Java 核心技术(高阶)

第二章 Java泛型

第三节 泛型类型限定

华东师范大学 陈良育

泛型类型限定(1)

- 泛型
  - 编写的代码可以被很多不同类型的对象所重用
  - 特定场合下，需要对类型进行限定(使用某些特定方法)

```
public static <T extends Comparable> T getMin(T... a) {
    if (null == a || a.length <= 0) {
        return null;
    }
    T min = a[0];

    for (int i = 1; i < a.length; i++) {
        if (min.compareTo(a[i]) > 0) {
            min = a[i];
        }
    }
    return min;
}
```

泛型类型限定(2)

- 泛型限定
  - <T extends Comparable> 约定T必须是Comparable的子类
  - extends固定，后面可以多个，以&拼接，如<T extends Comparable & Serializable>
  - extends限定可以有多个接口，但只能一个类，且类必须排第一位
  - 逗号隔参数，<T extends File & Cloneable, U extends Serializable>

泛型继承原则(1)

- 泛型类之间的继承
  - Pair<S>和Pair<T>没有任何关系，无论S和T之间是什么关系

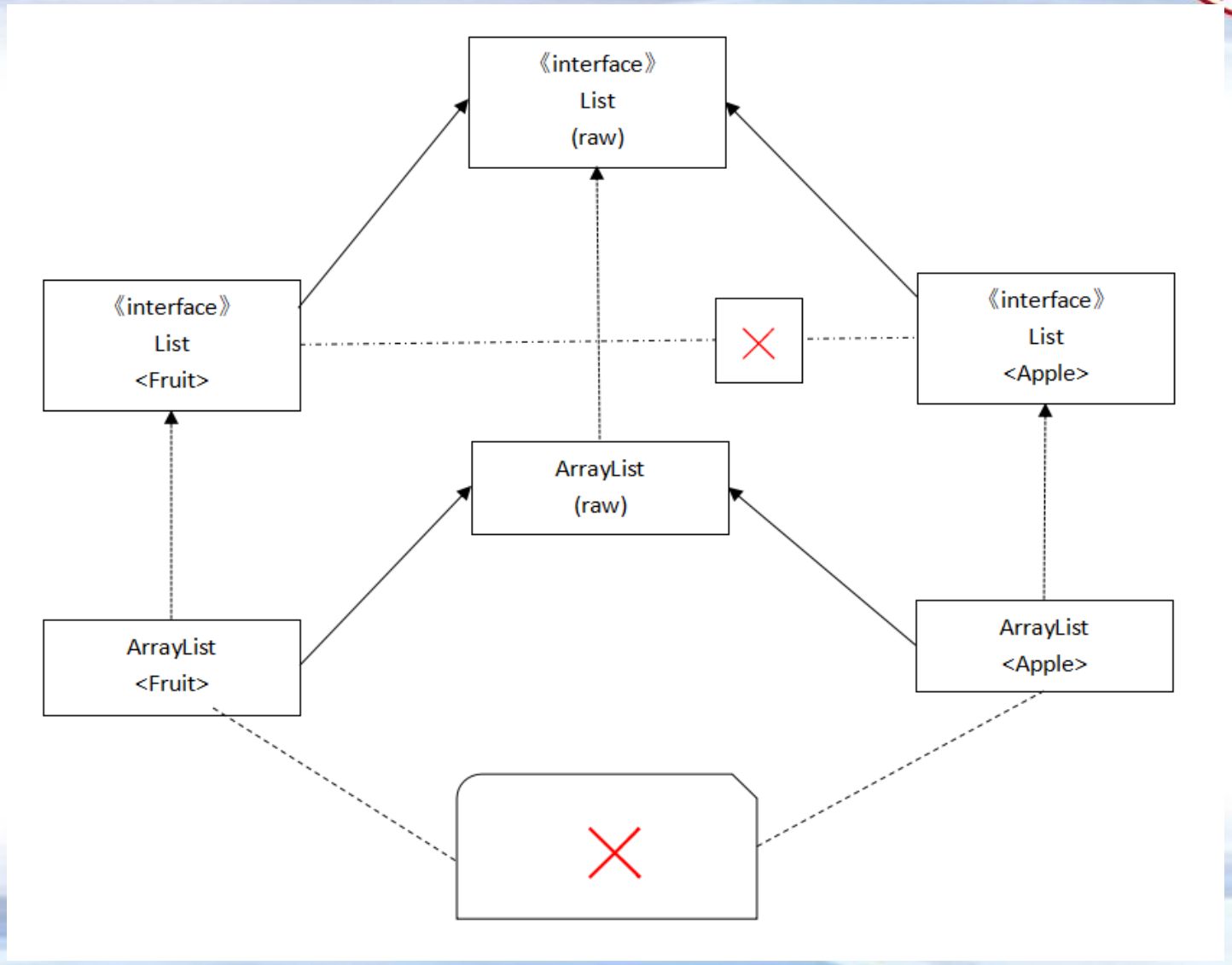
```
Pair<Apple> apples = new Pair<>(new Apple(3), new Apple(4));
Pair<Fruit> fruits = apples;
fruits.setFirst(new Orange(5));
```

  - 泛型类可以扩展或实现其他的类，如ArrayList<T> 实现List<T>

```
List<Orange> oranges = new ArrayList<Orange>();
```

泛型继承原则(2)

- 泛型类的继承



泛型通配符类型(1)

- 上限界定符，Pair<? extends S>
  - Pair能接收的参数类型，是S自身或子类
    - Pair<? extends Fruit>代表Pair<Fruit>, Pair<Apple>, Pair<Orange> 等

```
//Pair<Apple>  and Pair<Orange> 皆可
//Fruit的子类，都有getSize()方法
public void printFruits(Pair<? extends Fruit> fruits)
{
    Fruit first=fruits.getFirst();
    Fruit second=fruits.getSecond();

    System.out.println("The sizes of fruits are "
      + first.getSize() + "," + second.getSize());
}
```

泛型通配符类型(2)

- Pair<? extends S>，Pair能接收的参数类型，是S自身或子类
  - 只能get/不能set，编译器只能保证出来的类型，但不保证放入的对象是什么类型
    - Pair<? extends Fruit>代表Pair<Fruit>, Pair<Apple>, Pair<Orange> 等
    - ? extends Fruit getFirst(); //肯定可以转型到Fruit
    - void setFirst(? extends Fruit) //未知具体的类型，错误

```
Pair<Apple> apples = new Pair<>(new Apple(3), new Apple(4));
Pair<? extends Fruit> fruits = apples;
fruits.setFirst(new Orange(5)); //编译错误
```

泛型通配符类型(3)

- 下限界定符，Pair<? super S>，
  - Pair能接收的类型参数，是S的自身或超类
    - Pair<? super Apple> 代表Pair<Object>, Pair<Fruit>, Pair<Apple>等

```
Pair<? super Apple> fruits = new Pair<Fruit>();

fruits.setFirst(new Apple(5)); //Apple到超类的转型
fruits.setSecond(new GreenApple(5)); //GreenApple到超类的转型
fruits.setSecond(new Object());//无法转型为Apple的超类
```

泛型通配符类型(4)

- Pair<? super S>，Pair能接收的类型参数，是S本身或超类
  - 只能set/不能get，编译器保证放入的是S本身或超类，但不保证出来是什么具体类型
    - Pair<? super Apple> 代表Pair<Object>, Pair<Fruit>, Pair<Apple>等
    - void setFirst(? super Apple) //可以放入Apple及子类对象
    - ? super Apple　getFirst()　// 无法得知出来的对象类型，只能是Object

```
Pair<? super Apple> fruits = new Pair<Fruit>();

fruits.setFirst(new Apple(5)); //Apple到超类的转型
fruits.setSecond(new GreenApple(5)); //GreenApple到超类的转型

Fruit obj = fruits.getFirst(); //也未知其超类对象的方法，故报错
fruits.getFirst()).hello(); //也未知其超类对象的方法，故报错
```

泛型通配符类型(5)

- 泛型PECS原则
  - Producer Extends, Consumer Super
  - 要从泛型类读取类型T的数据，并且不能写入，可以使用？extends 通配符；(Producer Extends，泛型类是生产者，往外输出东西)
  - 如果要向泛型类写入类型T的数据，并且不需要读取，可以使用？super 通配符；(Consumer Super，泛型类是消费者，往内增加东西)
  - 如果既想写入又想读出，那就不用通配符

泛型通配符类型(6)

- 无限定类型的泛型
  - Pair<T>，原始类型
  - Pair<?>，无限定通配符，表示任意类型
    - 如Pair<Object>, Pair<apple>, Pair<Orange>
    - ? getFirst() //不确定出来是什么类型，只能赋值给Object
    - void setFirst() //无法放入任何对象，甚至是Object

总结

- 泛型类型限定
  - Pair<T> 和 Pair<S>没有关系
  - Pari<T extends Comparable & Serializable> 限定T的类型
  - PRES原则
    - Pair<? extends T>, 只能get，不能set，即只生产，不消费
    - Pair<? super T>, 只能set，不能get，即只消费，不生成

谢 谢！