



Workshop Minecraft 服务端插件开发

叶子 (Alan Richard) - August 14, 2021

Session 2



Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

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



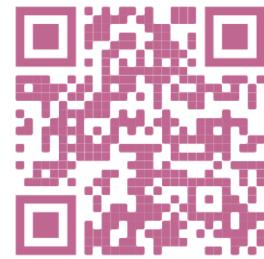
除了 OOP 还有什么编程？

kevin_Lin3rd
除了面向对象的编程还有什么编程，有什么特点



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



Saturday, August 14, 2021

5



Session 2: Java 基础

Saturday, August 14, 2021

6

cc BY NC ND Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

复习：ABC

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

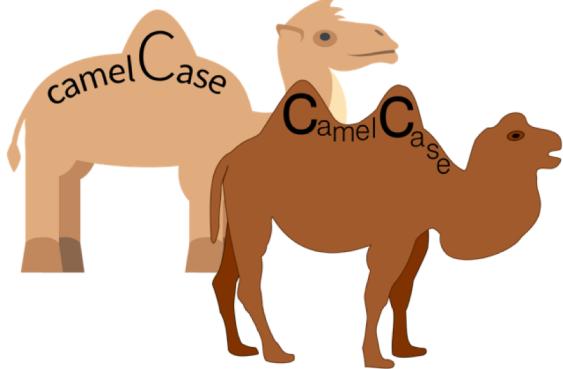


Image by Emoji One & Silver Spoon Sokpop

Saturday, August 14, 2021

7

cc BY NC ND Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

复习：ABC

- 缩进
 - 不像 Python 中那么严格
 - 让代码保持可读

Saturday, August 14, 2021

8

CC BY NC ND

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

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: Saturday, August 14, 2021
[https://joke-battles.fandom.com/wiki/Doge_\(Dogelore\)](https://joke-battles.fandom.com/wiki/Doge_(Dogelore))
<https://www.duitang.com/blue/?id=930071024>




9

CC BY NC ND

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

复习： ABC

- 面向对象编程（OOP）→ 「从属关系」

richard . devCourse

里查德 的 开发课

Saturday, August 14, 2021

10

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

OOP 基础

Saturday, August 14, 2021

11

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

类 Class

创建对象的蓝图

包含

- 方法 (methods)
- 属性 (attributes)

```

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 UserListener(), plugin: this);
17    }
18
19    @Override
20    public void onDisable() {
21        getLogger().info( msg: "我被禁用了" );
22    }
23
24 }

```

Saturday, August 14, 2021

12

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

```

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 UserListener(), plugin: this);
17     }
18
19     @Override
20     public void onDisable() {
21         getLogger().info( msg: "我被禁用了" );
22     }
23 }
24

```

Saturday, August 14, 2021

13

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

对象

Saturday, August 14, 2021

14



对象

- 实例化的类 (Instantiated classes)

- -- 创建对象

- `Object o = new Object();`

Saturday, August 14, 2021

15



创建对象

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

Saturday, August 14, 2021

16

创建对象

The diagram illustrates the creation of an object. On the left, the word "Object" is enclosed in a rounded rectangle with a pink border. An upward-pointing arrow originates from the bottom of this box and points to the variable "o" in the code. To the right of the arrow, the text "类型" (Type) is written. The code "Object o = new Object();" is displayed in black text on the right side of the slide.

Object o = new Object();

↑
类型

只需要创建
对象时声明

Saturday, August 14, 2021

17

创建对象

The diagram illustrates the creation of an object. The variable "o" is highlighted with a pink rounded rectangle. An upward-pointing arrow originates from the bottom of this box and points to the word "Object" in the code. To the right of the arrow, the text "变量名称" (Variable Name) is written. The code "Object o = new Object();" is displayed in black text on the right side of the slide.

Object o = new Object();

↑
变量名称

Saturday, August 14, 2021

18

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

创建对象

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

↑
关键词

Saturday, August 14, 2021

19

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

创建对象

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

↑
构建方法 *

Saturday, August 14, 2021

20



变量赋值

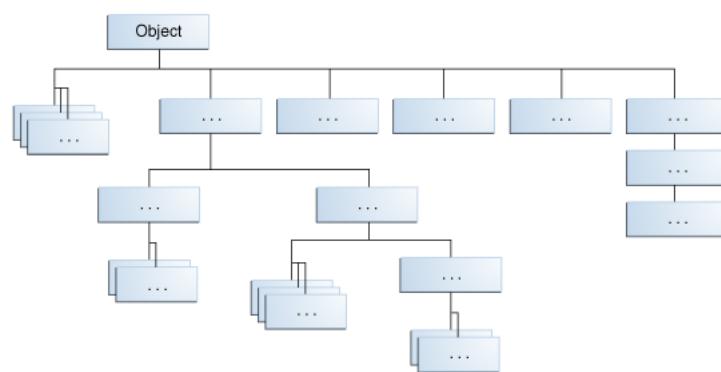
```
o = new Another();
o = "Hello";
```

Saturday, August 14, 2021

21



Object 生万物



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



Saturday, August 14, 2021

22

 Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

← 继承

- E.g. Axolotl



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

Saturday, August 14, 2021

23

 Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

继承

Package `org.bukkit.entity`

Interface **Axolotl**

All Superinterfaces:

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

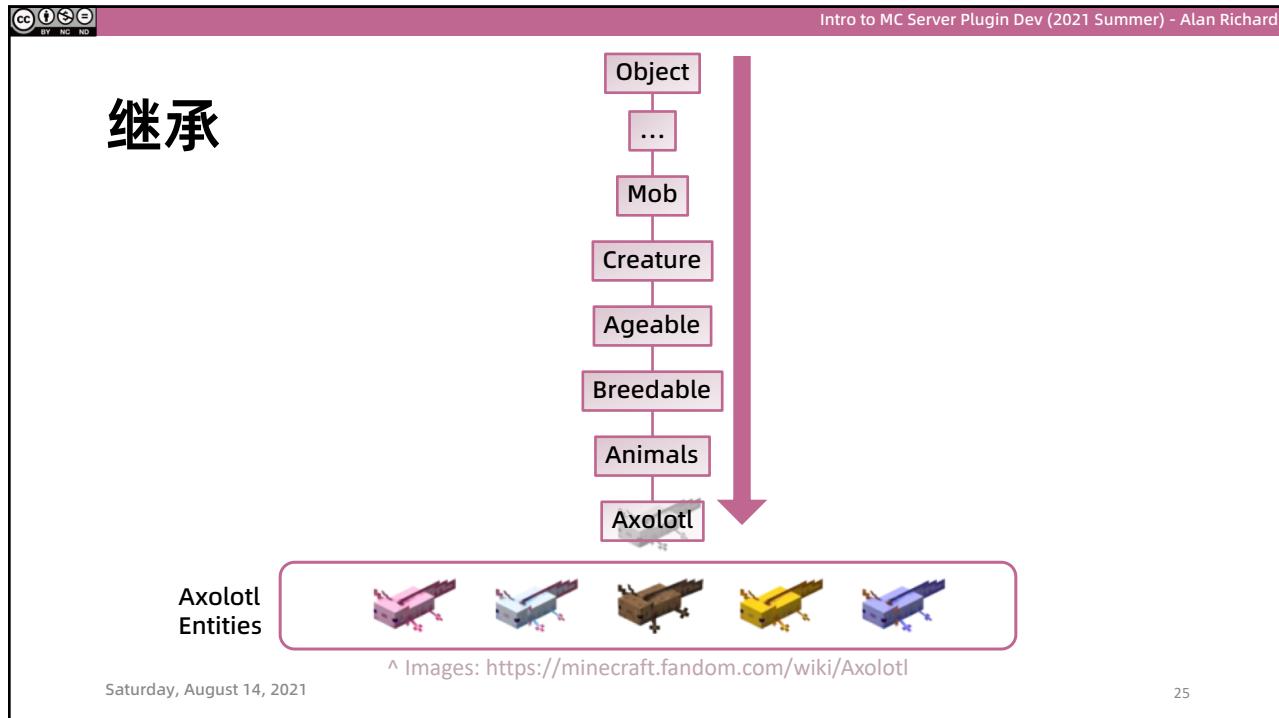
```
public interface Axolotl
extends Animals
```

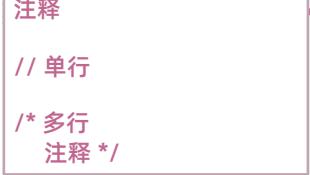
An Axolotl.



Saturday, August 14, 2021

24





Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

```

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 UserListener(), plugin: this);
17
18     }
19
20     @Override
21     public void onDisable() {
22         getLogger().info( msg: "我被禁用了" );
23     }
24 }

```

Saturday, August 14, 2021

27



Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

```

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 UserListener(), plugin: this);
17
18     }
19
20     @Override
21     public void onDisable() {
22         getLogger().info( msg: "我被禁用了" );
23     }
24 }

```

Saturday, August 14, 2021

28

调用了__个方法?
省略了“this.”
相当于Python中的self

```

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 UserListener(), plugin: this);
17     }
18
19     @Override
20     public void onDisable() {
21         getLogger().info( msg: "我被禁用了" );
22     }
23 }
24 
```

Saturday, August 14, 2021 29

返回值

- `return _____;`

```

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

```

Saturday, August 14, 2021 30

The screenshot shows the IntelliJ IDEA interface with the project 'helloplugin' open. The left pane displays the project structure, and the right pane shows the code editor for `HelloPlugin.java`. A callout box highlights the `getLogger()` method in the Java code, which is also visible in the decompiled `JavaPlugin.class` code on the right.

```

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( this ));
    @Override
    public void onDisable() {
        getLogger().info( msg: "我被禁用了" );
    }
}

```

Saturday, August 14, 2021

31



Saturday, August 14, 2021

32



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'; }`

Saturday, August 14, 2021

33



Non-Primitive 类型

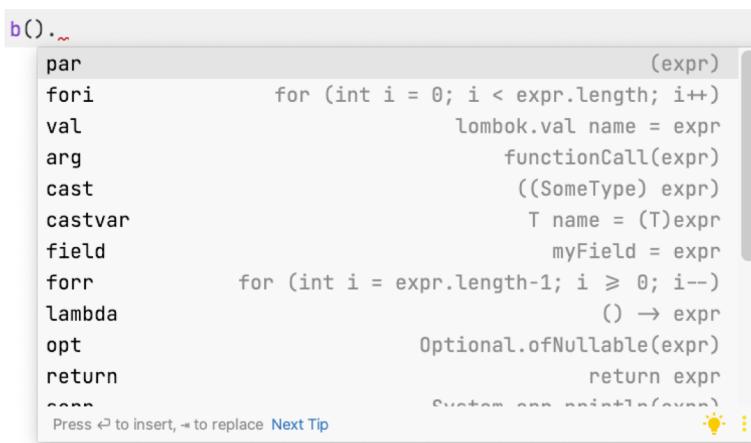
- `Integer bP() { return 0; }`
- `Double cP() { return 0.1; }`
- `Float dP() { return 0.1f; }`
- `Boolean eP() { return true; }`
- `Character fP() { return 'a'; }`

Saturday, August 14, 2021

34

cc BY NC ND Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

Primitive 值



The screenshot shows the IntelliJ IDEA code editor with a code completion dropdown open over the text "b().". The dropdown lists various methods for primitive types, each followed by its return type in parentheses:

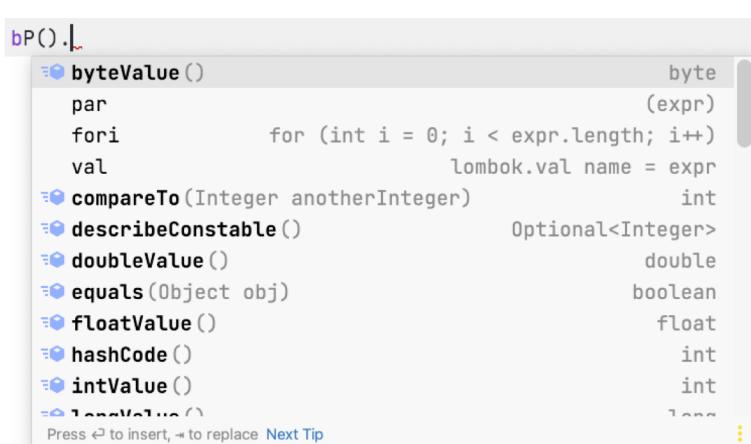
- par (expr)
- fori for (int i = 0; i < expr.length; i++)
- val lombok.val name = expr
- arg functionCall(expr)
- cast ((SomeType) expr)
- castvar T name = (T)expr
- field myField = expr
- forr for (int i = expr.length-1; i ≥ 0; i--)
- lambda () → expr
- opt Optional.ofNullable(expr)
- return return expr
- copy

At the bottom of the dropdown, there is a note: "Press ⌘ to insert, ⌘ to replace Next Tip".

Saturday, August 14, 2021 35

cc BY NC ND Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

Non-Primitive 对象



The screenshot shows the IntelliJ IDEA code editor with a code completion dropdown open over the text "bP().". The dropdown lists various methods for non-primitive objects, each followed by its return type in parentheses:

- byteValue() byte
- par (expr)
- fori for (int i = 0; i < expr.length; i++)
- val lombok.val name = expr
- compareTo(Integer anotherInteger) int
- describeConstable() Optional<Integer>
- doubleValue() double
- equals(Object obj) boolean
- floatValue() float
- hashCode() int
- intValue() int
- longValue() long

At the bottom of the dropdown, there is a note: "Press ⌘ to insert, ⌘ to replace Next Tip".

Saturday, August 14, 2021 36



Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

字符串 String

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

Saturday, August 14, 2021

37



Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

转义符 Escape

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

你好,我想给你发两行文字,这是第一行这是第二行

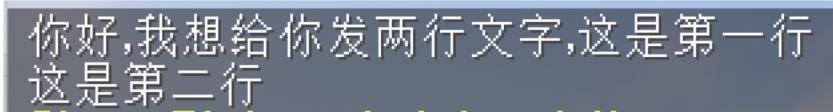
Saturday, August 14, 2021

38

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

转义符 Escape

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



Saturday, August 14, 2021

39

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

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。");
```



Saturday, August 14, 2021

40



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。);
```

Saturday, August 14, 2021

41



数据类型转换

Saturday, August 14, 2021

42



Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

转换 Casting

- 隐式转换
- 显式转换

Saturday, August 14, 2021

43



Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

隐式转换

- `sendMessage("鲁迅说了这" + 2 + "个字。")`

↑
String ↑ ↑
 int String

Saturday, August 14, 2021

44

CC BY NC ND

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

显式转换

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

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

`float ≠ int`

Saturday, August 14, 2021

45

CC BY NC ND

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

显式转换

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

↑

Saturday, August 14, 2021

46

cc BY NC ND Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

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);
```

Cannot resolve method 'sendMessage(int)'
[Wrap using 'String.valueOf\(\)'](#) More actions...

对于代码：(String)42

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

Saturday, August 14, 2021 47

cc BY NC ND Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

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

Saturday, August 14, 2021 48

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

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()
}

```

Saturday, August 14, 2021

49

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

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

Saturday, August 14, 2021

50

.. 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");  
}
```

Saturday, August 14, 2021

51

分支

Saturday, August 14, 2021

52

CC BY NC ND

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

boolean

- true
- false

Saturday, August 14, 2021

53

CC BY NC ND

Intro to MC Server Plugin Dev (2021 Summer) - Alan Richard

If Statement

```
graph TD; Condition[Condition] -- True --> DoThis[Do this]; Condition -- False --> DoThat[Do that or do nothing]
```

The diagram illustrates the logic of an if statement. At the top, a teal box labeled "Condition" has two arrows pointing downwards to two separate boxes. The left arrow, labeled "True", points to a purple box containing the text "Do this". The right arrow, labeled "False", points to a purple box containing the text "Do that or do nothing".

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

Saturday, August 14, 2021

54



If Statement

- 形如

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

Saturday, August 14, 2021

55



循环

Saturday, August 14, 2021

56



For Loop

- 形如

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

Saturday, August 14, 2021

57



判断 – 执行

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

Saturday, August 14, 2021

58



While Loop: 判断 – 执行

▪ 形如

```
▪ while (event.getPlayer().isSneaking()) {  
    event.getPlayer().damage(10);  
}
```

Saturday, August 14, 2021

59



进阶 For Loop: 遍历

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

Saturday, August 14, 2021

60



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。");  
    }  
}
```

Saturday, August 14, 2021

61



前置概念

Saturday, August 14, 2021

62



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

Saturday, August 14, 2021

63



Demo: 定时飞行

Saturday, August 14, 2021

64