**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. 理解泛型的常用规则
5. 理解自定义泛型类、泛型接口与泛型方法
6. 理解泛型通配符的使用
7. 理解有界通配符和有界类型形参的使用
8. 理解泛型的继承规则
9. **能力目标**
10. 会创建自定义泛型类
11. 会创建自定义泛型接口
12. 会创建自定义泛型方法
13. 会使用泛型通配符
14. 会使用有界通配符
15. 会使用有界类型形参
16. 会继承泛型类和实现泛型接口
17. **速度目标**
18. 10分钟内，根据需求完成自定义泛型类与泛型接口
19. 5分钟内，根据需求完成自定义泛型方法
20. **课时分配**
21. **第一课时**
22. **知识目标**

* 理解泛型的概念
* 理解泛型的应用场景
* 理解泛型的初步使用

1. **能力目标**

* 理解泛型的使用特点

1. **第二课时**
2. **知识目标**

* 理解泛型类的概念
* 理解泛型类的语法规则
* 理解泛型接口的概念
* 理解泛型接口的语法规则

1. **能力目标**

* 会创建自定义泛型类
* 会创建自定义泛型接口

1. **第三课时**
2. **知识目标**

* 理解泛型方法的概念
* 理解泛型的常用规则

1. **能力目标**

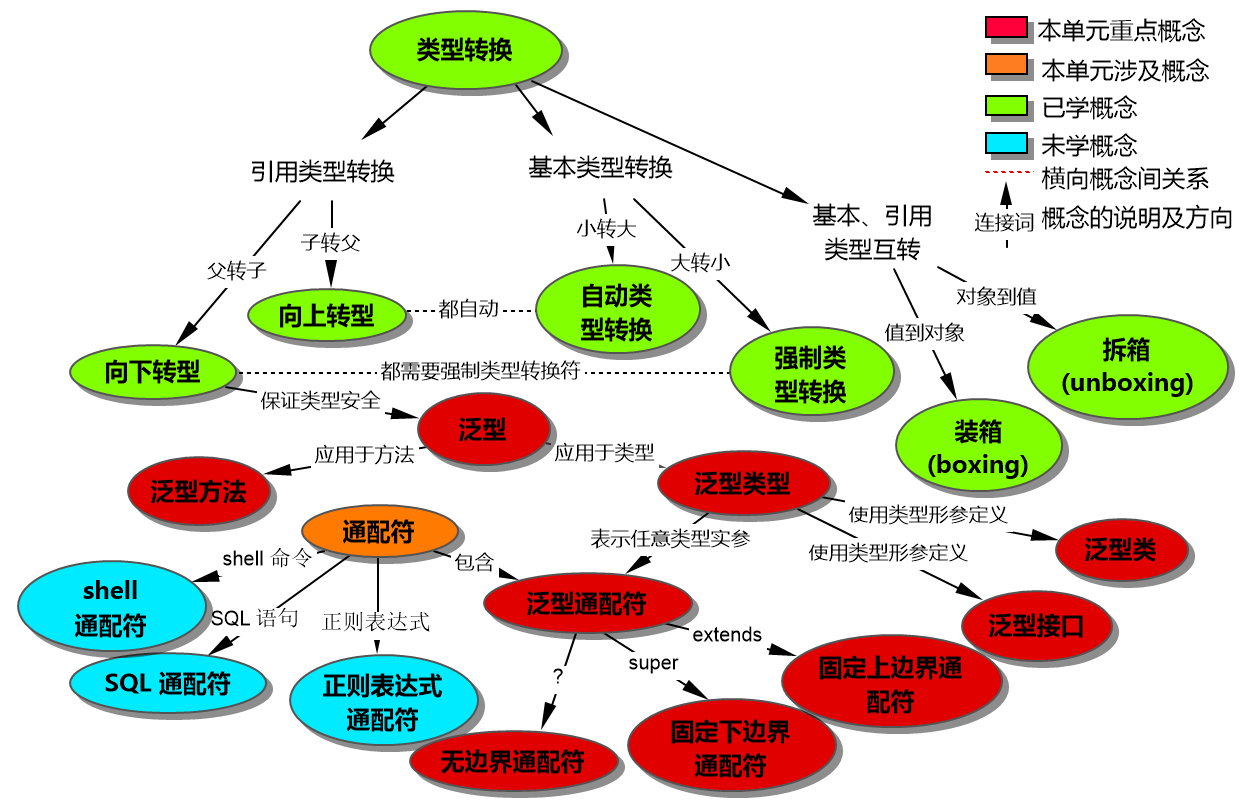
* 会创建自定义泛型方法
* 能够熟练运用泛型的常用规则

1. **第四课时**
2. **知识目标**

* 理解通配符与泛型通配符的概念
* 理解右边界的泛型通配符
* 理解类型形参中的有界

1. **能力目标**

* 会使用无界与有界泛型通配符
* 掌握类型形参中的边界限定

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** | **时间（min）** |
| 0 | 课堂礼仪 | 提升学生气势，精神饱满。 | 1 |
| 1 | 知识回顾 | 巩固之前的知识。 | 5 |
| 2 | 本单元学习目标 | 明确学习目标，做到学习有的放矢。 | 5 |
| 3 | 为什么使用泛型 | 通过案例使学生认识到学习泛型的必要性。 | 5 |
| 4 | 什么是泛型 | 理解泛型的概念 | 5 |
| 5 | 泛型可以用在哪里 | 理解泛型的三种应用场景 | 5 |
| 6 | 泛型的初步使用 | 了解泛型的初步使用 | 5 |
| 7 | 使用泛型的注意事项 | 归纳泛型的使用特点 | 5 |
| 8 | 课堂小结 | 及时进行课堂总结 | 5 |

1. **脚本切片零**

|  |  |  |
| --- | --- | --- |
| **课堂模块** | **教学内容/台词** | **教学标准/素材** |
| **课堂礼仪** | **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. **脚本切片一：视频切片1**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 知识回顾 | **讲师：**我们先来回顾上一单元所学习的枚举。  **讲师：**同学们请思考两个问题，欢迎举手回答：  （1）什么是枚举？（枚举是使用enum声明的、由一组预定义的本类型常量组成的引用数据类型。）  **讲师：**我们知道，枚举默认继承了java.lang.Enum类，是一种特殊的类，不同于普通类，因此在Java的引用数据类型中，将枚举也作为一种相对独立的引用数据类型。  **讲师：**那么，枚举与普通类的区别有哪些呢？  **讲师：**第一，默认继承的父类不同，枚举默认继承java.lang.Enum类，普通类默认继承Object类。  **讲师：**第二，声明时所用的关键字不同，声明枚举时用enum关键字，声明普通类时用class关键字。  **讲师：**第三，枚举在创建后不可以被继承，普通类一般可以被继承。  **讲师：**第四，构造器的访问权限修饰符不同，枚举构造器只能用private权限修饰符，普通类的构造器可以根据需要选用4种权限修饰符（private，默认default，protected，public）中的任何一种。  **讲师：**第五，实例化方式不同，枚举实例在枚举体内定义，系统自动为枚举实例名添加public static final予以修饰，普通类一般用new创建实例。  **讲师：**第六，枚举项及其常量名可以作为switch...case...表达式中的匹配项，普通类的对象一般不可以。  **讲师：**关于枚举，我们先复习到这里，我们接下来学习泛型。 |

1. **脚本切片二：视频切片2**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 本单元学习目标 | **讲师：**本单元的学习目标有两个，一个是技能目标，另一个是理论目标。  技能目标是同学们学完本单元，能够做出来什么？这是一项重要的工作能力指标。  **讲师：**理论目标是同学们学完本单元，是否能够用语言和文字清晰完整地描述概念图里的概念？这项能力对于培养同学们今后面试，以及工作中的专业交流，研究力的提升非常重要。  **讲师：**本单元的技能目标是能够完成梦想成真小游戏。这个小游戏是什么意思呢？  **讲师：**主要是说，同学们在编程中设计一个方法，再自定义一些动物类型（如马，羊，牛，兔，象，长颈鹿等），无论给这个方法传入您喜欢的何种类型动物，该方法都会相应地返回您所想要的小动物，从而使您“梦想成真”。  **讲师：**这个案例程序执行后，控制台打印如图所示。  **讲师：**相信在学完本单元后，同学们都可以运用泛型来实现这个功能。  **讲师：**我们接下来看本单元的理论目标。 |
| **讲师：**这是本单元的概念图。浅绿色部分是我们已经学习过的概念，红色部分是本单元要学习的重点概念，橙色部分是本单元涉及到的概念，浅蓝色部分为本单元以外，且尚未学习过的概念。  **讲师：**在我们过去学习的类型转型中，基本数据类型与其包装类型可以实现自动装箱和拆箱，这不是我们本单元要重点了解的部分。  **讲师：**我们重点看下类型转换中的自动类型转换与强制类型转换，以及向上转型和向下转型。  **讲师：**子类型的对象转换为父类型对象时，被称为向上转型，这时进行自动类型转换。  **讲师：**父类型对象转换为子类型对象时，被称为向下转型，这时需要进行强制类型转换。如果强制类型转换时，类型不一致，会造成类型转换异常。同学们在编程时，或许会在控制台中看到过类似ClassCastException的异常，这通常是由于类型转换时的不一致引起的。  **讲师：**为了保证类型转换时的程序安全性，尽可能避免不必要的异常，使得我们写出的程序更健壮，因此我们在本单元学习泛型。  **讲师：**我们会在本单元的学习中将这些红色的概念一个一个地攻克。 |

1. **脚本切片三：视频切片3**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 为什么使用泛型 | **讲师：**我们先来看Java程序从源代码编辑到编译，再到运行的过程：    **讲师：**总的来说，Java程序从源代码到运行，主要经历编译期和运行时两个阶段。同学们平时用Eclipse，MyEclipse等编程工具写代码时，有时保存代码会发现代码中会有打红叉的情况，这说明程序在编译期就产生了错误，无法经过编译期，因此也就到不了运行时。  **讲师：**我们在判断一个对象的类型时，通常分为两个阶段，一个是编译期类型，另一个是运行时类型。在判断对象的类型方面，有一个规则：  编译期看左侧，运行时看右侧。  **讲师：**比如，在一个赋值表达式中：  Object val=new String(“123”);  **讲师：**String是Object的子类，将String对象赋值给父类变量，属于向上转型，会进行自动类型转换。当一个变量被赋值一个对象时，这个变量也会被指代为对象，即被称为对象。  **讲师：**对于对象val而言，它的编译期类型是赋值表达式左侧声明的Object类型；它的运行时类型就是赋值表达式右侧的类型，即String类型。运行时类型是对象的实际类型，编译期类型是对象的形式类型。  **讲师：**如果一个对象的编译期类型和运行时类型不一致，那么要把对象转换为实际类型的对象，就需要进行强制类型转换。若在强制转换时类型不一致，就会发生类型转换异常，使得程序中止运行。  **讲师：**我们接下来以代码为例。 |
| **讲师：**我们先声明一个名称为NoGeneric的类，并在其内部声明一个类型为Object、名称为val的私有成员属性。类的成员属性也称为成员变量。为了供外部访问类的私有成员属性，通常会在类内再声明相应的getter方法。  **讲师：**同学们请留意，在Java语言中，Object是所有类的父类。凡是父类出现的地方，都可以用相应的子类代替。所以，成员变量val的实际类型可以是Object或者其子类。  **讲师：**我们再通过该类的公有构造方法传参来为成员属性val赋值。类的构造方法也称为构造器。当前类的构造器传的是Object类型参数，所以，构造器在实际调用中可以传入Object或者其子类的对象。  **讲师：**我们再声明一个返回值类型为Object的公有成员方法getVal()，这是类的一个getter方法，用来获取私有成员属性val。  **讲师：**请再看测试类的一段代码。我们在程序执行的入口函数main方法内，创建NoGeneric对象，并在构造器中传入String字符串对象“123”。  **讲师：**我们再声明Object类型的变量val，该变量指向getVal方法返回的对象。为了称谓方便，我们将变量val称为对象val。  **讲师：**结合声明NoGeneric类在声明时的代码，getVal方法返回的实际上是一个String对象。所以，对象val的编译期类型为Object，运行时的类型为String。由于对象val的编译期类型和运行时类型不一致，要想获得对象val的实际类型，就要进行强制类型转换。  **讲师：**在强制类型转换时，若目标类型与对象的实际类型一致，则不会发生类型转换异常；否则就会产生类型转换异常。  **讲师：**如果我们在编程中未能区分清楚val的实际类型，用其他类型，如Integer去会产生什么后果？比如，用Integer去强制转换val？  **讲师：**如图所示，程序执行时，控制台打印出异常ClassCastException，意思是说“类型转换异常”，程序也随之中止运行。 |
| **讲师：**在JDK1.5之前，为了实现代码复用，我们通常将Object作为通用类，在使用时用相应的具体类型进行强制转换。  **讲师：**比如，我们刚才声明的NoGeneric类，就将Object作为成员属性的类型，以及成员方法的返回值类型。由于Object是所有类的父类，所以使用时用具体子类进行强制转换就可以了。这种做法在一定程度上的确实现了写一份代码，应付多种需要的情形。  **讲师：**然而，这样做的代价是，对象在编译期与运行时易产生类型不一致，从而引发类型转换异常。  **讲师：**那么，有没有一种更好的方法，既能够提高代码的重用性，即复用性，又可以避免编译期和运行时类型不一致，从而在根源上规避类型转换异常呢？有。  **讲师：**从JDK1.5开始，Java引入了泛型机制，用以解决类型转换异常问题，增强类型转换时的程序安全性，同时提高代码重用性，提升程序执行效率。  **讲师：**我们接下来看什么是泛型。 |

1. **脚本切片四：视频切片4**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 什么是泛型 | **讲师：**我们在学习泛型的概念时，先大致了解泛型怎么使用。  **讲师：**以泛型在类中的使用为例，其语法格式为：  class 类名称<T>{ }  **讲师：**需要注意的是，标识符T是形式上的类型，在使用时传入具体的类型即可。  **讲师：**我们在类体内可以将T作为成员类型，包括方法的返回值类型。  **讲师：**接下来请看泛型使用的简单案例。 |
| **讲师：**这段代码与之前相似，不同的是，我们在类名称右边多了一对尖括号，在尖括号里写入了一个标识符T。  **讲师：**然后在类体内，我们将原先所用的Object类都换成了标识符T。那么，这样使用有什么效果呢？  **讲师：**请看测试类代码。  **讲师：**我们仍是在main方法中创建类的对象，在创建对象时，在尖括号中传入具体的类型String，并通过构造器传参传入字符串“123”.  **讲师：**然后，我们调用对象obj的getVal方法，发现该方法返回值的类型在编译期就自动转换为String类型。 |
| **讲师：**同学们思考下：如果创建类的对象时，我们传入Integer或者其他类型的对象，那么，标识符T是不是在编译期就自动转换为相应的类型。答案是对的，同学们可以在课下通过编程工具练习验证下。这种情况是否说明，标识符T很像一个参数化的类型，只是做下形式上的声明，在使用时传入具体类型，标识符T就自动转换为相对应的类型？这样的话，我们设计好一个类，可以运用于多种场景，是不是提高了代码的复用性？  **讲师：**然后在控制台打印，程序能够正常执行。  **讲师：**可见，类在使用泛型机制后，使得对象的编译期类型与运行时类型一致，所以无需进行强制类型转换，因而规避了类型转换异常。 |
| **讲师：**基于上述代码分析，我们来看泛型的概念。  **讲师：**泛型的英文单词为genericity。我们前面声明类的名称时用Generic，是Genericity的形容词。  **讲师：**我们接下来用种差加属的方式推导泛型的概念。  **讲师：**被定义项是泛型，我们前面说“泛型机制”，该机制可以实现将类型参数化，也就是将类做个形式上的声明，使用时传入具体类型，那么实际类型随之在编译期就进行了确定。  **讲师：**同学们再思考下，我们还学习过Java中的哪些机制？继承，对。继承主要体现了子类对父类信息的复用。  **讲师：**我们再来比较泛型与继承有哪些差异？一是参数化类型，二是可以提高代码复用性，三是在编译期强制进行类型检查，也就是说，如果我们在编译期所传值与要求的实际类型不相符，那么在编译期就会报错。  **讲师：**所以，泛型是通过参数化类型提高代码复用性，并在编译期强制进行类型检查的机制。  **讲师：**我们再来看泛型的外延，泛型可以分别应用于类，接口和方法。 |

1. **脚本切片五：视频切片5**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 泛型可以用在哪里 | **讲师：**那么，泛型机制可以用在哪里呢？  **讲师：**泛型应用于类形成泛型类，泛型应用于接口形成泛型接口，泛型应用于方法形成泛型方法。  **讲师：**Java中提供了内置的一系列泛型类和泛型接口，以及泛型方法的声明格式。 |
| **讲师：**在Java中，类和接口都属于引用数据类型，所以泛型类和泛型接口也可以统称为泛型类型。 |

1. **脚本切片六：视频切片6**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 使用泛型的注意事项 | **讲师：**那么，我们在使用泛型时需要注意哪些事项呢？  **讲师：**第一，在类型声明时对类型进行参数化。  **讲师：**第二，在使用时传入具体的类型，这时，泛型参数自动转换为具体的类型。  **讲师：**第三，程序在编译期会强制进行类型检查，从而确保类型一致，避免类型转换异常，使得程序运行更加安全。 |

1. **脚本切片七：视频切片7**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 课堂小结 | **讲师：**我们接下来总体回顾本节课的内容。  **讲师：**我们先是回顾了上一个单元学习的枚举，在夯实旧知识的基础上，学习新的内容。  **讲师：**我们学习了泛型的概念，知道了泛型是通过参数化类型提高代码复用性，并在编译期强制进行类型检查的机制。  **讲师：**另外，我们还学习了泛型的使用场景。比如，泛型应用于类形成泛型类，应用于接口形成泛型接口，应用于方法形成泛型方法。  **讲师：**我们下一节将深入学习泛型类和泛型接口。 |
| **讲师：**这是本节课的课堂拓展内容，介绍了泛型标识符的含义。  **讲师：**这页内容供同学们参考。 |

1. **脚本切片八：视频切片8**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 课堂拓展：  泛型标识符的含义 | **讲师：**这是本节课的课堂拓展内容，介绍了泛型标识符的含义。  **讲师：**这页内容供同学们参考。 |

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** | **时间** |
| 1 | 学习目标 | 明确学习目标 | 10min |
| 2 | 什么是泛型 | 明确概念 | 9min |
| 3 | 如何创建泛型类 | 学会创建泛型类 |  |
| 4 | 如何创建泛型类的对象 | 学会创建泛型类的对象 | 2min |
| 5 | 子类如何继承泛型类 | 掌握子类对泛型类的继承 | 10min |
| 6 | 自定义泛型类的案例 | 通过案例掌握自定义泛型类 | 2min |
| 7 | 使用泛型类的注意事项 | 了解泛型类使用中的注意事项 | 4min |
| 8 | 什么是泛型接口 | 掌握泛型接口的概念 |  |
| 9 | 如何使用泛型接口 | 掌握泛型接口的使用 |  |
| 10 | 使用泛型接口的注意事项 | 了解泛型接口使用中的注意事项 |  |
| 11 | 课堂编程 | 通过课堂编程巩固所学知识 |  |
| 12 | 课堂小结 | 总结本堂课的内容 |  |

1. **脚本切片一 ：视频切片1**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 学习目标 | **讲师：**我们上节课学习了泛型的概念和使用场景。  **讲师：**本节课将学习泛型在类和接口中的使用。  **讲师：**请问，我们为什么学习泛型类和泛型接口呢？  **讲师：**通过声明类型形参，我们在使用时可以传入多种类型实参，从而提高代码复用性。 |

1. **脚本切片二：视频切片2**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 什么是泛型 | **讲师：**泛型类的单词是Generic Class。  **讲师：**被定义项是泛型类，请结合之前的泛型初步使用，使用了泛型的类，本质上仍是一个类，那么在类种，将没有使用泛型机制的类就通称为非泛型类。二者的差异为是否使用类型形参进行了定义。  **讲师：**所以，泛型类是使用类型形参定义的类。 |

1. **脚本切片三：视频切片3**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 如何创建泛型类 | **讲师：**泛型类的语法格式为：  class 类名<类型形参列表>{ }  **讲师：**这里说的“形参列表”，是指可以传入多个类型形参，中间以英文逗号割开即可。  **讲师：**其中，类型形参在创建泛型类时使用，需要声明在尖括号中，类型形参是形式上的参数，如果传入多个类型形参，则以英文逗号隔开。  **讲师：**类型形参不仅可以用在泛型中，也可以用在后面要学习的泛型接口和泛型方法中。  **讲师：**我们结合一段代码来看。创建一个泛型类，在类名称右边添加一对尖括号，尖括号里声明类型形参T。这就创建了一个简单的泛型类。 |

1. **脚本切片四：视频切片4**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 如何创建泛型类的对象 | **讲师：**我们创建了泛型类，那么，应该如何创建泛型类的对象呢？  **讲师：**我们先看创建泛型类的对象的语法格式：  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：**其中，类型实参是在创建泛型类的对象时指定，只能是引用数据类型；如果说类型形参是声明时传入的形式上的类型，那么，类型实参则是使用时传入的实际类型。  **讲师：**当然，类型实参除了可以运用于泛型类，还可以运用于泛型接口，泛型方法。  **讲师：**我们来看创建泛型类的对象的代码。  **讲师：**如图所示，创建泛型类的对象时，传入类型实参。 |

1. **脚本切片五：视频切片5**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 子类如何继承泛型类 | **讲师：**我们已经学习了创建泛型类及其对象，那么，如何实现子类继承泛型类呢？  **讲师：**在子类继承泛型类时，可以通过两种方式实现。一种是为作为父类的泛型类传入类型形参，另一种是为作为父类的泛型类传入类型实参。 |

1. **脚本切片六：视频切片6**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 自定义泛型类的案例 | **讲师：**我们接下来再看一个创建泛型类及其对象的案例。  **讲师：**创建一个泛型类，并在它的main方法中实现创建泛型类及其对象。然后调用对象的成员方法，会发现泛型类在声明时传入的类型形参自动转换为使用时传入的类型实参。 |

1. **脚本切片七：视频切片7**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 使用泛型类的注意事项 | **讲师：**总的来说，使用泛型类时的注意事项有：  **讲师：**第一，实例化泛型类时，需指定具体类型；  **讲师：**第二，编译时进行类型检查；  **讲师：**第三，当类型实参为空时，默认是Object类型。 |

1. **脚本切片八：视频切片8**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 什么是泛型接口 | **讲师：**泛型接口的英文单词是Generic Interface。  **讲师：**被定义项是泛型接口，我们参考刚才学习的泛型类，使用了泛型机制的接口仍是接口，同样，未使用泛型机制的接口就称为非泛型接口。二者的差异是是否使用类型形参定义。  **讲师：**所以，泛型接口是使用类型形参定义的接口。 |

1. **脚本切片九：视频切片9**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 如何使用泛型接口 | **讲师：**声明泛型接口的语法格式为：  interface 接口名<类型形参列表>{ }  **讲师：**我们来看代码，声明了两个泛型接口，在接口名称的右边添加了尖括号，在尖括号中添加类型形参T。  **讲师：**在接口体内，既可以将类型形参T作为方法传入的形参类型，也可以作为方法的返回值类型。 |
| **讲师：**我们知道，声明接口主要是为了供子类实现。  **讲师：**那么，子类可以如何实现泛型接口呢？  **讲师：**一种方式是，为泛型接口传入类型形参。语法格式和代码示例请看课件：  class 类名<类型形参列表> implements 接口名<类型形参列表>{ }  **讲师：**另一种方式是，为泛型接口传入类型实参。语法格式和代码示例请看课件：  **讲师：**语法格式：class 类名 implements 接口名<类型实参列表>{ } |

1. **脚本切片十：视频切片10**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 使用泛型接口的注意事项 | **讲师：**那么，我们在使用泛型接口时需要注意哪些事项呢？  **讲师：**第一，在实现接口时要指定类型实参。  **讲师：**第二，程序在编译期会进行类型检查。  **讲师：**第三，当类型实参为空时，默认是Object类型。 |

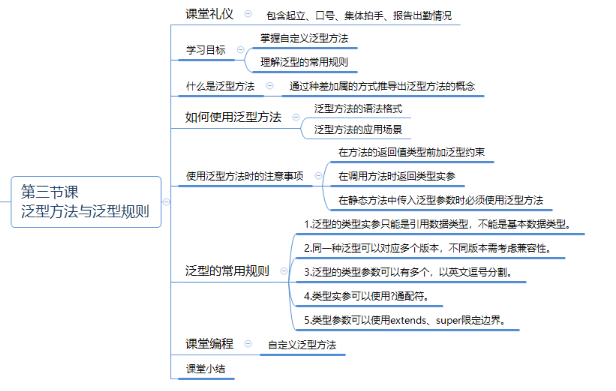
1. **脚本切片十一：视频切片11**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 课堂编程 | **讲师：**本节课的课堂编程是创建泛型类和泛型接口。  **讲师：**同学们请参考创建泛型类和泛型接口的语法格式，并分别创建泛型类的对象，以及实现泛型接口，并通过程序执行入口函数main方法来执行程序。  **讲师：**练习时间10分钟，请同学们打开编程工具练习。 |

1. **脚本切片十二：视频切片12**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 课堂小结 | **讲师：**现在总结本节课的内容。  **讲师：**我们先是复习了上节课学习的泛型概念及其使用场景，然后分别学习泛型类和泛型接口，并通过课堂编程予以巩固。  **讲师：**我们下一节将深入学习自定义泛型方法和泛型的常用规则。 |

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** | **时间** |
| 1 | 学习目标 | 明确学习目标 | 1min |
| 2 | 什么是泛型方法 | 掌握泛型方法的概念 | 12min |
| 3 | 如何使用泛型方法 | 掌握泛型方法的使用 | 3min |
| 4 | 泛型方法的使用特点 | 掌握泛型方法与非泛型方法的区别 | 10min |
| 5 | 使用泛型方法的注意事项 | 理解泛型方法的注意事项 | 4min |
| 6 | 泛型的常用规则 | 了解泛型的常用规则 | 5min |
| 7 | 课堂编程 | 通过编程巩固所学知识 | 4min |
| 8 | 课堂小结 | 总结本节课的内容 |  |

1. **脚本切片一 视频切片1**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 学习目标 | **讲师：**我们上节课学习了自定义泛型类和泛型接口。  **讲师：**本节课将学习自定义泛型方法，以及泛型的常用规则。  **讲师：**那么，我们为什么学习泛型方法呢？这是由于同泛型类和泛型接口相似，泛型方法声明一次，可以衍生出多个版本，从而提高代码的重用性。  **讲师：**那么，为什么学习泛型的常用规则呢？  **讲师：**这是因为，只有掌握了规则，才能更好地指导我们的实践。 |

1. **脚本切片二：视频切片2**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 什么是泛型方法 | **讲师：**泛型方法的单词为Generic Function。  **讲师：**被定义项是泛型方法，与泛型类和泛型接口相似，使用了泛型机制的方法仍是方法，那么未使用泛型机制的方法就是非泛型方法。二者的差异是是否使用类型形参定义。  **讲师：**因此，泛型方法是使用类型形参定义的方法。 |

1. **脚本切片三：视频切片3**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 如何使用泛型方法 | **讲师：**那么，如何使用泛型方法呢？  **讲师：**我们先来看声明泛型方法时的语法格式：  权限修饰符 <类型形参列表> 返回值类型 方法名(参数列表) { }  **讲师：**同学们可以看到，与普通方法不同，泛型方法在返回值类型前添加泛型约束。  **讲师：**我们结合一段代码来看。  **讲师：**在代码中，我们声明了一个泛型方法，在方法的返回值类型前添加了一对尖括号，在尖括号里声明了类型形参T，并在方法中传入类型T的参数。  **讲师：**在调用泛型方法时，我们传入不同的具体类型对象，类型形参T就会自动转换为相应的具体类型。 |
| **讲师：**那么，泛型方法通常用在什么场景中呢？  **讲师：**一种场景是在需要提高代码重用性时，可以使用泛型方法。  **讲师：**另一种场景是当在静态方法中传入泛型参数时，必须使用泛型方法。  **讲师：**比如，在下面的代码中，声明了一个泛型类型，作为普通成员方法，尽管传入了泛型参数，但是不必是泛型方法；然而，对于静态成员方法来说，如果传入了泛型参数，这个静态成员方法就必须是泛型方法，否则会在编译期报错。 |

1. **脚本切片四：视频切片4**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 泛型方法的使用特点 | **讲师：**我们在来思考下，泛型方法与非泛型方法有哪些异同呢？  **讲师：**我们先来看普通的非泛型方法，在方法声明时传入形参值的类型，这样的话，在调用方法时，传入的实参值类型与形参值的类型一致即可。  **讲师：**我们再来看泛型方法，在方法声明时也声明了类型形参T，在方法调用时，传入什么类型的对象，参数类型就会自动转换为相应的类型。  **讲师：**可见，泛型方法与非泛型方法相同之处有：声明时传入形参，使用时传入实参。  **讲师：**二者不同之处有：非泛型方法在声明时限定实参值的类型，泛型方法在声明时限定实参类型的范围。xin |

1. **脚本切片五：视频切片5**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 使用泛型方法的注意事项 | **讲师：**那么，泛型方法在使用时的注意事项有哪些呢？  **讲师：**第一，在声明泛型方法时，要在返回值类型前添加泛型约束。  **讲师：**第二，在调用泛型方法时需要指定类型实参。  **讲师：**第三，在静态成员方法中传入泛型参数时，必须该方法必须是泛型方法。 |

1. **脚本切片六：视频切片6**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 泛型的常用规则 | **讲师：**那么，泛型有哪些常用的规则呢？  **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：**比如，在代码中，我们在创建泛型类的对象时，可以传入String类型，然而传入int基本数据类型时就会在编译期报语法错误。  **讲师：**第二，同一种泛型可以对应多个版本，不同版本需要考虑兼容性。  **讲师：**比如，我们使用同一个泛型类，在为其创建对象时，传入不同的类型实参，虽然这些创建的对象均属于同一个类型ArrayUtil，但却是这个类不同的版本，因此互不兼容。 |
| **讲师：**这里说的不兼容是什么意思呢？  **讲师：**比如，我们定义一个方法时，为其传入一个泛型类对象，具体如ArrayUtil<Person>的对象，那么，尽管同为ArrayUtil类，但是若给该方法传入ArrayUtil<Student>版本的对象，该方法就会在编译期报语法错误。 |
| **讲师：**第三，泛型的类型参数可以有多个，以英文逗号隔开。  **讲师：**比如，在代码中，创建泛型接口时传入两个类型形参K和V，中间以英文逗号隔开；创建泛型类时也传入两个类型形参，中间以英文逗号隔开；创建泛型类的对象时，将传入的两个类型实参用英文逗号隔开。  **讲师：**第四，类型实参可以用？通配符。  **讲师：**这是代码示例。关于通配符，我们会在下一节继续学习。  **讲师：**第五，类型参数可以使用extends、super关键字限定边界。  **讲师：**这是代码示例。那么，extends和super分别是什么意思，能起到什么作用？我们也会在下一节深入学习。 |

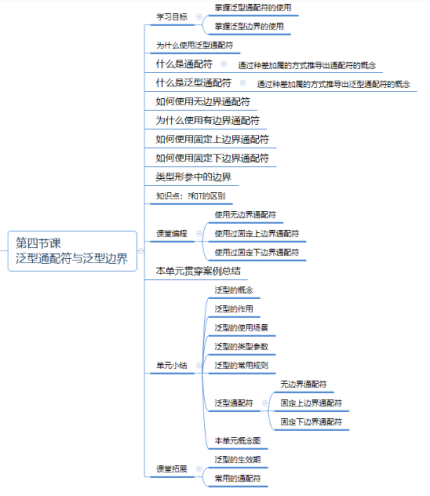
1. **脚本切片七：脚本切片7**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 课堂编程 | **讲师：**本节课的课堂编程要求同学们创建自定义泛型方法，并且能够调用泛型方法。  **讲师：**练习时间5分钟，请同学们打开编程工具练习。 |

1. **脚本切片八：脚本切片8**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 课堂小结 | **讲师：**我们做下课堂回顾。  **讲师：**本节课先是回顾了泛型类与泛型接口。  **讲师：**在此基础上，我们学习了泛型方法，然后又学习了泛型中常用的规则，在这里面，我们还涉及到了通配符与边界问题。  **讲师：**我们下一节将学习泛型通配符与泛型边界。 |

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** | **时间** |
| 1 | 学习目标 | 明确学习目标 | 5min |
| 2 | 为什么使用泛型通配符 | 理解为什么使用无界通配符。 | 3min |
| 3 | 什么是通配符 | 理解通配符的概念 | 2min |
| 4 | 什么是泛型通配符 | 理解泛型通配符的概念 | 8min |
| 5 | 如何使用无边界通配符 | 掌握无边界通配符的使用 | 5min |
| 6 | 为什么使用有边界通配符 | 理解有边界通配符的用途 | 2min |
| 7 | 如何使用固定上边界通配符 | 掌握固定上边界通配符的使用 | 10min |
| 8 | 如何使用固定下边界通配符 | 掌握固定下边界通配符的使用 |  |
| 9 | 类型形参中的边界 | 掌握类型形参中的边界使用 | 5min |
| 10 | 知识点：?和T的区别 | 理解?和T的区别 |  |
| 11 | 课堂编程 | 通过课堂编程巩固所学知识 |  |
| 12 | 本单元贯穿案例总结 | 总结本单元贯穿案例 |  |
| 13 | 单元总结 | 总结本单元的所学知识 |  |
| 14 | 课堂拓展：泛型的生效期；  常用的通配符 | 了解泛型的生效期，以及常用的通配符 |  |

1. **脚本切片一：视频切片1**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 学习目标 | **讲师：**我们上节课学习了泛型方法，以及泛型的常用规则。  **讲师：**我们本节课将学习泛型通配符与泛型边界的使用。  **讲师：**那么，我们为什么要使用泛型通配符与泛型边界呢？让我们在接下来的学习中解答这个问题。 |

1. **脚本切片二：视频切片2**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 为什么使用泛型通配符 | **讲师：**我们声明一个Gift父类，再声明该类的两个子类。  **讲师：**其中，Gift表示礼物；一个子类BirthdayGift表示生日礼物类，另一个子类LoverGift表示情人节礼物。  **讲师：**然后，我们再声明一个泛型类Box，表示礼品盒，意为里面可以放礼物。  **讲师：**接着，我们再定义一个工具类BoxUtil，用来操作Box类。在BoxUtil类内，我们定义一个open方法，在方法中传入形参类型Box<Gift>，我们知道，这是Box类的一个版本。  **讲师：**在BoxUtil类的main方法中，当我们在调用open方法时，传入Box的另一个版本Box<BirthdayGift>的对象，发现在编译期检查出异常，表现为代码中打红叉。  **讲师：**我们在上一节学习泛型的规则时，其中有一条“同一种泛型可以对应多个版本，不同版本需考虑兼容性”，若未考虑兼容性的情况下，同种类型的不同版本会互不兼容。可以说，我们当前的案例就生动地印证了这个规则。  **讲师：**那么，出现这种情况的原因是什么呢？ |
| **讲师：**我们知道，尽管BirthdayGift是Gift的子类，但是Box<BirthdayGift>并非Box<Gift>的子类，更不与Box<Gift>等价，二者是泛型类Box的两个不同版本。  **讲师：**那么，如何避免泛型类的多个版本不兼容的问题呢？多写几个重载的open方法？  **讲师：**这种做法显然使得代码繁琐，而且代码重用性也不够强。  **讲师：**有没有一种符号可以兼容Box类的所有版本呢？有。  **讲师：**比如，我们在声明open方法时，可以将所传参数类型设置为Box<?>。这个英文？号就是泛型通配符。  **讲师：**那么，什么是通配符，什么又是泛型通配符？ |

1. **脚本切片三：视频切片3**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 什么是通配符 | **讲师：**通配符的英文单词是wildcard。  **讲师：**被定义项是通配符，结合刚才的案例可知，通配符有？等特殊字符，所以，通配符是一种特殊字符。  **讲师：**那么，同为特殊字符，除了通配符还有什么呢？加减乘除等数学符号是一种特殊字符；千克kg，兆比特Mb等单位符号。  **讲师：**通配符与数学符号，单位符号等有什么差异呢？通配符可以代替一个或多个真正字符。  **讲师：**所以，通配符的定义是：通配符是代替一个或多个真正字符的特殊字符。  **讲师：**通配符的外延范畴有shell通配符（我们通常用shell访问操作系统），正则表达式通配符（正则表达式通常用来对字符串是否符合某个规则进行匹配），SQL通配符（通常在操作数据库的语句中使用），还有泛型通配符。  **讲师：**我们接下来重点学习泛型通配符。 |

1. **脚本切片四：视频切片4**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 什么是泛型通配符 | **讲师：**那么，什么是泛型通配符呢？  **讲师：**被定义项是泛型通配符，根据刚才的学习可知，泛型通配符是一种通配符。其他通配符还有shell通配符，正则表达式通配符，SQL通配符等。  **讲师：**泛型通配符与其他通配符有什么区别呢？第一，代表任意类型；第二，作为类型实参；第三，实例化对象时使用。。  **讲师：**因此，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：**根据泛型通配符是否对范围边界进行限定，以及限定方向，泛型通配符的外延范畴有无边界通配符，固定上边界通配符和固定下边界通配符。 |

1. **脚本切片五：视频切片5**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 如何使用无边界通配符 | **讲师：**无边界通配符是仅以？表示的泛型通配符。？默认匹配Object类，也就是可以匹配任意引用数据类型。  **讲师：**我们结合前面的案例代码可知，Gift，BirthdayGift、LoverGift均是Object的子类。所以，Box<?>可以兼容Box<Gift>、Box<BirthdayGift>等Box版本。  **讲师：**我们在代码中可以看到，在声明open方法时传入Box<?>类型的参数，我们在main方法中创建类的对象，并调用open方法时，发现使用open方法可以调用Box的所有版本。  **讲师：**这说明Box<?>兼容了泛型类Box的所有版本。 |

1. **脚本切片六：视频切片6**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 为什么使用有边界通配符 | **讲师：**然而，泛型通配符?默认匹配Object，可以兼容泛型类Box的任意版本，甚至包括一些不和业务逻辑的版本。**讲师：**比如，在代码中，我们调用open方法时，传入Box<Integer>的对象，这也是泛型类Box的一个版本。我们在业务逻辑本来是在Box礼品盒中装礼品，然而这时却传入了一个非礼物的类型Integer。  **讲师：**尽管程序在编译期和运行时均正常，然而运行结果却并不符合我们的业务逻辑需求。  **讲师：**那么，是否可以限定泛型通配符?的匹配范围呢？可以为其限定边界。  **讲师：**根据对泛型通配符?向上和向下限定边界，可以分为固定上边界通配符与固定下边界通配符。 |

1. **脚本切片七：视频切片7**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 如何使用固定上边界通配符 | **讲师：**固定上边界通配符是使用extends限定类型实参上界的通配符。语法格式为：<? extends 上界父类>  也就是说，在使用泛型通配符?时，用extends向上限定边界，实参类型可以是上界父类自身或者其子类。  **讲师：**比如，我们在声明open方法时，传入的形参类型是Box<? extends Gift>，这时，我们再在main方法体中调用open方法，在传入泛型类Box的多个版本时，就要求尖括号<>里的实际类型必须是Gift类或者其子类，否则在编译期会报错。这样的话，就向上约束了泛型类Box的版本范围，增强了对泛型运用的控制。 |

1. **脚本切片八：视频切片8**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 如何使用固定下边界通配符 | **讲师：**固定下边节通配符则是使用super限定实参类型下界的通配符。  **讲师：**它的语法格式是：  <? super 下界子类>  **讲师：**意思是说，使用super向下限定边界，实参类型可以是下界子类自身或者其父类。  **讲师：**比如，我们在open中传入Box<? super Birthday>类型的参数，意味着所传实参类型在泛型类Box的版本中，尖括号<>中应该传入BirthdayGift或者其父类，其他的Box版本则不被兼容。  **讲师：**比如，在main中，我们调用open方法时，传入Box<LoverGift>版本类型的对象，则程序在编译期报错，这是由于LoverGift并非BirthdayGift父类，所以该版本的Box类型不被兼容。  **讲师：**总的来说，我们在open方法中传入泛型类型Box时，Box在此前已经被声明，所以Box采用无边界、固定上边界或者固定下边界的通配符时，相当于对对泛型类Box的类型实参做了范围限定。那么，泛型类Box在声明时，是否可以设置边界呢？也是可以的。 |

1. **脚本切片九：视频切片9**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 类型形参中的边界 | **讲师：**结合泛型在类，接口与方法中的运用，我们均可以在声明时就通过extends限制类型形参的上边界。  具体请看代码。  **讲师：**需要注意的是，类型形参可以限制上边界，却不能限制下边界，否则会报语法错误。 |

1. **脚本切片十：视频切片10**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 知识点：?和T的区别 | **讲师：**我们在前面多次使用了泛型通配符?和标识符T，那么二者的应用场景和范围限定有什么异同呢？  **讲师：**?是一种通配符，在泛型中传入类型实参时使用；在范围限定方面，可以用于无边界，固定上边界，以及固定下边界。  **讲师：**T是一种表示泛型参数的标识符，在创建泛型类、泛型接口和泛型方法时可以作为类型形参；在范围限定方面，可以用于无边界，固定上边界。 |

1. **脚本切片十一：视频切片11**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 课堂编程 | **讲师：**本节课的课堂编程是，同学们创建一个父类及其两个子类，在创建一个泛型类，以及一个测试泛型的工具类，然后在工具类中分别测试无边界通配符，固定上边界通配符和固定下边界通配符。 |

1. **脚本切片十二：视频切片12**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 本单元贯穿案例总结 | **讲师：**这是我们本单元之处的贯穿案例。  **讲师：**先设计一个动物类Animal作为父类，然后分别声明马Horse，羊Sheep，牛Cattle等作为Animal的子类。  **讲师：**在测试类中定义一个泛型方法getAnimal，所传类型形参通过extends限定上界父类为Animal，该方法所传参数为类型形参T的对象，并返回所传入的对象。  **讲师：**然后在测试类的main方法中，分别创建喜欢的小动物对象，然后传入泛型方法getAnimal。这时会惊奇地发现：您传入哪种喜欢的小动物，就会返回相应的小动物从而“梦想成真”。 |

1. **脚本切片十三：视频切片13**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 单元总结 | **讲师：**本节课主要学习了泛型通配符与泛型边界。  **讲师：**泛型通配符用英文?表示，可以通过extends限定上边界，通过super限定下边界。  **讲师：**我们再整体回顾本单元的学习内容。  **讲师：**首先，我们学习了泛型的概念。泛型是通过参数化类型提高代码重用性，并在编译期强制进行类型检查的机制。  **讲师：**泛型的作用有：代码可重用性好；避免运行时类型转换异常，使得我们写出的程序更加安全；由于在程序编译期就确定了对象的实际类型，因此使得程序执行效率高。  **讲师：**泛型的使用场景有三种，分别可以用于类、接口和方法，在此基础上形成了泛型类，泛型接口和泛型方法。  **讲师：**泛型中的类型参数可以分为两种，一种是类型形参，用于创建泛型类，泛型接口和泛型方法；另一种是类型实参，用于创建泛型类的对象，实现泛型接口，以及调用泛型方法。 |
| **讲师：**泛型的常用规则有5个：  **讲师：**第一，类型实参只能用引用数据类型，不能用基本数据类型。  **讲师：**第二，同一种泛型可以有多个版本，不同版本需要考虑兼容性。  **讲师：**第三，泛型的类型参数可以有多个，中间以英文逗号隔开。  **讲师：**第四，类型实参可以使用?通配符。  **讲师：**第五，类型实参可以使用extends、super分别限定上下边界，类型形参只能用extends限定上边界。  在泛型通配符和泛型边界方面，仅使用通配符?表示无边界通配符；用extends配合通配符?使用则构成固定上边界通配符；用super配合通配符?使用则构成固定下边界通配符。 |
| **讲师：**我们接下来整体梳理本单元的概念图，也是检测我们的理论目标的完成情况。  **讲师：**我们来共同回忆：  **讲师：**什么是泛型？  **讲师：**泛型的使用场景有哪些？  **讲师：**泛型类型包括哪些？  **讲师：**什么是泛型通配符？  **讲师：**泛型通配符包括哪三种情况？ |

1. **脚本切片十四：视频切片14**

|  |  |
| --- | --- |
| **课堂模块** | **教学内容/台词** |
| 课堂拓展：泛型的生效期 | **讲师：**这两页是课堂拓展部分，一页介绍了泛型的生效期，另一页介绍了常用的通配符。  **讲师：**请同学们作为自习资料，有问题请问。 |
| 课堂拓展：常用的通配符 |