**Java进阶**

**第二单元 泛型**

1. **本单元教学目标**
2. **知识目标**
3. 理解泛型的概念
4. 理解泛型规则限制
5. 了解系统提供的泛型类和泛型接口
6. 理解泛型类、泛型接口、泛型方法
7. 理解泛型通配符的作用
8. 理解有界通配符和有界类型形参的作用
9. 理解泛型的继承规则
10. **能力目标**
11. 会创建自定义泛型类
12. 会创建自定义泛型接口
13. 会创建自定义泛型方法
14. 会使用泛型通配符
15. 会使用有界通配符
16. 会使用有界类型形参
17. 会继承泛型类和实现泛型接口
18. **速度目标**
19. 10分钟内，根据需求完成泛型接口的定义
20. 10分钟内，根据需求完成泛型类的定义
21. 10分钟内，根据需求完成泛型方法的定义
22. 代码量800行
23. **课时分配**
24. **第一课时**
25. **知识目标**

* 理解泛型的作用
* 了解系统提供的泛型类和泛型接口
* 理解泛型类的概念
* 理解类型形参的概念
* 理解类型实参的概念

1. **能力目标**

* 会创建自定义泛型类

1. **第二课时**
2. **知识目标**

* 理解泛型接口的概念
* 理解泛型方法的概念
* 理解泛型参数的作用域

1. **能力目标**

* 会创建泛型接口
* 会创建泛型方法

1. **第三课时**
2. **知识目标**

* 理解泛型的概念
* 理解泛型通配符的概念
* 理解泛型规则限制

1. **能力目标**

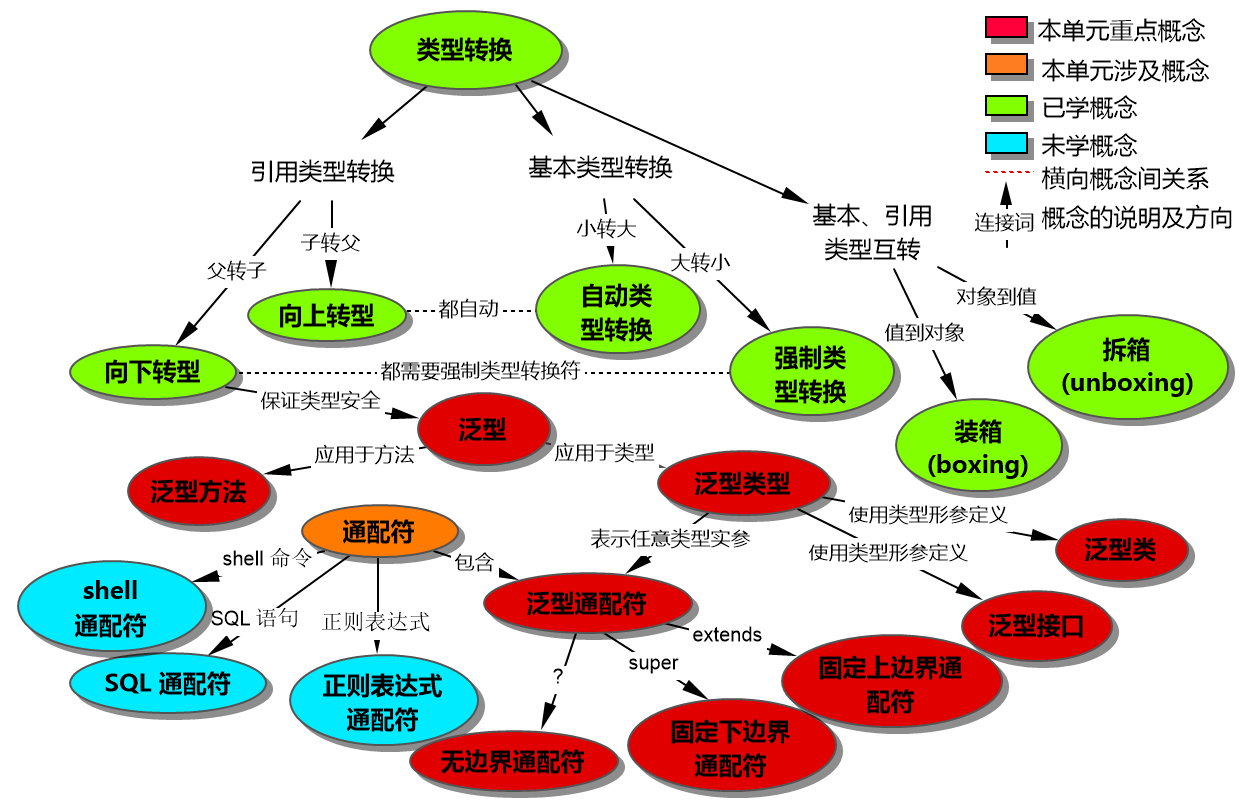
* 会使用泛型通配符
* 会使用有界通配符

1. **第四课时**
2. **知识目标**

* 理解有界类型形参

1. **能力目标**

* 会使用有界类型形参

1. **本单元概念图**
2. **教学过程脚本设计——第一课时**
3. **脚本设计思路**



1. **脚本切片设计思路**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **内容** | **目标** | **设计亮点** | **时间** |
| 1 | 课堂礼仪 | 1. 提升学生气势，精神饱满。 2. 增加团队意识。 | 1. 突出学院特色，增加学生学习的信心，坚定学习信念。 | 1min |
| 2 | 回顾 | 1. 巩固之前的知识。 | 1. 学习新知识，不忘旧知识，形成知识网络图。 | 3min |
| 3 | 知识目标、能力目标和贯穿案例 | 1. 锻炼学生自学能力。 2. 明确学习目标，做到学习有的放矢。 | 1. 培养学生建立自主学习能力，实现自我学习的突破潜力无极限。 | 3min |
| 4 | 为什么使用泛型 | 1. 通过案例使学生认识到学习泛型的必要性。 | 1. 通过案例导入技能点 | 5min |
| 5 | 什么是泛型 | 1. 认识常见的泛型类型。 2. 了解泛型的作用。 | 1. 通过源码演示，增加学生的认知能力。 | 6min |
| 6 | 自定义泛型类 | 1. 理解类型形参的概念。 2. 掌握泛型类的创建。 | 1. 通过类比学习类型形参。 2. 通过类比学习泛型类的创建 | 7min |
| 7 | 使用泛型类创建对象 | 1. 掌握使用泛型类创建对象。 | 1. 通过类比讲解创建泛型类的对象。 | 7min |
| 8 | 泛型类的概念 | 1. 理解泛型类的概念。 | 1. 通过“属+种差”下定义。 | 2min |
| 9 | 从泛型类派生子类 | 1. 掌握泛型类派生子类的实现方式。 | 1. 通过类比讲解泛型类的继承。 | 5min |

1. **脚本切片一**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **课堂礼仪**  **60s** | **讲师：**上课！  **班长：**起立!  **讲师：**同学们好。  **学生：**老师好。  **讲师：**我们班的口号是？  **学生：**…。  **讲师：**检查工装。  **班长：**1234。  **讲师：**报告出勤情况。  **班长：**应到\*人，实到\*人。  **讲师：**请坐。 | 1. 讲师进入教室前整理着装，提前30秒面带自信的笑容走进教室，给学生打招呼。当上课铃声响起，喊出“上课”，声音高于授课声音。 2. 学生起立、喊口号和拍手时，讲师需严格检查学生整齐度和速度，如凌乱需要重新做。 3. 口号结束后，讲师需跟学生一起有节奏的鼓掌（必须站在讲台前）。 4. 学生报告出勤情况时，讲师要面带微笑，并以肯定的眼神看着班长（报告出勤的同学）。 |

1. **脚本切片二：视频切片1**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **回顾**  **120s** | **1、准备讲解本单元内容**  根据ppt引导学生回答问题，并给出完整答案。   * 什么是枚举？ * 枚举和普通类的区别是什么？ * 如何创建枚举类型？ | 1. 面带微笑，声音洪亮，有激情。 2. 回顾可以采用提问、编码、默写的方式进行。 |
| **作业点评**  **60s** | **2、点评（作业进度、作业质量、代码量、共性问题）**  按照教学标准实施作业点评。 | 1. 作业点评突出提交人数、未提交人数，作业完成质量，共性问题。 2. 对于完成好的学生要点名表扬，完成不好的学生要点名警示。 3. 强调代码量必须完成，和学生达成一致。 |

1. **脚本切片三：视频切片2 视频切片3**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **预习检查**  **120s** | **1、通过提问方式检查预习情况**  通过提问方法检查预习情况，没必要给出正确答案，但必须强调预习的重要性。   * 什么是泛型？ * 泛型有哪些规则限制？ * 泛型可以应用于哪些地方？ * 泛型通配符有哪些？ | 1. 关注学生状态、表情，通过状态表情判断学生是否预习或者预习是否充分，可针对状态特殊的学生进行提问。 2. 回答精确给予表扬，鼓掌。 3. 回答不精确或不全面，引导学生找出精确答案，给予鼓励。 4. 没有回答出来的学生要适当批评，起到警示的作用。 |
| **目标**  **180s** | **2、讲解本单元知识目标**  通过所学知识，使用浅显易懂的方式讲解本单元知识目标。   * 理解泛型的概念 * 理解泛型规则限制 * 了解系统提供的泛型类和泛型接口 * 理解泛型类、泛型接口、泛型方法 * 理解泛型通配符的作用 * 理解有界通配符和有界类型形参的作用 * 理解泛型的继承规则 | 1. 逐一介绍本单元的知识目标。 |
| **3、讲解本单元能力目标**  通过所学知识，使用浅显易懂的方式讲解本单元能力目标。   * 会创建自定义泛型类 * 会创建自定义泛型接口 * 会创建自定义泛型方法 * 会使用泛型通配符 * 会使用有界通配符 * 会使用有界类型形参 * 会继承泛型类和实现泛型接口 | 1. 强调本单元的能力目标，让学生了解，学完本单元应具备的能力。 |
| **4、讲解本单元速度目标**  突出时间规定和代码量。   * 10分钟内，根据需求完成泛型接口的定义 * 10分钟内，根据需求完成泛型类的定义 * 10分钟内，根据需求完成泛型方法的定义 * 代码量800行 | 1. 强调时间规定和代码量。 |

1. **脚本切片四：视频切片4**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **为什么使用泛型**  **300s** | **1、通过数组存储班级学生姓名**  **讲师：**在学习新知识前，看一个案例。  讲师：班长通过学习，创建了一个辅助类来管理全班学生的姓名信息。  讲师：实现如下功能。  讲师：可以添加学生姓名，可以根据下标获得学生姓名。  讲师：也可以记录学生数。  讲师：储存所有学生姓名可不可以使用数组？  学生：可以。【参考回答】  讲师：可以编写一个类实现功能，使用数组存储所有的学生信息。  讲师：使用size储存当前学生数。  讲师：每次增加学生，size加1。  讲师：通过方法可以实现添加、查询、获得学生数的操作。 | 1. 通过案例引出知识点。 2. 讲解需求。 3. 演示代码和效果。 |
| **2、使用固定类型时可重用性差**  讲师：如果要管理商品信息、管理学生信息呢？  讲师：该怎么操作？  学生：各抒己见。  讲师：要替换下数据类型，把String换成Student。  讲师：添加方法的参数类型是Student，get方法的返回值类型也是Student。  讲师：如果管理的是商品信息呢？  学生：各抒己见。  讲师：同样，数据类型进行更改，把Student变为Goods。  讲师：实现了相同的功能，由于数据类型的改变，需要书写多个类。  讲师：代码的可重用性差。 | 1. 引导学生回答问题。 2. 突出是由于数据类型改变导致的问题。 3. 突出代码可重用性差。 |
| **3、使用Object类型修改代码**  讲师：思考一下，当数组的类型定义成什么时，可以存放所有的类型？  讲师：既可以放商品，又可以放学生？  学生：各抒己见。  讲师：数组定义成Object可以吧？  学生：可以。【参考回答】  讲师：Object是所有类的祖先。  讲师：定义成Object之后，既可以存放Student类型，又可以存放Goods类型。  讲师：重用性得到了提升吧？  学生：是。【参考回答】  讲师：向数组中添加学生信息时没有问题吧？  学生：没有。  讲师：当添加学生信息是发生向上转型。  讲师：当获取数据时涉及到向下转型。 | 1. 引导学生使用Object解决问题。 2. 强调装箱和拆箱。 3. 强调由于存放时没有限制类型，导致类型不安全。 |
| **4、使用Object时存在问题**  讲师：既然可以存放任意类型，就可以出现即存放Student，又存放字符串，再存放一个int类型数据的情况。  讲师：现在代码出问题了吗？  学生：各抒己见。  讲师：存放时没有，但是读取时存在问题。  讲师：代码中没有限制数组中要存放何种类型的数据。  讲师：第一条数据可以强转成Student类型。  讲师：“张三”可以转换成Student吗？  学生：不能。【参考回答】  讲师：要想把12放入数组，是不是要转换成Object？  学生：是。【参考回答】  讲师：是不是进行了装箱操作？  学生：是。【参考回答】  讲师：装箱操作是不是带来大量的系统消耗？  学生：是。【参考回答】  讲师：这段代码的可重用性得到了提升。  讲师：但是，涉及到大量装箱和拆箱操作，执行效率低。  讲师：同时，类型不安全。  讲师：在进行强制转换时，对象的原类型不确定。  讲师：在转换成目标类型时，容易出现异常。  讲师：那么，有没有一种方式，使装箱和拆箱操作减少，同时能保证类型安全和代码的可重用性呢？ | 1. 说明数据类型过多，造成强制类型转换报错。 2. 强调使用Object存放数据时无法保证类型安全。 3. 强调使用Object存放数据时存在装箱和拆箱操作。 |

1. **脚本切片五：视频切片5**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **什么是泛型**  **360s** | **1、通过共性代码引出解决方案**  讲师：来看一下这两段代码，研究一下有什么相同之处？  讲师：是不是除了类型之外，其它代码完全一样？  学生：各抒己见。  讲师：左边代码用的是Student类，右边代码是Goods类。  讲师：方法在调用时，可以传递不同的值。  讲师：如果可以像方法参数一样。  讲师：在创建对象时，指定类型为Student。  讲师：同时，可以指定其它的类型，比如Goods。  讲师：那么，这段代码的重用性就特别好了。  讲师：能不能像方法参数一样，在使用时指定具体类型。  讲师：红框中指定一个占位符，使用时指定具体的类型。  讲师：就可以只写一段代码来替代这些类了吧？  学生：是的。【参考回答】 | 1. 通过引导学生，共同找出两段代码的共同点。 2. 表明解决问题的方案。 |
| **2、常见的泛型类型**  讲师：Java提供了一种把类型当作参数传递的功能，就是泛型。  讲师：Java利用泛型提供了很多泛型类和泛型接口。  讲师：常用的泛型类和泛型接口有Map、HashMap、Collection、List、ArrayList、Set、HashSet以及Class。  讲师：前面几个是下单元重点学习的接口和类。  讲师：Class是在反射中要重点学习的类。  讲师：其中，尖括号中是参数，相当于方法的形参。  讲师：方法参数在使用时指定具体的值，此处在使用时指定具体的类型。  讲师：把类中使用这个参数的地方全都替换成具体的类型。 | 1. 说明常见的泛型类和接口。 2. 说明各个标识符的作用。 3. 演示ArrayList源码和创建对象。 4. 说明当指定具体类型后，代码中出现E的地方都被替换。 |
| **3、标识符的含义**  讲师：其中K、V等标记符可以使用其它字母，但是一般会使用下面几个。  讲师：E，代表元素，一般在集合中使用。  讲师：T，代表类型，在自定义泛型类时经常使用。  讲师；K和V也经常在集合中使用，集合的具体内容会在下一单元学习。  讲师：N代表使用时指定的类型应该是数值类型。  讲师：还有?，以及其它标识符。  讲师：使用时做到见名知意。  讲师：为了能理解泛型，来看一下ArrayList类。  讲师：public class ArrayList后跟了一个<E>。  讲师：在第432行的get方法处，使用了E。  讲师：其中public 后面应该是什么？  学生：返回值类型。【参考回答】  讲师：可以感觉到E是一种类型。  讲师：现在来使用一下ArrayList，创建一个ArrayList对象。  讲师：ArrayList<String> list1 = new ArrayList<String>();  讲师：list1.get()获得的数据是String类型。  讲师：ArrayList<String> list1 = new ArrayList<String>();  讲师：list2.get()获得的数据是Student类型。  讲师：通过创建对象指定E的具体类型，是不是重用性特别好？  学生：是。【参考回答】  讲师：当使用add()方法增加其它类型数据时，提示类型不匹配。  讲师：可见，泛型提升了可重用性。 | 1. 说明标识符的含义。 2. 说明可以使用其它字母，但是建议使用PPT中标注的，见名知意。 |
| **4、泛型的作用**  讲师：使用参数创建类，在使用时传递具体的类型。  讲师：也就是说，泛型的本质是参数化类型。  讲师：这样编写的代码有什么好处呢？  讲师：是不是同一份代码上可以操作多种数据类型了？  学生：是。【参考回答】  讲师：也就是可重用好。  讲师：当调用ArrayList的add方法传入非指定类型数据时会怎样？  学生：各抒己见。  讲师：出现编译错误，提示类型不匹配。  讲师：限定了数据类型，类型是安全的。  讲师：使用Object数组时可以存放任何类型数据。  讲师：但是，当取数据时，需要进行强制类型转换，容易产生异常。  讲师：既然限制了类型，会减少装箱和拆箱操作，提高执行效率。  讲师：此处，因为add的参数类型和get的返回值类型都是E。  讲师：因为存取时类型相同，强制类型转换不会产生异常。  讲师：减少了装箱和拆箱操作，所以执行效率高。  讲师：这三点就是泛型的作用。 | 1. 强调泛型的本质。 2. 说明泛型的作用。 3. 强调可重用好、类型安全、执行效率高。 |

1. **脚本切片六：视频切片6 视频切片7**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型的使用方式**  **60s** | **1、泛型的三种使用场景**  **讲师：**刚才看到，泛型可以应用到类中或接口中。  讲师：除此之外，还可以应用到哪些地方呢？  讲师：泛型可以应用到接口、类和方法中。  讲师：比如Map接口、Collection接口、List接口、Set接口。  讲师：这些都是使用泛型而形成的泛型接口。  讲师：HashMap、ArrayList、HashSet和Class都是使用泛型而形成的泛型类。  讲师：Collection接口中的toArray方法是使用泛型而形成的泛型方法。  讲师：其中，应用于类型形成泛型类型，包含泛型接口和泛型类。  讲师：应用于方法形成泛型方法。  讲师：那么，如何定义泛型类型和泛型方法呢？ | 1. 强调泛型可以应用于泛型类、泛型接口和泛型方法。 2. 强调泛型类和泛型接口是泛型类型。 |
| **类型形参**  **180s** | **2、泛型类的创建和类型形参的定义**  讲师：先来自定义泛型类。  讲师：自定义泛型类的语法是public class 类名，后面跟<>。  讲师：演示ArrayList时，<>中写了一个E。  讲师：<>中是类型形参列表。  讲师：方法中的参数相当于数值形参。  讲师：看一下右侧代码，此处的T就是类型形参。  讲师：方法中的形参是接受值，此处是接受类型。  讲师：所以T就可以声明变量，作为方法的参数类型，作为返回值类型。  讲师：add方法中，是不是把T作为参数类型了？  学生：是。【参考回答】  讲师：get方法中，是不是把T作为返回值类型了？  学生：是。【参考回答】 | 1. 比较普通类和泛型类语法。 2. 说明T的作用。 |
| **3、类型形参的内涵**  讲师：其中类型形参列表是创建泛型类、泛型接口、泛型方法时声明在尖括号中的形参。  讲师：它是在创建泛型类、泛型接口、泛型方法时使用。  讲师：需要在<>中。  讲师：是形式参数。  讲师：如果有多个类型形参，需要用逗号隔开。 | 1. 强调类型形参的内涵。 |
| **创建泛型类**  **180s** | **4、创建泛型类**  讲师：看右侧代码，并没有使用T类型声明数组。  讲师：是因为无法直接创建T类型的实例。  讲师：不知道T的具体类型，是否有无参构造，内部结构是完全未知的。  讲师：所以，无法创建直接创建T的实例。  讲师：所以定义的是Object类型的数组。  讲师：它的构造方法带<T>吗？  学生：不带。【参考回答】  讲师：记住，构造方法名和类名相同，不带类型形参。 | 1. 强调构造方法的书写。 2. 强调T类型无法直接创建实例。 |
| **5、泛型类中的方法**  讲师：接下来，看一下使用了T类型的两个方法。  讲师：先看第一个方法。  讲师：add方法使用T作为方法参数类型。  讲师：添加数据前，判断数组是否已满。  讲师：当元素个数大于等于数组长度时，说明数组已满。  讲师：此时需要扩充数组容量。  讲师：创建新数组，并将原数组元素赋值给新数组。  讲师：最后，将新数组赋值给成员变量。  讲师：添加时需要执行size++，并添加新元素。  讲师：获得元素时需要根据下标。  讲师：如果下标超过数组最大下标，抛出元素下标越界的异常。  讲师：否则，返回当前下标对应的元素。  讲师：当添加和获得时，元素的类型相同吗？  学生：相同。【参考回答】  讲师：类型都是T。  讲师：在添加时是T类型，获取时再强制转换成T，有问题吗？  学生：没有。【参考回答】  讲师：存数据和取数据的类型相同，所以类型安全。  讲师：在使用Object直接存取数据时，可以添加任意类型的数据。  讲师：在获取时进行强制类型转换，不确定对象的具体类型。  讲师：强制转换会产生异常，导致类型不安全。 | 1. 强调使用泛型后，类型安全。 |

1. **脚本切片七：视频切片8**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型类的对象** | **1、创建泛型类对象的语法**  讲师：到目前为止，就根据泛型创建了泛型类。  讲师：如何去使用泛型类创建对象呢？  讲师：接下来学习泛型类创建对象。  讲师：当创建对象时指定T的具体类型。  讲师：来看一下语法。  讲师：类名<类型实参列表> 对象名 = new 类名<类型实参列表>(参数列表);  讲师：和普通类创建对象的区别是有类型实参列表。  讲师：比如，ArrayUtil<String> au = new ArrayUtil<String>();  讲师：当创建au对象后，ArrayUtil中所有的T都会替换成String。 | 1. 说明语法规则。 |
| **2、泛型实参**  讲师：其中，String是类型实参。  讲师：类型实参是泛型类创建对象、实现泛型接口或调用泛型方法时表示具体数据类型的实参。  讲师：注意，类型形参是创建类、创建接口、创建方法时使用。  讲师：类型实参时使用类、使用接口、使用方法时指定。  讲师：类型实参只能是引用数据类型。  讲师：同时它是实际参数。  讲师：方法中实参是具体的值，类型实参是具体的类型。 | 1. 讲解泛型实参的定义和内涵。 |
| **3、创建泛型类对象的其它语法**  讲师：在创建对象时，还有其它语法，比如菱形语法。  讲师：菱形语法省略了等号右边类名后的<>中的实参列表。  讲师：比如ArrayUtil<String> au = new ArrayUtil<>();  讲师：当然，这两个菱形都可以省略。  讲师：省略后，就像普通类一样来创建对象了。  讲师：使用这种方式创建对象，类型实参是什么类型呢？  学生：各抒己见。  讲师：默认类型实参为Object类型。 | 1. 简单说明创建对象的其它语法。 |
| **4、测试**  讲师：创建完对象后，来访问成员方法。  讲师：首先创建ArrayUtil的实例，指定T类型为String类型。  讲师：add方法的参数类型就是String了吧？  学生：是。【参考回答】  讲师：get方法的返回值类型也是String了吧？  学生：是。【参考回答】  讲师：看一下代码演示，测试ArrayUtil类。  讲师：当指定T为String时，可以添加字符串。  讲师：当指定T为Integer类型时，可以添加Integer。  讲师：调用时add方法时，提示需要Integer类型的参数。  讲师：也就是，在创建对象时确定具体数据类型。  讲师：类型实参只能是引用数据类型。  讲师：来看一下是否能指定为int类型。  讲师：提示错误，只能是引用数据类型。  讲师：可见，它是在编译时进行类型安全检查。 | 1. 强调编译时进行类型安全检查。 |

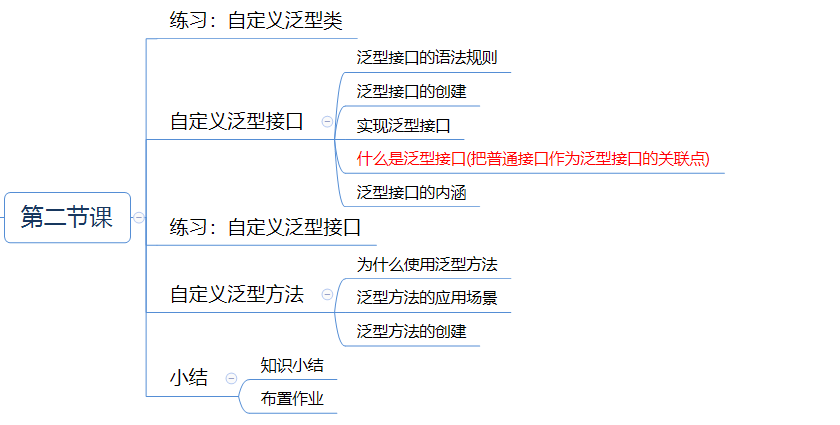
1. **脚本切片八：视频切片9**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型类的概念** | **1、泛型类的定义**  讲师：到目前就可以创建和使用泛型类了。  讲师：现在可以给泛型类下定义了。  讲师：首先来看一下泛型类的英文单词。  讲师：Generic是一般的、泛化的、通用型的。  讲师：泛型类中使用T泛指某种类型。  讲师：泛型类就是Generic Class。  讲师：那么，到底什么是泛型类呢？  讲师：泛型类的属是什么？  学生：类。【参考回答】  讲师：泛型类也是通过class创建的类吧？  学生：是。【参考回答】  讲师：它的创建和普通类的创建有什么不同？  学生：各抒己见。  讲师：是不是使用类型形参定义的？  学生：是。【参考回答】  讲师：泛型类是使用类型形参定义的类。 | 1. 带领学生找到泛型类的种差+属。 2. 和学生共同得出泛型类的定义。 |
| **2、泛型类的内涵和使用要点**  讲师：泛型类有哪些内涵呢？  学生：各抒己见。  讲师：它是使用类型形参定义的。  讲师：在使用泛型类时，需要注意下面几点。  讲师：第一，实例化泛型类的对象时，指定泛型的具体数据类型。  讲师：第二，类型实参默认是Object类型。  讲师：第三，编译时进行类型安全检查。  讲师：泛型是个语法糖，在编译阶段进行类型检查。  讲师：编译完成后，泛型就被擦除掉。  讲师：大家掌握了泛型类的概念了吗？ | 1. 总结泛型类的内涵。 2. 带领学生总结泛型类在使用时的要点。 |

1. **脚本切片九：视频切片10**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **从泛型类派生子类** | **1、两种继承泛型类的方式**  讲师：掌握了泛型类的创建和使用，思考一个问题。  讲师：如何继承一个泛型类呢？  学生：各抒己见。  讲师：泛型类中的构造方法是不带类型参数的。  讲师：派生的子类中需不需要带类型参数呢？  讲师：可以继续声明类型形参。  讲师：public class ArrayUtil<T> extends Addable<T> { }。  讲师：当创建ArrayUtil的实例时，可以指定T的类型。  讲师：ArrayUtil和Addable中使用T的地方都会被替换。  讲师：第二方式是传入类型实参。  讲师：在继承父类时，传入类型实参列表。  讲师：public class SubArrayUtil extends ArrayUtil <String> { }。  讲师：父类中的T在继承前就会被替换成String。  讲师：当不指定类型实参时，默认为Object类型。  讲师：创建泛型类的对象时，不指定类型实参，是不是也是Object？  学生：是。【参考回答】 | 1. 强调两种实现方式的区别。 |
| 讲师：接下来看一下方法的重写。  讲师：当不传入类型实参，也就是子类也使用T时，重写方法的声明不变。  讲师：可以看到，原先是T的地方，依然是T。  讲师：SubArrayUtil是不是个泛型类？  学生：是。【参考回答】  讲师：子类中也可以使用T。  讲师：当传入类型实参时。  讲师：此处指定了String，会把父类中所有出现T的位置替换成String。  讲师：子类重写时，T类型已经进行了替换。  讲师：来看一下eclipse中的代码。  讲师：在第一种方式中，把子类的T删除可以吗？  学生：各抒己见。  讲师：删除后报语法错误，为什么？  学生：各抒己见。  讲师：父类是泛型类，当创建子类的对象时，T无法确定具体类型。  讲师：所以，必须在创建子类的对象时指定T的具体类型。  讲师：在父类和子类中都声明T类型变量时。  讲师：来看一下，创建对象后，这两个变量的类型。  讲师：当指定类型实参为String时，两个变量都是String类型。  讲师：可见，同时指定了父子类中T的类型。  讲师：第二种，指定父类的类型实参后，会先进行替换，后重写。  讲师：现在替换成Integer。  讲师：下面重写方法相应的地方也要替换成Integer。  讲师：重写时，会先把父类中所有的类型形参替换成类型实参，然后进行重写。 | 1. 进行代码演示。 |

1. **教学过程脚本设计——第二课时**
2. **脚本设计思路**



1. **脚本切片设计思路**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **内容** | **目标** | **设计亮点** | **时间** |
| 1 | 学员操作：泛型类 | 1. 掌握泛型类的创建和使用。 | 1. 边讲边练，巩固所学技能点。 | 10min |
| 2 | 泛型接口的创建和使用 | 1. 掌握泛型接口的创建。 2. 掌握如何实现泛型接口。 | 1. 使用类比方式讲解。 | 9min |
| 3 | 泛型接口的概念 | 1. 掌握泛型接口的概念。 | 1. 通过“属+种差”下定义。 | 2min |
| 4 | 学员操作：泛型接口 | 1. 掌握泛型接口的使用。 | 1. 边学边练，实时掌握技能点。 | 10min |
| 5 | 为什么使用泛型方法 | 1. 理解泛型方法的使用场景。 | 1. 通过案例引出场景。 | 2min |
| 6 | 泛型方法的创建 | 1. 掌握泛型方法的创建。 | 1. 通过类比方式讲解。 | 4min |

1. **脚本切片一 ：视频切片1**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **练习：自定义泛型类**  **600s** | **讲师：**掌握了泛型类的创建和使用，来使用泛型类做一个上机练习。  讲师：主要训练泛型类的创建和使用。  讲师：创建一个泛型类，用来表示点的坐标信息。  讲师：一个坐标是由x，y表示的。  讲师：在保证数据安全的前提下，支持坐标值可以使用整数、小数等数据类型表示。  讲师：是不是可以使用泛型？  学生：是。【参考】  讲师：主要实现思路如下。  讲师：第一步，创建泛型类DrawPoint。  讲师：第二步，使用泛型类型声明坐标变量。  讲师：第三步，测试。  讲师：给大家10分钟时间完成相应任务。 | 1. 讲解训练点和需求。 2. 说明练习时间。 |

1. **脚本切片二：视频切片2 视频切片3**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型接口的创建和实现**  **240s** | **1、创建泛型接口**  **讲师：**接下来，看一下泛型在接口中的使用，形成泛型接口。  讲师：首先来看一下基本语法。  讲师：定义泛型接口和普通接口有什么区别？  学生：各抒己见。  讲师：多了一个类型形参列表。  讲师：泛型类的定义个泛型接口是相似的。  讲师：知道语法之后，来定义泛型接口。  讲师：定义两个泛型接口Addable和Getable。  讲师：两者都声明了类型形参T。  讲师：T作为类型，可以在接口中声明常量、作为方法的参数类型、方法的返回值类型。 | 1. 讲解泛型接口的语法。 |
| **2、实现泛型接口**  讲师：创建完泛型接口后，需要实现泛型接口。  讲师：第一种，继续声明类型形参。  讲师：在实现类中声明了类型形参列表。  讲师：通过下面三个语句来了解一下语法。  讲师：这样就可以在new ArrayUtil时指定T的具体类型。  讲师：这样就可以把ArrayUtil类和Addable中的T同时替换。  讲师：第二条语句实现了两个接口。  讲师：当new ArrayUtil<String>时，会同时替换实现类和接口中的T为String。  讲师：第三语句，在实现类中指定了两个类型形参。  讲师：接口中的类型形参必须出现在实现类中。  讲师：比如new ArrayUtil<Integer,String>，其中Integer对应T。  讲师：会把实现类和Addable中的T替换成Integer。  讲师：String对应S位置，会把实现类和Getable中的S替换成String。  讲师：第二种，传入类型实参。  讲师：在接口名后有类型实参列表。  讲师：在泛型类继承时学过，传入类型实参会造成什么效果？  学生：各抒己见。  讲师：先把类型形参替换成类型实参，然后进行重写。  讲师：类型实参列表可以省略，省略之后类型实参是什么？  学生：各抒己见。  讲师：类型实参省略时，默认是Object类型。  讲师：这样就掌握了泛型接口的创建以及实现泛型接口。 | 1. 与继承泛型类进行比较，对比教学。 |
| **重写泛型接口中的方法**  **300s** | 讲师：接下看一下方法的重写。  讲师：第一种情况，继续声明类型形参。  讲师：重写方法的声明格式不变，保留类型形参。  讲师：在创建ArrayUtil类的对象时指定T的具体类型。  讲师：同时替换实现类和接口中的T。  讲师：实现两个接口时，保证实现类和接口中的类型形参名对应。  讲师：当创建new ArrayUtil<Integer,String>()时。  讲师：Integer对应T，会替换成ArrayUtil和Addable中的T。  讲师：Addable声明了T，重写add方法时保留T。  讲师：Getable声明了S，重写get方法时保留S。  讲师：第二种方式，传入类型实参。  讲师：此时是先替换还是先重写？  学生：替换。【参考回答】  讲师：会把Addable中的类型形参替换成String，然后进行重写。  讲师：下面代码中，会把Addable中的类型形参替换成String，把Getable中的类型形参替换成Object。  讲师：来看一下源代码。  讲师：需要注意，类型形参名在实现接口时和创建接口时可以不同。  讲师：implements后的接口中声明的类型形参必须在实现类中出现。  讲师：大家对泛型接口的创建和使用掌握了吗？  学生：掌握了。【参考回答】 | 1. 举例说明讲解内容。 |

1. **脚本切片三：视频切片4**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型接口的概念**  **120s** | **1、泛型接口的定义**  讲师：通过刚才的学习，就掌握了泛型接口的创建和使用。  讲师：接下来，尝试给泛型接口下定义。  讲师：首先，来看一下泛型接口的英文单词。  讲师：泛型接口是Generic Interface。  讲师：到底什么是泛型接口呢？  讲师：感觉泛型接口的属是什么？  学生：各抒己见。  讲师：泛型接口和普通接口有什么区别？  学生：各抒己见。  讲师：泛型接口是使用类型形参定义的接口。 | 1. 通过种差+属的方式，带领学生共同得出泛型接口的定义。 |
| **2、泛型接口的内涵和使用要点**  讲师：泛型接口的内涵就是使用类型形参定义。  讲师：在使用泛型接口时，有哪些要点呢？  讲师：实现接口时可以指定类型实参。  讲师：如果不指定类型实参，可以继续使用类型形参。  讲师：类型实参可以延迟到创建对象时指定。  讲师：类型实参默认是Object类型。 | 1. 总结泛型接口的内涵和使用要点。 |

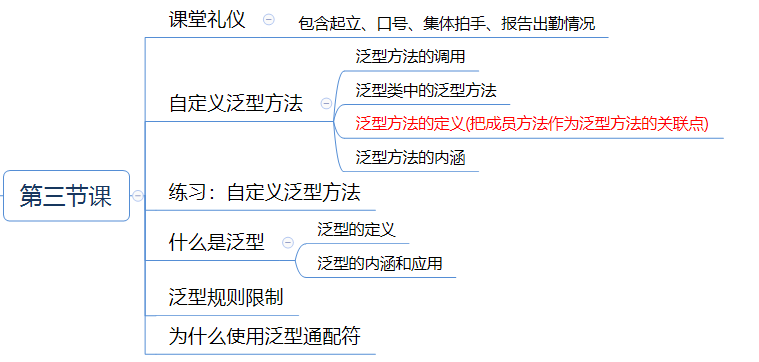
1. **脚本切片四：视频切片5**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **练习：自定义泛型接口**  **600s** | **讲师：**掌握了泛型接口的创建和使用后，来做一个上机练习。  讲师：主要训练泛型接口的创建和实现。  讲师：需要实现以下功能。  讲师：定义泛型接口Moveable，声明移动方法。  讲师：实现Moveable接口，并测试。  讲师：主要实现思路如下。  讲师：第一步，创建泛型接口Moveable，声明move(x,y)方法。  讲师：x，y代表点的横纵坐标，使用类型形参进行声明。  讲师：第二步，使用DrawPoint实现Moveable接口。  讲师：第三步，测试。  讲师：给大家10分钟时间完成相应任务。 | 1. 讲解训练点和需求。 2. 说明练习时间。 |

1. **脚本切片五：视频切片6 视频切片7**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **为什么使用泛型方法**  **120s** | **1、静态方法无法使用泛型类型中声明的T**  **讲师：**看下面一个案例。  讲师：根据项目需求，提供把ArrayUtil类型变量转换为数组的功能。  讲师：功能需求如下。  讲师：转换时类型对应，比如ArrayUtil<String>变量转换的结果String[]数组。  讲师：由于功能通用，同时便于调用，使用静态方法实现。  讲师：在泛型类中创建了一个泛型方法，使用泛型类中声明的类型形参。  讲师：思考一下，能使用泛型类中声明的类型形参吗？  学生：各抒己见。  讲师：泛型实参是在什么时候指定的？  学生：实例化对象时。【参考回答】  讲师：实例化泛型类的对象时指定泛型实参。  讲师：但是，静态方法在类加载时加载。  讲师：static方法在调用前需要实例化对象吗？  学生：不需要。【参考回答】  讲师：在调用静态方法是，T可能没有具体类型。  讲师：实例化泛型类对象时指定类型实参，静态方法是在类加载时加载。  讲师：调用前不需要实例化对象。  讲师：此时，调用change方法是不允许的。  讲师：如何在静态方法中使用泛型呢？  讲师：使用泛型方法就可以了。 | 1. 通过需求讲解为什么使用泛型方法。 2. 说明静态类中无法使用类中声明的T的原因。 |
| **泛型方法的运用场景和语法**  **240s** | **2、创建泛型方法**  讲师：在静态方法中使用泛型时，必须使用泛型方法。  讲师：只想在某几个方法种运用泛型时，可以使用泛型方法。  讲师：来看一下创建泛型方法的语法。  讲师：和普通方法有什么区别？  学生：各抒己见。  讲师：泛型类、泛型接口、泛型方法都使用了类型形参。  讲师：泛型方法与普通方法的区别是有类型形参。  讲师：泛型方法不但可以在泛型类中、也可以在普通类中。  讲师：在方法中声明的T，需要在方法调用时指定具体类型。  讲师：下面定义的方法是泛型方法吗？  学生：不是。【参考回答】  讲师：不是，没有类型形参列表，T是泛型类中声明的类型。  讲师：它是使用类中声明的T类型，不是泛型方法。  讲师：转换方法中，只需要在static后加入类型形参即可。  讲师：在转换时，判断数组长度是否能存放整个ArrayUtil的信息。  讲师：如果长度不够，抛出异常。  讲师：如果长度足够，通过循环把ArrayUtil的信息赋值到数组中。 | 1. 强调类型形参列表的位置。 |

1. **教学过程脚本设计——第三课时**
2. **脚本设计思路**



1. **脚本切片设计思路**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **内容** | **目标** | **设计亮点** | **时间** |
| 1 | 课堂礼仪 | 1. 提升学生气势，精神饱满。 2. 增加团队意识。 | 1. 突出学院特色，增加学生学习的信心，坚定学习信念。 | 1min |
| 2 | 泛型方法的调用 | 1. 掌握泛型方法的使用。 | 1. 边讲边练。 | 12min |
| 3 | 泛型方法的概念 | 1. 理解泛型方法的概念。 | 1. 通过“属+种差”得出定义。 | 3min |
| 4 | 学员操作：泛型方法 | 1. 掌握泛型方法的使用。 | 1. 边学边练，巩固所学知识。 | 10min |
| 5 | 泛型的概念 | 1. 理解泛型的概念。 | 1. 通过“属+种差”得出定义。 | 4min |
| 6 | 泛型规则限制 | 1. 理解泛型规则限制。 | 1. 引导学生自主思考。 | 5min |
| 7 | 为什么使用泛型通配符 | 1. 理解泛型通配符的作用。 | 1. 通过需求驱动学生思考。 | 4min |

1. **脚本切片一**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **课堂礼仪**  **60s** | **讲师：**上课！  **班长：**起立!  **讲师：**同学们好。  **学生：**老师好。  **讲师：**我们班的口号是？  **学生：**…。  **讲师：**检查工装。  **班长：**1234。  **讲师：**报告出勤情况。  **班长：**应到\*人，实到\*人。  **讲师：**请坐。 | 1. 讲师进入教室前整理着装，提前30秒面带自信的笑容走进教室，给学生打招呼。当上课铃声响起，喊出“上课”，声音高于授课声音。 2. 学生起立、喊口号和拍手时，讲师需严格检查学生整齐度和速度，如凌乱需要重新做。 3. 口号结束后，讲师需跟学生一起有节奏的鼓掌（必须站在讲台前）。 4. 学生报告出勤情况时，讲师要面带微笑，并以肯定的眼神看着班长（报告出勤的同学）。 |

1. **脚本切片二：视频切片1 视频切片2**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型方法的调用**  **180s** | **1、泛型方法使用时的注意事项**  讲师：接下来，看一下如何调用泛型方法。  讲师：调用泛型方法和普通方法有什么区别？  学生：各抒己见。  讲师：泛型方法在调用时指定类型形参的具体类型。  讲师：此处会把change方法中的类型形参替换成String来处理。  讲师：其中，类型实参是可以省略的。  讲师：如果省略，会隐式推断类型实参的类型。  讲师：调用泛型方法有两种方式，显式指定类型实参和隐式推断类型实参。  讲师：显式指定后，当传其它类型数据时会报错。  讲师：下面来研究隐式推断。 | 1. 强调调用方法时指定类型实参。 |
| **隐式推断的几种情况**  **300s** | **2、隐式推断**  讲师：如果整个参数列表和返回值中只有一处使用类型形参。  讲师：比如下面代码，在方法声明中只有一处使用类型形参。  讲师：func(new String[3], 1, 4)可见T是String，没有歧义。  讲师：返回值类型使用了T时，调用时类型实参和变量的类型相同。  讲师：如果整个参数列表和返回值中有多处使用类型形参。  讲师：add(T a,T b)中都使用了T类型。  讲师：如果a和b的类型相同，类型实参就是该类型。  讲师：add(“aab”,new Object())是不同类型，最后类型实参是什么？  讲师：有一个类型是其它类型的父类时，实际类型是此父类类型。  讲师：add(1,1.5)，其中1和1.5不是同类型，不具备父子关系。  讲师：会找到两者的父类Number。  讲师：此时，会限制T的类型上限为Number。  讲师：T可以是Number或其子类类型。  讲师：其中?会在后面讲解。  讲师：如果在泛型方法中有泛型类型的参数。  讲师：copy方法种有ArrayUtil类型的参数。  讲师：当传入ArrayUtil对象时，直接指定了T的实际类型是String。  讲师：所以，后面使用T时，只能是String类型。  讲师：下面的代码是错误的，不能传Integer。  讲师：使用隐式推断不容易代码阅读，存在安全隐患，不建议使用。 | 1. 带领学生共同分析出答案。 |
| **泛型类种的泛型方法**  **240s** | **3、类型参数的作用域**  讲师：接下来看一个比较特别的情况，泛型类中的泛型方法。  讲师：声明泛型类，类中声明两个泛型方法。  讲师：第一个泛型方法声明的类型形参名和类中声明的名称相同。  讲师：第二个泛型方法声明了一个类型形参E。  讲师：在main方法声明了泛型类的对象，指定类型实参为Integer。  讲师：v1的类型是什么？  学生：各抒己见。  讲师：没有指定方法调用时的类型实参，通过隐式推断得出是Integer。  讲师：v2的类型是什么？  学生：各抒己见。  讲师：如果方法中的T是声明对象时指定的，那么此处就是错误的。  讲师：回想一下，如果成员变量和局部变量同名时，谁的优先级高？  学生：局部变量。【参考回答】  讲师：如果泛型方法中声明的类型形参和泛型类中声明的类型形参同名时，泛型方法中的类型形参起作用。  讲师：所以v2的类型是Double。  讲师：第三条显式指定了为String，v3的类型是什么？  学生：String。【参考回答】  讲师：通过“哈哈”推断出E的类型是String。  讲师：但是返回值类型是T。  讲师：T是泛型类中声明的，在创建对象时指定具体类型，为Integer类型。 | 1. 强调泛型方法的作用域。 |
| 讲师：通过上述学习，可以总结出下面内容。  讲师：泛型方法可以出现在普通类和接口中，也可以出现在泛型类型中。  讲师：泛型类型中，类型参数的作用域是整个类或接口。  讲师：泛型方法中，类型参数的作用域是整个方法  讲师：当产生冲突时，方法中的类型参数会覆盖类或接口中的类型参数 | 1. 可以带领学习共同得出结论。 |

1. **脚本切片三：视频切片3**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型方法的概念**  **180s** | **1、泛型方法的定义**  讲师：掌握了泛型方法的使用后，就可以给泛型方法下定义了。  讲师：泛型方法的英文是Generic Method。  讲师：什么是泛型方法呢？  讲师：泛型方法的定义和泛型类、泛型接口应该是相似的。  讲师：泛型方法的属是什么？  学生：方法。【参考回答】  讲师：泛型方法和普通方法有什么本质区别？  学生：使用类型形参定义。【参考回答】  讲师：泛型方法是使用类型形参定义的方法。 | 1. 引导学生共同的出泛型方法的概念。 2. 引导学生共同总结泛型方法的内涵。 |
| **2、泛型方法的内涵和注意事项**  讲师：泛型方法的内涵是使用类型形参定义。  讲师：在使用泛型方法是需要注意什么？  学生：各抒己见。  讲师：第一，调用方法时指定泛型实参。  讲师：第二，优先于泛型类型中的泛型参数。  讲师：第三，静态方法中使用泛型必须使用泛型方法。 | 1. 引导学生共同得出内涵和注意事项。 |

1. **脚本切片四：视频切片4**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **练习：自定义泛型方法**  **600s** | 讲师：现在，通过上机来练习泛型方法的创建和使用。  讲师：通过定义泛型方法，实现将DrawPoint转换为数组的功能。  讲师：需要注意，数组元素类型和表示点的数据类型一致。  讲师：实现思路如下。  讲师：第一步，创建工具类。  讲师：第二步，创建转换方法，实现将DrawPoint转换数组的功能。  讲师：第三步，测试泛型方法。  讲师：给大家10分钟时间完成。 | 1. 讲解训练点和需求。 2. 说明练习时间。 |

1. **脚本切片五：视频切片5 视频切片6**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型的概念**  **240s** | **1、泛型的定义**  讲师：Generic是一般的，泛化的。  讲师：泛型就是Generics。  讲师：通过泛型是不是实现了在同一份代码上操作多种数据类型？  学生：是。【参考回答】  讲师：他的属是什么？  学生：各抒己见。  讲师：通过反射实现了在同一份代码上操作多种数据类型。  讲师：所以，它是一种机制。  讲师：泛型是通过参数化类型来实现在同一份代码上操作多种数据类型的机制。 | 1. 引导学生共同得出泛型的定义。 |
| **2、泛型的内涵和使用场景**  讲师：泛型的内涵有哪些呢？  讲师：第一，通过参数化类型来实现。  讲师：第二，使用/调用时传入具体类型。  讲师：第三，编译时检查，类型安全。  讲师：泛型主要有以下几方面应用。  讲师：应用类型形参泛型类型，比如泛型类和泛型接口。  讲师：泛型应用于方法形成泛型方法。  讲师：静态方法使用泛型时，必须使用泛型方法。  讲师：其中泛型类型是使用类型形参创建的引用数据类型。 | 1. 回顾前面所学，与学生共同得出泛型的内涵和使用场景。 |

1. **脚本切片六：视频切片7**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型的规则限制**  **300s** | **1、泛型的五个规则限制**  讲师：泛型在使用时需要注意什么呢？  讲师：来看一下泛型规则限制。  讲师：第一，泛型的类型实参只能是引用数据类型，不能是基本数据类型。  讲师：尖括号中可以放String、Integer等引用数据类型。  讲师：可以是int类型吗？  学生：不可以。【参考回答】  讲师：第二，同一种泛型可以对应多个版本。  讲师：当传递不同的类型实参时对应泛型的不同版本。  讲师：比如，ArrayUtil<Object>和ArrayUtil<Persion>是两个版本。  讲师：他们都对应一个class文件。  讲师：通过反编译文件发现，编译后变成Object类型。  讲师：泛型存在于编译阶段，编译后就进行了类型擦出。  讲师：使用instanceof判断类型时，不能带<>。  讲师：第三，泛型的类型参数可以有多个，用逗号分隔。  讲师：比如Map接口，以及自定义的Gen类。  讲师：当创建实例时，指定对应个数的类型实参即可。  讲师：第四，类型参数可以使用通配符。  讲师：第五，类型参数可以使用extends、super语句。  讲师：extends在继承时学习过，super在继承中代表父类对象。  讲师：稍后，我们会学习这两个关键字在泛型中的使用。 | 1. 通过回顾讲解泛型的规则限制 2. 通过后两条规则引出通配符。 |

1. **脚本切片七：脚本切片8**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **为什么使用泛型通配符**  **240s** | **1、通过案例引出泛型通配符**  讲师：来看以下案例。  讲师：有一个箱子，箱子中有礼物，需要打开箱子查看礼物。  讲师：需要设计几个类？  学生：各抒己见。  讲师：定义Gift，重写toString()方法，可以显示礼物信息。  讲师：创建两个子类。  讲师：BirthdayGift表示生日礼物，LoverGift表示情人节礼物。  讲师：定义泛型类Box，可以存放各种类型的礼物，并显示礼物信息。  讲师：使用工具类负责打开箱子。  讲师：现在，通过代码来模拟以下。  讲师：open()方法拥有Box<Gift>类型的参数。  讲师：Box<BirthdayGift>能传递给box方法吗？  学生：不能。【参考回答】  讲师：泛型的每个版本在编译阶段看成多个不同的类型。  讲师：使用Box<Gift>作为参数类型时，无法传递泛型类型的其它版本。  讲师：怎么去优化这个版本呢？ | 1. 通过案例讲解为什么使用泛型通配符。 2. 强调泛型类型的每个版本在编译阶段看成不同的类型。 |
| **2、使用重载解决时代码繁琐**  讲师：不难发现，Box<Gift>和Box<BirthdayGift>不具备父子关系。  讲师：如何增加代码的通用性，可以传递各个版本的对象？  学生：各抒己见。  讲师：可以使用重载解决。  讲师：但是代码繁琐，重用性不强。  讲师：有没有一个符号可以匹配泛型类Box的所有版本呢？  讲师：泛型中使用“泛型类型<?>”匹配泛型类型的所有版本。  讲师：其中“?”是一种通配符。 | 1. 说明使用重载可以解决问题。 2. 强调使用重载时代码繁琐。 |

1. **教学过程脚本设计——第四课时**
2. **脚本设计思路**



1. **脚本切片设计思路**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **内容** | **目标** | **设计亮点** | **时间** |
| 1 | 通配符的概念 | 1. 理解通配符的概念。 | 1. 通过“属+种差”下定义。 | 5min |
| 2 | 无界通配符 | 1. 理解无界通配符的概念。 | 1. 通过“属+种差”下定义。 | 3min |
| 3 | 为什么使用有界通配符 | 1. 理解为什么使用无界通配符。 | 1. 通过对比引出新知识。 | 2min |
| 4 | 有界通配符的使用 | 1. 掌握固定上边界通配符。 2. 掌握固定下边界通配符。 | 1. 通过类比讲解。 | 8min |
| 5 | 泛型通配符的概念 | 1. 理解泛型通配符的概念。 | 1. 利用“属+种差”下定义。 | 5min |
| 6 | 泛型类型的有界类型形参 | 1. 掌握泛型类的有界类型形参。 | 1. 通过对比学习新知识。 | 2min |
| 7 | 学员操作-泛型通配符 | 1. 掌握泛型通配符的使用。 | 1. 边讲边练。 | 10min |
| 8 | 总结 | 1. 梳理本单元知识点。 | 1. 通过提问和概念图进行总结。 | 5min |

1. **脚本切片一：视频切片1**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **通配符的概念**  **300s** | **1、常见的通配符**  讲师：什么是通配符呢？  讲师：现在来研究一下通配符。  讲师：针对不同的语言，通配符是有区别的。  讲师：shell语言中使用\*匹配0或多个字符，使用?匹配任何一个字符。  讲师：使用?匹配任意一个字符。  讲师：正则表达式中的通配符和shell略有不同。  讲师：可以使用正则表达式验证某个字符串是否符合某个规范。  讲师：SQL语言中，也有通配符  讲师：从表格中可以发现，通配符就是为了替代一到多个真正字符的吧？  学生：是。【参考回答】 | 1. 简单介绍常用的通配符。 2. 共同得出通配符是为了替代一到多个真正字符的。 |
| **2、通配符的定义**  讲师：通配符的英文是Wildcard。  讲师：通配符的属是什么？  学生：各抒己见。  讲师：它代表一个或多个真实的字符，所以它很特殊。  讲师：是不是一种特殊字符？  学生：是。【参考回答】  讲师：通配符的作用是什么？  学生：各抒己见。  讲师：是不是代替一个或多个真正字符？  学生：是。【参考回答】  讲师：通配符是代替一个或多个真正字符的特殊字符。 | 1. 引导学生共同得出通配符的定义。 |
| **3、通配符的内涵和外延**  讲师：通配符的内涵是什么？  讲师：第一，可以代替一个或多个真正字符的特殊字符。  讲师：第二，是一类键盘字符，是可以通过键盘输入的。  讲师：通配符种类很多。  讲师：包含shell通配符、正则表达式通配符、SQL通配符、泛型通配符。 | 1. 通过回顾，和学生共同总结出通配符的内涵和外延。 |

1. **脚本切片二：视频切片2**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **无边界通配符**  **180s** | 讲师：泛型中只有“？“这一个通配符。  讲师：”?”匹配任意引用数据类型，表示的是泛型实参。  讲师：比如，Box<?>可以匹配Box<Gift>、Box<LoverGift>等Box类型的所有版本。  讲师：”?”匹配的也是一到多个字符，多个字符表示的是类型名。  讲师：现在就可以把open方法的参数类型改为Box<?>类型了。  讲师：在测试类种，可以传递Box各个版本的实例。  讲师：”?”没有限制要传递的类型，只要是引用数据类型即可。  讲师：所以叫做无边界通配符。  讲师：无边界通配符是以?表示任意引用数据类型的通配符。 | 1. 强调泛型中只有一个通配符。 2. 强调?表示的是泛型实参。 |

1. **脚本切片三：视频切片3**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **为什么使用有界通配符**  **120s** | 讲师：使用无边界通配符时，可以传递泛型类型的所有版本。  讲师：他的类型是无法限制的，存在隐患。  讲师：题目中要求打开的是礼物，不符合题意。  讲师：Box<Gift> box = new Box<Gift>(new Gift());  讲师：当把box传入open方法时，看一下执行过程。  讲师：box.setData(null)，类型实参必然是引用数据类型。  讲师：所有引用数据类型都有null值，此处可以赋值。  讲师：box.setData(“礼物”)会报错吗？  讲师：当创建box对象时，指定T为Gift类型，但是赋值为String类型。  讲师：？可以匹配任意类型，不能确定data的类型。  讲师：不能进行赋值操作。  讲师：如果能限制？的取值范围，问题就解决了。  讲师：如何限定？匹配类型的范围呢？ | 1. 强调存在隐患。 |

1. **脚本切片四：视频切片4 视频切片5**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **有界通配符的使用**  **480s** | **1、使用固定上边界通配符**  讲师：来看一下？所能匹配类型的范围。  讲师：使用extends限制上边界。  讲师：比如，Box<? extends 类型>，类型指定为Gift时。  讲师：当传递类型实参时，只能是Gift及其子类。  讲师：当open方法传递Box<LoverGift>对象时可以吗？  学生：可以。【参考回答】  讲师：LoverGift是Gift的子类，所以没有问题。  讲师：可以在方法中读取box的信息吗？  学生：可以。【参考回答】  讲师：读取时，获得的类型是什么？  讲师：限制的类型为指定类型及其子类，所以都可以向上转型为父类。  讲师：访问时按照父类类型进行访问。  讲师：box.setData(new BirthdayGift())有问题吗？  讲师：open方法指定可以传递Gift及其子类的对象。  讲师：所以，存在是BirthdayGift对象的场景。  讲师：在实际调用时，指定的类型实参是LoverGift。  讲师：所以data的实际类型是LoverGift，此处赋值出错。  讲师：为了保证代码的安全，在使用上边界时，可以使用父类类型访问，但是不能修改值，null值除外。  讲师：所以，两句setData都是错误的。  讲师：Box<String> bs = new Box<String>(new String());  讲师：String不是Gift类型或其子类，编译错误。  讲师：固定上边界通配符是使用extends限定类型实参的类型上限的通配符。 | 1. 强调限制上限后，类型实参可以是限定类型以及其子类。 2. 通过案例讲解上边界通配符。 |
| **2、使用固定下边界通配符**  讲师：能不能限制类型实参的下边界呢？  讲师：使用super限定下边界。  讲师：比如Box<? super LoverGift>，可以传递LoverGift及其父类。  讲师：可以传递Object类型吗？  学生：可以。【参考回答】  讲师：类型实参可以是super关键字后指定的类型或其父类。  讲师：box.setData(new LoverGift())有问题吗？  学生：没有。【参考回答】  讲师：当类型实参传递LoverGift及其父类时，data可以赋值为LoverGift的实例。  讲师：在赋值时，直接赋值或自动向上转型。  讲师：如果赋值为Object对象，可以吗？  学生：不行。【参考回答】  讲师：如果类型实参传递的是LoverGift时，赋值Object实例会报错。  讲师：所以，赋值时只能赋值成super关键字后类型的对象。  讲师：类型实参传递String肯定不可以。  讲师：类型实参传Birthday可以吗？  学生：不可以。【参考回答】  讲师：因为都不是LoverGift类型或其父类。  讲师：因为类型不确定，不能访问。  讲师：限制上边界时，可以访问不能修改。  讲师：限制下边界时，可以修改不能访问。  讲师：固定下边界通配符是使用super限定类型实参的类型下限的通配符。 | 1. 强调泛型实参可以是super关键字后指定的类型或其父类。 2. 通过案例讲解下边界通配符。 |

1. **脚本切片五：视频切片6 视频切片7**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **什么是泛型通配符**  **180s** | **1、泛型通配符的概念**  讲师：掌握了泛型通配符的使用，就可以给泛型通配符下定义了。  讲师：泛型通配符的属是什么？  学生：通配符。【参考回答】  讲师：其它通配符表示一到多个真实的字符，泛型通配符呢？  学生：各抒己见。  讲师：泛型通配符是在泛型中用来表示任意类型实参的通配符。  讲师：泛型通配符有哪些内涵呢？  讲师：可以代表任意类型，匹配泛型类型的多个版本。  讲师：泛型通配符是作为类型实参使用的。  讲师：声明对象时使用。  讲师：类型实参在什么时候使用？  学生：各抒己见。  讲师：泛型通配符是在实例化对象时使用的，一般用于方法的参数。  讲师：泛型通配符分为无边界通配符、固定上边界通配符、固定下边界通配符。 | 1. 引导学生共同得出泛型通配符的定义。 2. 引导学生共同得出泛型通配符的内涵和外延。 |
| **泛型通配符的区别**  **120s** | 讲师：看一下三种泛型通配符的区别。  讲师：无边界通配符可以接受任意的泛型类型版本。  讲师：固定上边界通配符声明的对象是只读的。  讲师：按照extends关键字后的类型进行访问。  讲师：固定下边界通配符声明的对象是只写的。  讲师：按照super关键字后的类型进行赋值操作。  讲师：如果既想读又想写，就不要使用泛型通配符。 | 1. 对比三种泛型通配符。 2. 讲解三种泛型统配符的特点。 |

1. **脚本切片六：视频切片8**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **泛型类型的有界类型形参**  **120s** | 讲师：创建泛型类、泛型接口、泛型方法时，能不能设定类型形参的类型边界？  学生：各抒己见。  讲师：可以通过<T extends 类型>类型形参的上边界。  讲师：比如，class Box<T extends Gift>{}。  讲师：可以在创建对象时传入Gift及其子类类型的类型实参。  讲师：在创建接口时也可以设置上边界。  讲师：创建方法时也可以设置上边界。  讲师：注意，不可以设置下边界。  讲师：当设置上边界后，创建对象、实现接口、调用方法时，类型实参只能是指定类型或其子类。 | 1. 强调不可以设置下边界。 |
| 讲师：T和?都可以使用extends限定上边界。  讲师：他们有什么区别呢？  学生：各抒己见。  讲师：？在实例化对象时作为类型实参使用，一般作为方法的参数。  讲师：T在创建泛型类、泛型接口、泛型方法时作为类型形参使用。  讲师：?可以是无边界的，也可以固定上边界或下边界。  讲师：T只能是无边界或固定上边界。 | 1. 强调T是作为类型形参使用的。 2. 强调?是作为类型实参使用的。 |

1. **脚本切片七：视频切片9**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **练习：泛型通配符** | 讲师：通过学习泛型通配符后，来做一个上机练习。  讲师：主要训练泛型统配符的使用。  讲师：实现如下功能。  讲师：增加DrawPoint的管理类，可以存放多个点的信息。  讲师：使用泛型通配符创建add方法，可以增加点信息。  讲师：实现步骤如下。  讲师：创建DrawPointManager类，定义成员变量存在数据。  讲师：创建方法进行相应操作。  讲师：测试功能。 | 1. 讲解训练点和需求。 2. 说明练习时间。 |

1. **脚本切片八**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **总结**  **300s** | 讲师本次课相关知识点，通过提问方式进行总结。 | 1. 通过大量提问，巩固学生学习成果。 2. 提问范围尽量覆盖本单元重难点问题和相关概念。 |
| **概念图**  **300s** | 通过概念之间的联系讲解概念图，通过提问的方式引导学生完成概念图的描述。 | 1. 强调概念图的重要性。 2. 强调本单元的重要概念。 |