biến static:

```
System.out.println("Current language: " + Application.language);
```

Static method

- Cú pháp khai báo static method:

```
modifier static data_type method_name(){
    //body
}
```

- Ví dụ:
  - Khai báo phương thức static

```
class Application{
    public static String getVersion(){
        return "1.0";
    }
}
```

  - Gọi phương thức static

```
System.out.println("Current version: " + Application.getVersion());
```

Một số ràng buộc

- Phương thức static chỉ có thể gọi các phương thức static khác
- Phương thức static chỉ có thể truy xuất các biến static
- Phương thức static không thể sử dụng từ khoá *this* hoặc *super*
- Có thể khởi tạo biến static thông qua khối khởi tạo static
- Ví dụ:

```
class Application{
    public static String language;

    static {
        if(System.getProperty("lang").equals("en")){
            language = "english";
        } else {
            language = "spanish";
        }
    }
}
```

Tổng kết

- Từ khoá static được sử dụng để khai báo các thành phần thuộc lớp
- package được sử dụng để nhóm các lớp có liên quan đến nhau trong cùng một đơn vị
- Getter/setter là cơ chế để kiểm soát truy cập đến các trường dữ liệu của đối tượng

Demo

Thuộc tính static

Phương thức static

Tổng kết

- Kế thừa là cơ chế cho phép một lớp thừa hưởng các đặc điểm và hành vi của một lớp khác
- Lớp được kế thừa gọi là lớp cha, lớp kế thừa gọi là lớp con
- Ghi đè phương thức là hình thức lớp con viết lại các phương thức đã có của lớp cha
- Sử dụng mũi tên rỗng để biểu diễn mối quan hệ kế thừa giữa các lớp
- Java không hỗ trợ đa kế thừa
- Từ khoá final được sử dụng để ngăn chặn việc kế thừa từ một lớp và việc ghi đè phương thức
- Đa hình là cơ chế cho phép một biến kiểu cha có thể trỏ đến các đối tượng kiểu con
- Lớp Object là lớp cha của tất cả các lớp trong Java
- Phương thức toString() được sử dụng để trả về một chuỗi mô tả đối tượng
- Ép kiểu là hình thức chuyển đổi tham chiếu đối tượng từ một kiểu này sang tham chiếu đối tượng thuộc kiểu khác

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: *Thừa kế (Inheritance)*