Workshop
Minecraft 服务端插件开发

叶子 (Alan Richard) - August 14, 2021

Session 2

编程作业1

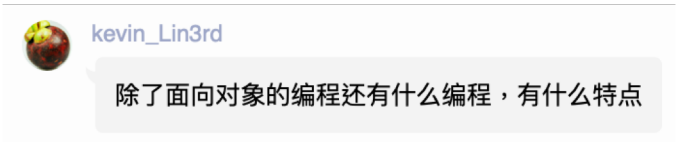
基于 lecture 中给出的演示代码完成以下功能：

原封不动地复读玩家所说的话（消息）

加分项：

复读时为原消息添加色彩

除了 OOP 还有什么编程?



Programming Paradigm

- 编程范型 / 编程范式 / 程式设计法
- 常见
  - 函数式编程
  - 指令式编程
  - 过程式编程
  - 面向对象编程

Session 2: Java 基础

复习：ABC

- camelCase
  - 变量
  - 方法（函数）名
- CamelCase
  - 类名

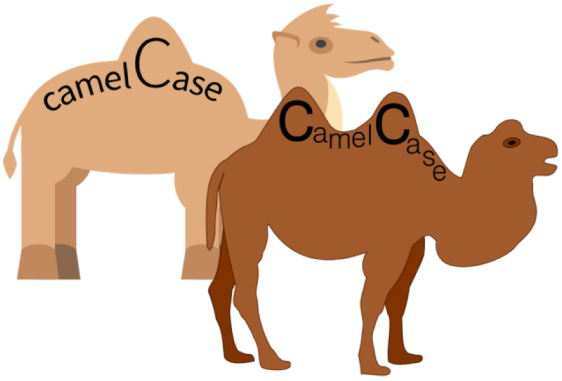


Image by Emoji One & Silver Spoon Sokpop

复习： ABC

- 缩进
  - 不像 Python 中那么严格
  - 让代码保持可读

复习： ABC

- 常见「符号」
  - 赋值： meaningOfLife = 42;
  - 加： c = a + b;
  - 减： c = a - b;
  - 乘： c = a * b;
  - 除： c = a / b;
  - 模： c = a % b; // ex: a = 11, b = 2, c = 1 (11/2 = 5 ⋯ 1)

Images from:
https://joke-battles.fandom.com/wiki/Doge_(Dogelore)
https://www.duitang.com/blog/?id=930071024

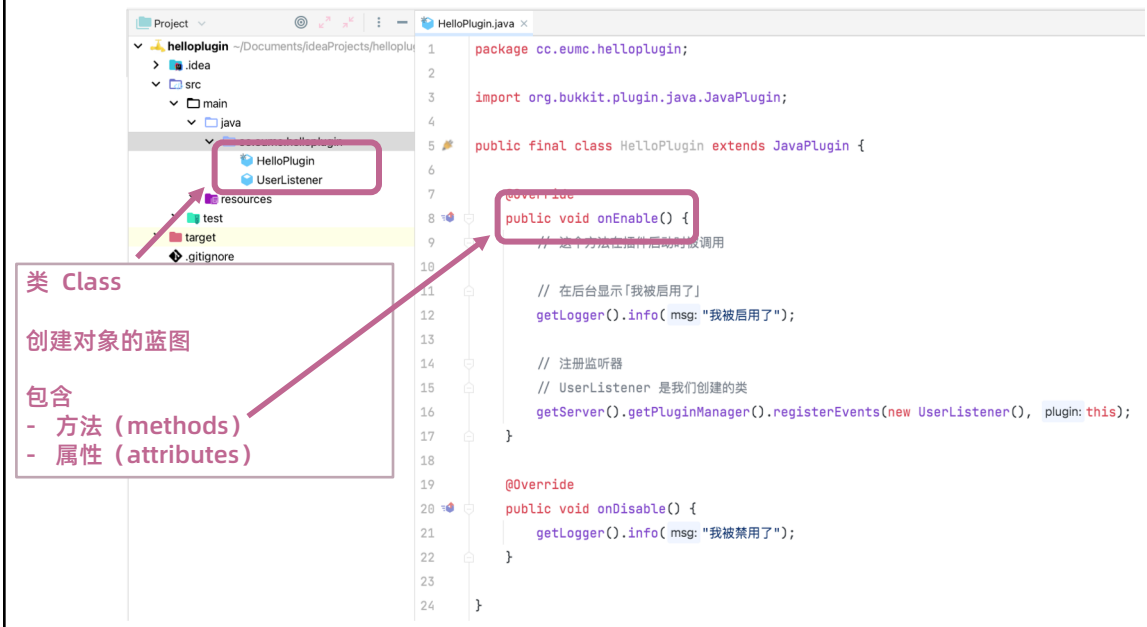
复习： ABC

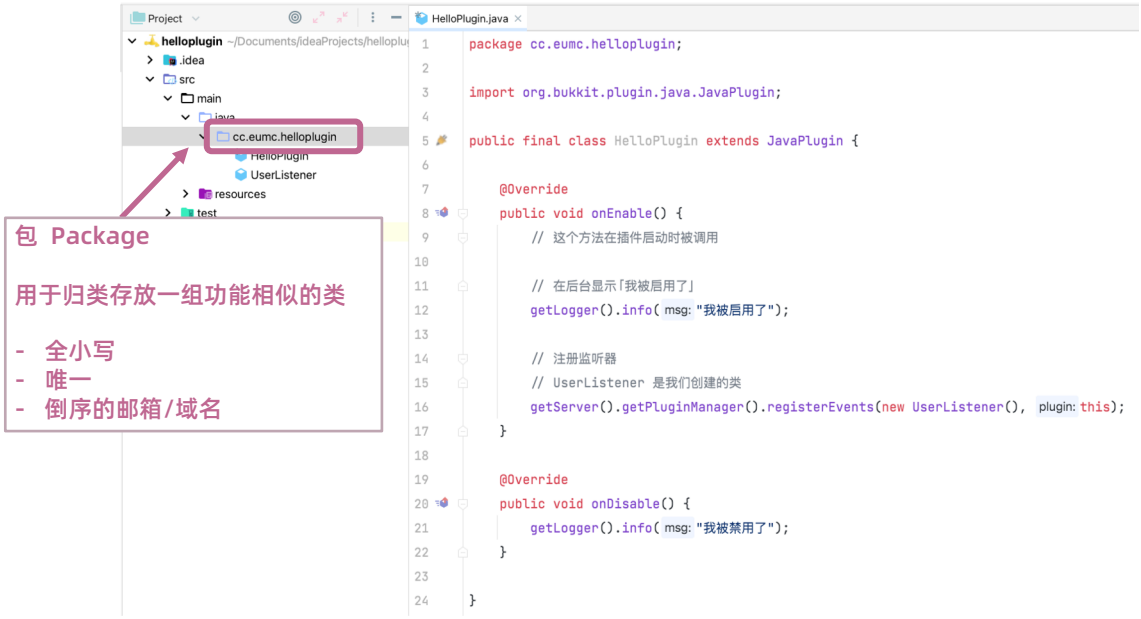
- 面向对象编程（OOP） → 「从属关系」

richard . devCourse

里查德 的 开发课

OOP 基础


Project
helloplugin ~/Documents/ideaProjects/helloplu
.idea
src
main
java
HelloPlugin
UserListener
resources
test
target
.gitignore
HelloPlugin.java
1 package cc.eumc.helloplugin;
3 import org.bukkit.plugin.java.JavaPlugin;
5 public final class HelloPlugin extends JavaPlugin {
8 public void onEnable() {
11 // 在后台显示「我被启用了」
12 getLogger().info(msg: "我被启用了");
15 // UserListener 是我们创建的类
16 getServer().getPluginManager().registerEvents(new UserListener(), plugin: this);
17 }
20 public void onDisable() {
21 getLogger().info(msg: "我被禁用了");
22 }
24 }
类 Class
创建对象的蓝图
包含
- 方法（methods）
- 属性（attributes）



包 Package

用于归类存放一组功能相似的类

- 全小写
- 唯一
- 倒序的邮箱/域名

对象

对象

- 实例化的类（Instantized classes)
  - –– 创建对象
    - Object o = new Object();

创建对象

```
Object o = new Object();
```

创建对象

```
Object o = new Object();
```

类型

只需要创建
对象时声明

创建对象

```
Object o = new Object();
```

变量名称

创建对象

```
Object o = new Object();
```

关键词

创建对象

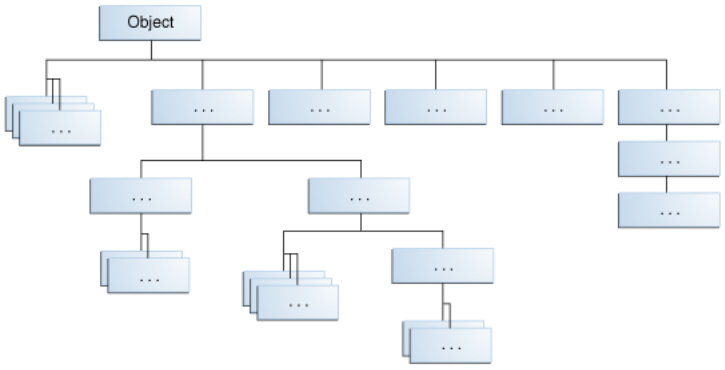
```
Object o = new Object();
```

构建方法 *

变量赋值

```
o = new Another();
o = "Hello";
```


Object 生万物

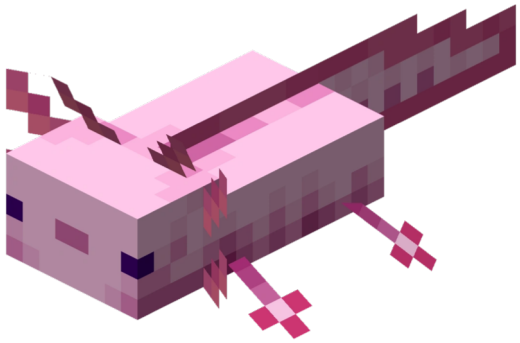


^ Ref: https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

→

← 继承

- E.g. Axolotl



^ Ref: https://minecraft.fandom.com/wiki/Axolotl

继承

Package org.bukkit.entity

Interface Axolotl

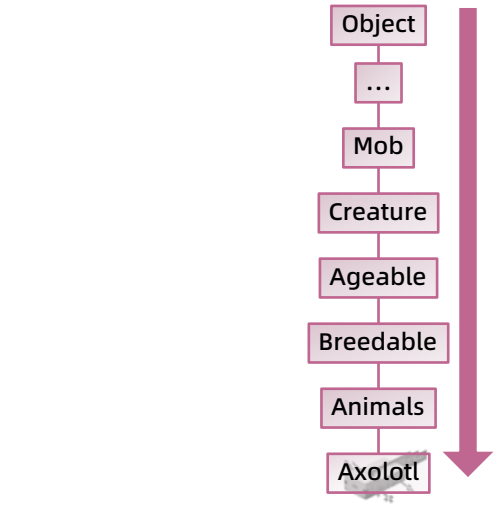
All Superinterfaces:

Ageable, Animals, Attributable, Breedable, CommandSender, Creature, Damageable, Entity, LivingEntity, Lootable, Metadatable, Mob, Nameable, Permissible, PersistentDataHolder, ProjectileSource, ServerOperator

```
public interface Axolotl
extends Animals
```

An Axolotl.

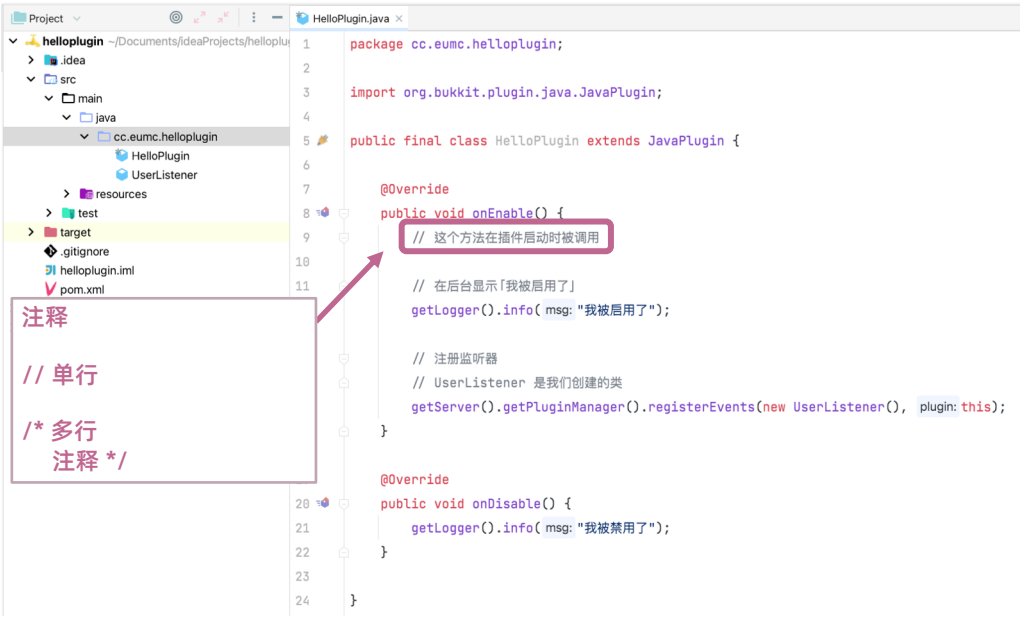
继承

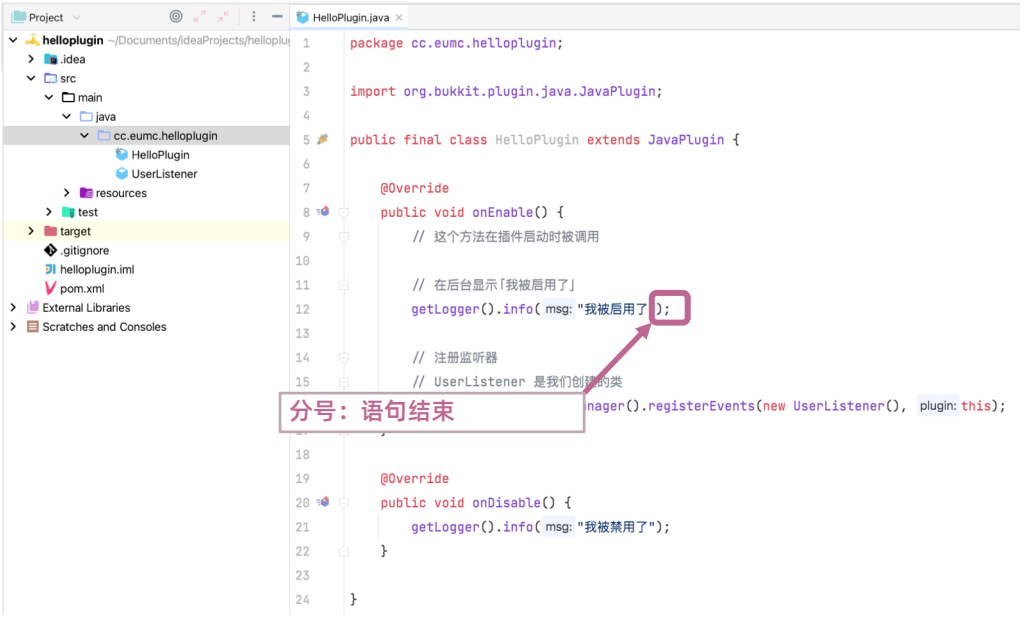


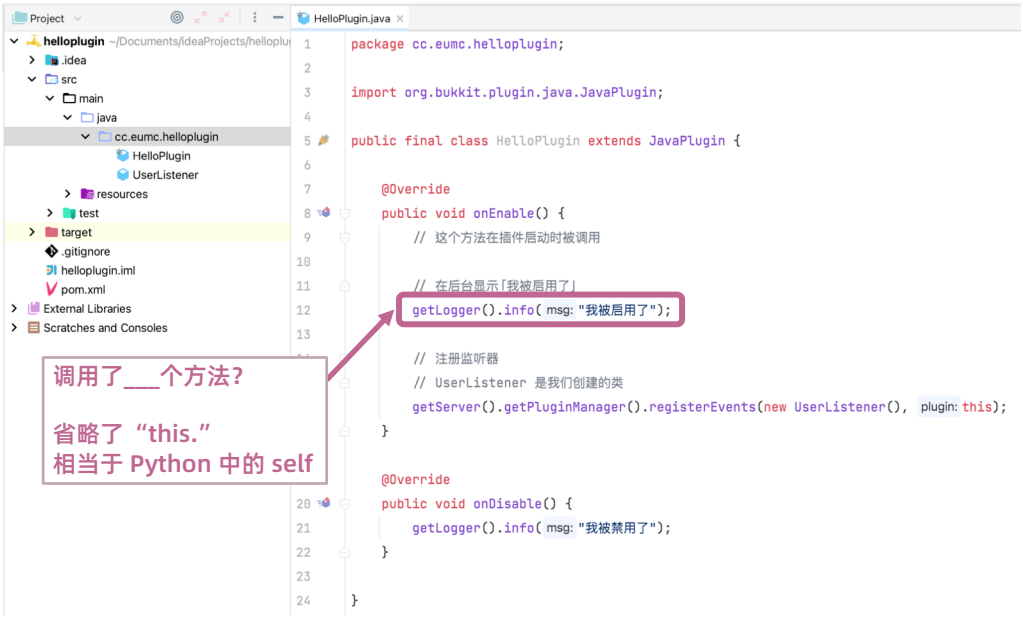
Axolotl Entities

^ Images: https://minecraft.fandom.com/wiki/Axolotl

语法基础


Project
helloplugin ~/Documents/ideaProjects/helloplu
.idea
src
main
java
cc.eumc.helloplugin
HelloPlugin
UserListener
resources
test
target
.gitignore
helloplugin.iml
pom.xml
HelloPlugin.java
package cc.eumc.helloplugin;
import org.bukkit.plugin.java.JavaPlugin;
public final class HelloPlugin extends JavaPlugin {
@Override
public void onEnable() {
// 这个方法在插件启动时被调用
// 在后台显示「我被启用了」
getLogger().info(msg: "我被启用了");
// 注册监听器
// UserListener 是我们创建的类
getServer().getPluginManager().registerEvents(new UserListener(), plugin: this);
}
@Override
public void onDisable() {
getLogger().info(msg: "我被禁用了");
}
}
注释
// 单行
/* 多行
注释 */

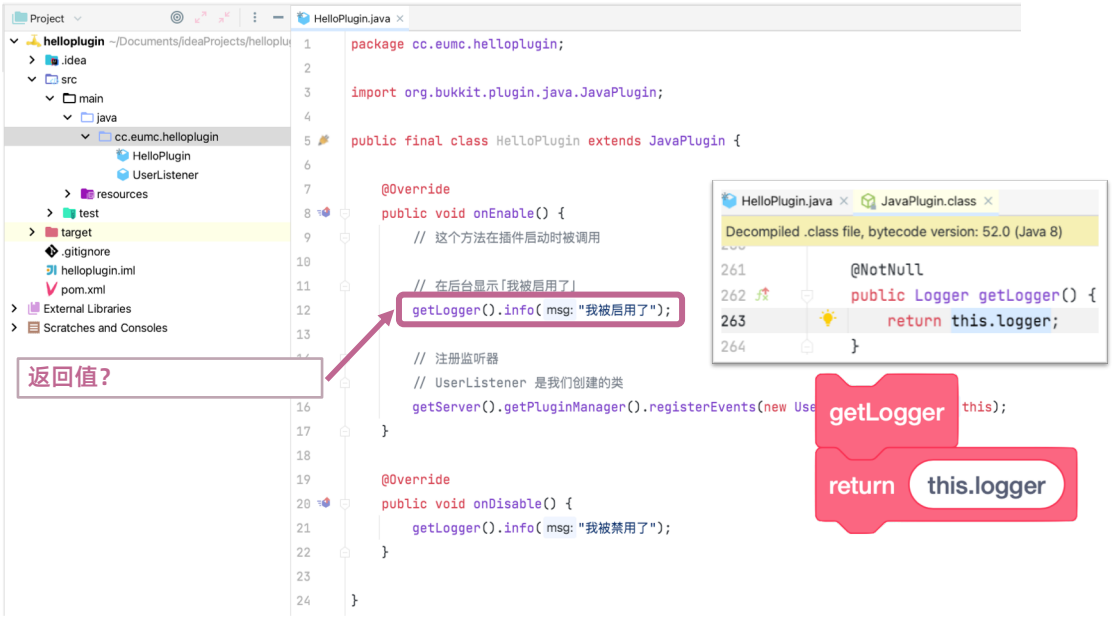

Project
helloplugin ~/Documents/ideaProjects/helloplu
.idea
src
main
java
cc.eumc.helloplugin
HelloPlugin
UserListener
resources
test
target
.gitignore
helloplugin.iml
pom.xml
External Libraries
Scratches and Consoles
HelloPlugin.java
package cc.eumc.helloplugin;
import org.bukkit.plugin.java.JavaPlugin;
public final class HelloPlugin extends JavaPlugin {
@Override
public void onEnable() {
// 这个方法在插件启动时被调用
// 在后台显示「我被启用了」
getLogger().info(msg: "我被启用了");
// 注册监听器
// UserListener 是我们创建的类
nager().registerEvents(new UserListener(), plugin: this);
@Override
public void onDisable() {
getLogger().info(msg: "我被禁用了");
}
}
分号：语句结束



返回值

- return _____;

```
void a() {}
int b() { return 0; }
double c() { return 0.1; }
float d() { return 0.1f; }
boolean e() { return true; }
```


Project
helloplugin ~/Documents/ideaProjects/helloplu
.idea
src
main
java
cc.eumc.helloplugin
HelloPlugin
UserListener
resources
test
target
.gitignore
helloplugin.iml
pom.xml
External Libraries
Scratches and Consoles
HelloPlugin.java
1 package cc.eumc.helloplugin;
2
3 import org.bukkit.plugin.java.JavaPlugin;
4
5 public final class HelloPlugin extends JavaPlugin {
6
7 @Override
8 public void onEnable() {
9 // 这个方法在插件启动时被调用
10
11 // 在后台显示「我被启用了」
12 getLogger().info(msg: "我被启用了");
13
14 // 注册监听器
15 // UserListener 是我们创建的类
16 getServer().getPluginManager().registerEvents(new Use this);
17 }
18
19 @Override
20 public void onDisable() {
21 getLogger().info(msg: "我被禁用了");
22 }
23
24 }
返回值?
HelloPlugin.java
JavaPlugin.class
Decompiled .class file, bytecode version: 52.0 (Java 8)
261 @NotNull
262 public Logger getLogger() {
263 return this.logger;
264 }
getLogger
return
this.logger

数据类型

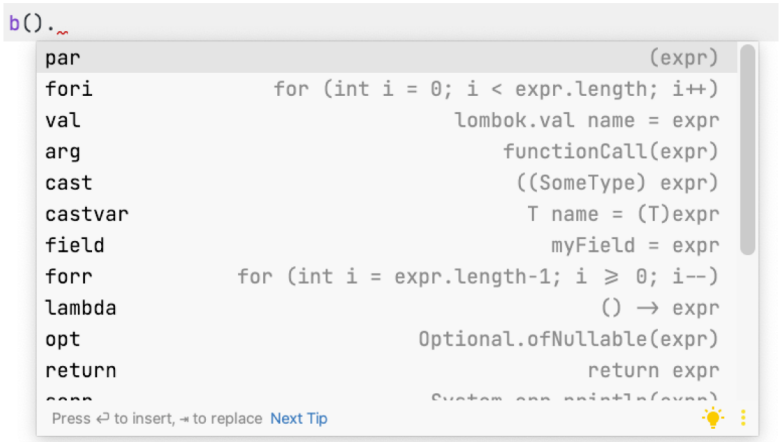
Primitive 类型

- `int b() { return 0; }`
- `double c() { return 0.1; }`
- `float d() { return 0.1f; }`
- `boolean e() { return true; }`
- `char f() { return 'a'; }`

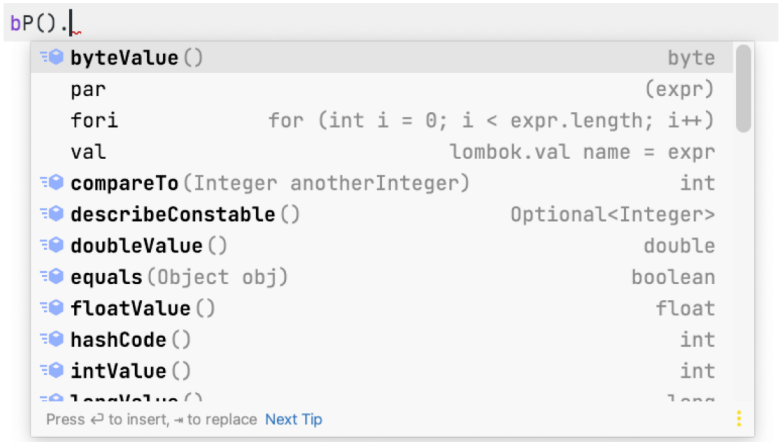
Non-Primitive 类型

- `Integer bP() { return 0; }`
- `Double cP() { return 0.1; }`
- `Float dP() { return 0.1f; }`
- `Boolean eP() { return true; }`
- `Character fP() { return 'a'; }`

Primitive 值



Non-Primitive 对象



字符串 String

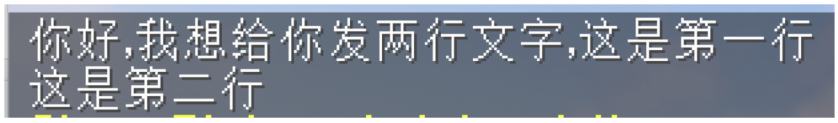
- 存储文本
- 双引号
- E.g.
  - event.getPlayer().sendMessage("Hello");

转义符 Escape

- event.getPlayer().sendMessage("你好，我想给你发两行文字" +
  "这是第二行");

转义符 Escape

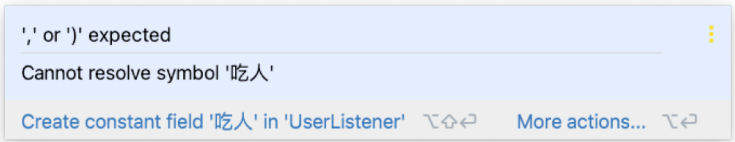
- event.getPlayer().sendMessage("你好，我想给你发两行文字\n" +
  "这是第二行");



转义符 Escape

- 改变字符原有的意思
  - E.g.
    - 双引号

```
sendMessage(""吃人"，周树人说。");
```



转义符 Escape

- 改变字符原有的意思
  - E.g.
    - 双引号

```
sendMessage( s: "\"吃人\"，周树人说。");
```


数据类型转换

转换 Casting

- 隐式转换
- 显式转换

隐式转换

- sendMessage("鲁迅说了这" + 2 + "个字。")

String int String

显式转换

- int playerExpClipped = player.getExp();

```
org.bukkit.entity.Player
public abstract float getExp()
```

float ⊄ int

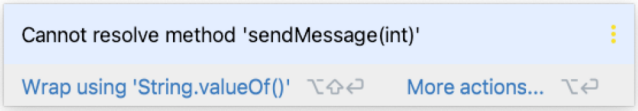
显式转换

- int playerExpClipped = (int) player.getExp();

特殊情况

- 「前缀小括号型」的显式转换并不是万能的

```
event.getPlayer().sendMessage(42);
```



对于代码：`(String)42`

```
|  Error:
|  incompatible types: int cannot be converted to java.lang.String
|  (String)42
|          ^^
```


特殊情况

- String 到 数字
  - (int) "42"　？？？？？
- Integer：Non-primitive 类型
  - 自带的静态（static）方法
    - Integer.parseInt(*String value*) → int

继承相关的类型转换

```
@EventHandler
public void onEntityDamage(EntityDamageEvent event) {
    Entity entity = event.getEntity();
```

```
org.bukkit.event.entity.EntityEvent
@NotNull
public org.bukkit.entity.Entity getEntity()
```


继承相关的类型转换

```
Entity entity = event.getEntity();

if (entity instanceof Player) {
    Player player = (Player) entity;
    player.sendMessage("你死掉了");
}
```

因为 Player 继承自 Entity

.. 或者简写成 （Java 14 新特性）

```
Entity entity = event.getEntity();

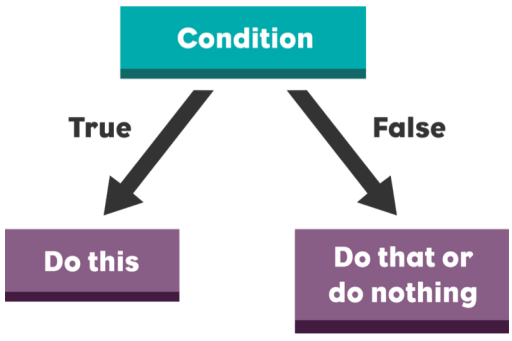
if (entity instanceof Player player) {
    player.sendMessage("你死掉了");
}
```


分支

boolean

- true
- false

If Statement



^ Ref: https://technovationchallenge.org/curriculum/coding-8/

If Statement

- 形如
  - if (event.getMessage().contains("vfrg2J(*!#%9u0g")) {
    event.getPlayer().kickPlayer("禁止脏话");
    } else {
    event.getPlayer().giveExp(1);
    }

循环

For Loop

- 形如
  - ```
    for (int i = 0; i < 100; i++) {
        event.getPlayer().sendMessage(i + "%");
    }
    ```


判断 – 执行

```
if (event.getPlayer().isSneaking()) {
   event.getPlayer().damage(10);
}
if (event.getPlayer().isSneaking()) {
   event.getPlayer().damage(10);
}
if (event.getPlayer().isSneaking()) {
   event.getPlayer().damage(10);
}
if (event.getPlayer().isSneaking()) {
   event.getPlayer().damage(10);
}
if (event.getPlayer().isSneaking()) {
   event.getPlayer().damage(10);
}
……
```
```

While Loop: 判断 – 执行

- 形如
  - while (event.getPlayer().isSneaking()) {
    event.getPlayer().damage(10);
    }

进阶 For Loop: 遍历

```
for (Type variableName : <Collection>) {
    // do something
}
```

进阶 For Loop: 遍历

- E.g. 在一个玩家上床的时候通知所有人
  - @EventHandler

```
public void onPlayerEnterBed(PlayerBedEnterEvent event) {
    for (Player player : Bukkit.getServer().getOnlinePlayers()) {
        player.sendMessage(event.getPlayer().getName() + " 上床了。");
    }
}
```


前置概念

硬编码 Hard-coded

- 数值/设置被写死在代码中的编码方式
- 例如
  - 规定超过 5 个人在线就关服
    - if (players.size() > 5) { < .. shutdown the server .. > }
  - 插件消息
    - sendMessage("Hello")
- 缺乏可维护性，想修改的时候要重新编译 *
- 下节课解决

Demo：定时飞行