为什么使用泛型

- 回顾HashSet的例题

为什么使用泛型

- 在Java中增加泛型之前，泛型程序设计使用继承来实现的
- 坏处：
- - 需要强制转换
- - 可向集合中添加任意类型的对象，存在风险。

泛型的使用

- List<String> list=new ArrayList<String>();

- Java SE7及以后的版本中，构造方法中可以省略泛型类型。
- List<String> list=new ArrayList<>();

多态与泛型

- class Animal{}
- class Cat extends Animal{}
- List<Animal> list=new ArrayList<Cat>();
- //变量声明的类型必须匹配传递给实际对象的类型

多态与泛型

- 其他的错误例子：
- List<Object> list=new ArrayList<String>();
- List<Number> numbers=new ArrayList<Integer>();

泛型作为方法参数

- 案例需求：
- 定义一个抽象类Goods，包含抽象方法 sell()
- 分别定义类Book、Clothes和Shoes继承Goods，并实现sell()方法，输出一句话，如： sell books
- 定义一个商品销售类GoodsSeller，模拟销售，包括方法：
- public void sellGoods(List<Goods> goods),循环调用List对象的sell()方法。
- 测试

- <? extends Goods>
- extends后面的内容也可以是接口

- <? super Goods>

自定义泛型类

泛型方法

要注意的问题

- 泛型的类型参数只能是类类型，不能是基本数据类型
  比如：List<Integer> list=new ArrayList<Integer>();
- 泛型的原理就是参数化类型

总结

- 为什么使用泛型
- 泛型作为方法参数
- 自定义泛型类
- 自定义泛型方法