LightHouseIT

2.3 封装

- Java 包
- 封装的概念
- 封装的基本语法

LightHouseIT

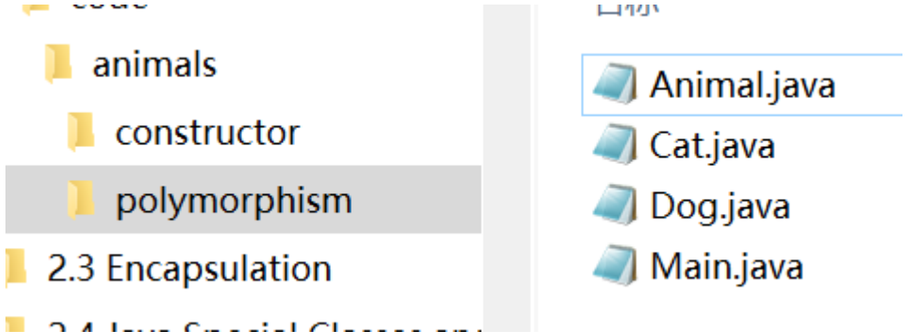
Java 中的包

- **包**[パッケージ]是 Java 中分类存储代码的基本形式。
- 包的表现和作用都类似于计算机的文件夹系统。
- 你可以把处理不同问题的代码放在不同的包内，方便对代码的查询和整理。
- 使用包还有一个好处：不同包内可以放入同名的代码文件（如 java.awt.Window 和 my.house.Window），不会产生冲突。（即提供了*命名空间*。）

包的声明

- 要声明一段代码属于哪一个包，我们必须做以下两件事：

1. 把代码放在和包名相符合的**文件夹**内：



这里，Animal.java 被放在了 animals / polymorphism 文件夹里，它对应的包就是 animals.polymorphism。

2. 在代码的**开头**用 package 关键字声明包名：

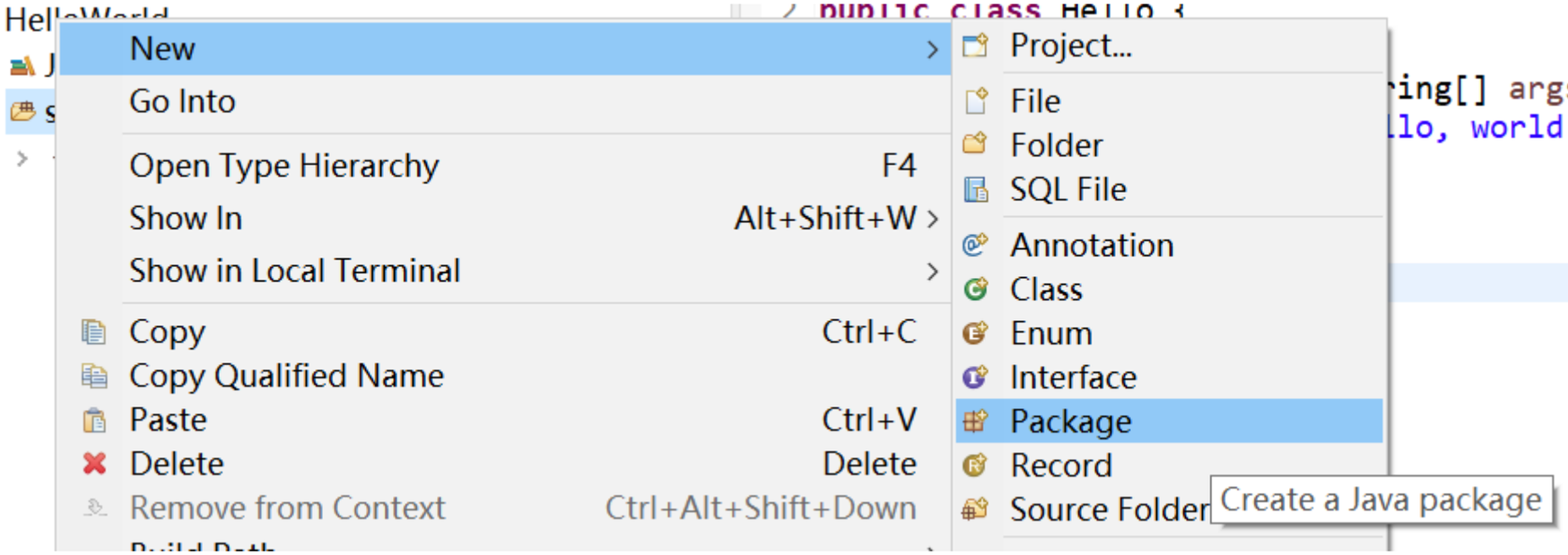
```
package animals.polymorphism;
```

接次页

接前页

- 觉得麻烦？在 Eclipse（以及绝大多数 Java IDE）中，你可以直接使用新建包的功能，新建出来的代码文件会自动添加上包的声明。

- 右键 src 文件夹或某个已有的包 → New → Package。



包的使用

- 如果要使用其他包里所定义的类，需要先从相应的包中**导入**相应的类。要导入一个类，使用 import 关键字，写上**包名 + 类名**，用“.”链接：

```
import animals.polymorphism.Cat;
```

- 也可以使用星号“*”一次性导入某个包里的所有类：

```
import animals.polymorphism.*;
```

- 如果你有不同包里名称相同的两个类，则需在使用类时书写完整路径加以区分：

```
java.awt.Window window = new java.awt.Window( );
my.house.Window houseWindow = new my.house.Window( );
```

Question and answer

Coffee Break

包的命名风格

一般来说，Java 中的每一个包名均为小写英文单词。如果一个包名有多个单词，可以使用下划线 “_” 相连接。

除了这个基本规则之外，Java 的包名还有一个广为人知的规定：一般包名的开头几个目录都会标识出开发公司 / 组织使用的*域名*（即官网网址）。并且一般标识顺序会与网址相反。比如，如果你的个人主页网址是 “home.zhangsan.com”，你写的 hello 包就可以起名为：

```
package com.zhangsan.hello;
```

在之后的课程中我们将会学习的 Spring Boot 架构，它最基本的包名是 “org.springframework.boot”。你能判断出 Spring 的官网网址吗？

LightHouseIT

封装

- 在面向对象编程中，**封装**[カプセル化]指的是将类的一部分**属性**或**功能**隐藏起来的过程。
- “隐藏起来”是指保证只有这个类（或与这个类有特定关系的对象）可以使用这些属性或功能。
- 封装的主要功能有二：
  - 避免对一些只因由对象本身修改的属性被修改，加强系统的*健壮性*。
  - 减少不必要的属性或功能的公开，增加代码的*可读性*。

接次页

接前页

Example

我们想要开发一个汽车类。想一想，下列属性和功能对于**车主**来说有哪些是**需要公开的**，哪些是**不应公开的**，哪些是**不必公开的**：

1. 汽车的高度、品牌、颜色、车厢数等；
2. 制造汽车时用到的功能，如焊接车壳；
3. 启动、停止汽车的功能；
4. 根据汽车的加速度计算出汽车速度的功能；
5. 修改汽车引擎的功能。
6. 修改汽车价格的功能。

接次页

接前页

- 可以看出，有些功能或属性是**不应该**被公开的。外部类对它们的修改可能会导致其他功能出错。比如如果用户修改了汽车引擎，计算汽车速度的功能可能无法正确运作。

- 同时，有些功能或属性是**不需要**被公开的。比如焊接车身的功能或计算车速的功能对用户没有用处，告诉用户这些功能只会让类的使用更加繁琐。

- 想一想，如果你要开发一个银行用户类，有哪些属性或功能需要公开？哪些需要隐藏？

目录

LightHouseIT

Java 中的封装

- 在 Java 中，我们通过**访问修饰符**[アクセス修飾子]实现对类的封装。访问修饰符可以指定哪些类可以访问某个成员变量或方法。

- 最基础的两个访问修饰符：public（公开）、private（私有）。

- 要为成员变量或方法添加访问权限，在定义的**开头**写上对应的修饰符：

```
1 public class User {
2     public String username;
3     private String password;
4 }
```

接次页

接前页

```
1 public int getCash() {
2     return 0;
3 }
4
5 private String getPassword() {
6     return password;
7 }
```

- 设置成 public 的成员变量和方法可以被任意其他类访问，而设置成 private 的只能在本类中被访问：

```
1 public class Main {
2     public static void main(String[] args) {
3         User user = new User();
4         user.username = "Alice";
5         user.password = "123"; // error
6         user.getCash();
7         user.getPassword(); // error
8     }
9 }
```

Try 01011 11010 01011

account 包

其它访问修饰符

- 除了 public 和 private，Java 中还有其他两种访问修饰符：

```java
public int a;
protected int b;
int c; // default 访问权限
private int d;
```

- 它们的访问权限可以总结如下：

修饰符	本类	本包	子类	外部包
public	√	√	√	√
protected	√	√	√	×
(default)	√	√	×	×
private	√	×	×	×

Try

access 包

获取器和设置器

- 先想一想这个问题：商品类的价格应不应该对顾客公开？
- 有时候，同一个属性的获取和设置需要不同的访问权限。但 Java 中，属性的访问权限同时影响它的获取和设置。
- 我们可以用**获取器**（Getter）和**设置器**（Setter）来解决这个问题。
- 想法：将属性的获取和设置写成两个独立的方法，分别设置它们的访问权限。

获取器和设置器的实现

- 首先，我们要将该属性的成员变量设成 private。接下来，我们分别为它设置 Getter 和 Setter:

```
1 private int price;
2
3 public int getPrice() {
4     return price;
5 }
6
7 private void setPrice(int price_) {
8     price = price_;
9 }
```

- 这样，其他类就可以获得它的价格，但不能设置价格了。

Try
01011
11010
01011
car 包

访问权限设置流程

- 我们可以总结访问权限的设置流程如下:

  1. 将所有成员变量和方法设置为 private。
  2. 为所有成员变量编写获取器和设置器。
  3. 根据需求调整所有方法的访问权限。
  4. 如果获取器或设置器的权限和成员变量一样，可以省略。

- 建议一开始就将访问权限设置成 private 以确保封装无误。

Question and answer

总结

Sum Up

1. Java 包的概念和语法：package 和 import。
2. 封装的概念。
3. Java 封装的语法：
   ① 访问修饰符：public protected default private。
   ② 获取器和设置器。

LightHouseIT

THANK YOU!