Java常用工具_集合

课程概要

学习目标

理解集合的概念

能够说出Java集合体系结构及List、Set、Map三种集合的特点

能够往List集合对象中添加自定义对象元素

能够使用循环和迭代器遍历三种集合

理解泛型的概念和作用

集合简介

什么是集合?

简称集，是用来存储多个元素的容器

集合和数组的区别

元素类型

集合：引用类型（存储基本类型时自动装箱）

数组：基本类型、引用类型

元素个数

集合：不固定，可任意扩容

数组：固定，不能改变容量

集合的好处

不受容器大小限制，可以随时添加、删除元素

提供了大量操作元素的方法（判断、获取等）

Java的集合体系

单列集合（Collection）

List：

ArrayList

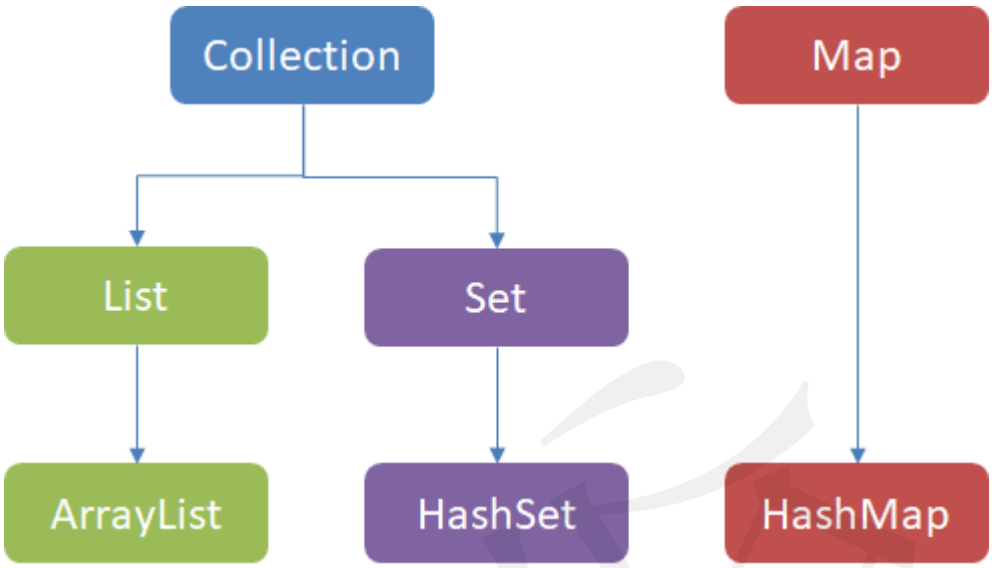
Set：

HashSet

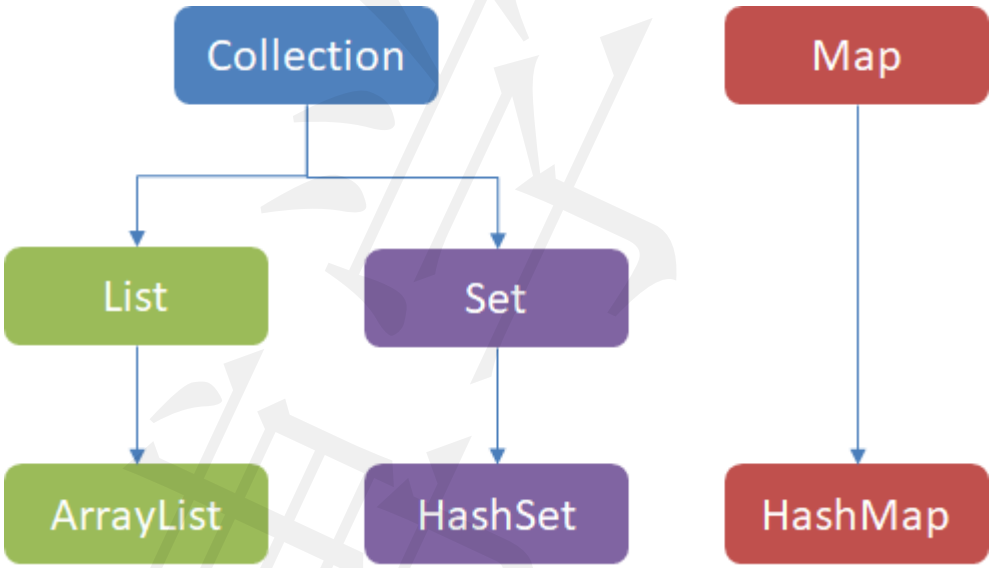
双列集合（Map： key，value ）

Map：

HashMap



List集合的特点和应用



List集合的特点

特点：

可重复、有序（存取顺序相同）

应用：

```
List list = new ArrayList();
```

案例：List集合的简单使用

需求：向List集合中添加三个元素，并遍历打印

分析：

1. 向集合中添加元素的方法为：add()
2. 遍历集合的方式：for循环
3. 获取集合中元素个数的方法：size()

步骤：

1. 创建集合对象：

```
List list = new ArrayList();
```

2. 分别创建三个Student对象
3. 使用add方法将Student对象添加到集合中
4. 使用for循环遍历集合并打印

增强for循环和迭代器

为什么需要增强for循环?

简化数组和集合的遍历

增强for循环的格式：

```
for(数据类型　变量名　:　数组或者集合对象)　{
    // 循环体，变量即元素
}
```

案例演示（使用增强for遍历List集合）

为什么需要迭代器?

对过程的重复，称为迭代。

迭代器是遍历Collection集合的通用方式，可以在对集合遍历的同时进行添加、删除等操作。

迭代器的常用方法

```
next():
    返回迭代的下一个元素对象
```

```
hasNext():
    如果仍有元素可以迭代，则返回true
```

3. 案例演示（使用迭代器遍历List集合）

泛型简介

什么是泛型?

即泛指任意类型，又叫参数化类型（ParameterizedType），对具体类型的使用起到辅助作用，类似于方法的参数。

集合类泛型的解释

表示该集合中存放指定类型的元素

案例演示（给List集合加上泛型Student）

```
List<Student> list = new ArrayList<>();
```

泛型的好处

类型安全 避免了类型转换

Collections工具类

Collections简介

针对集合进行操作的工具类。

成员方法

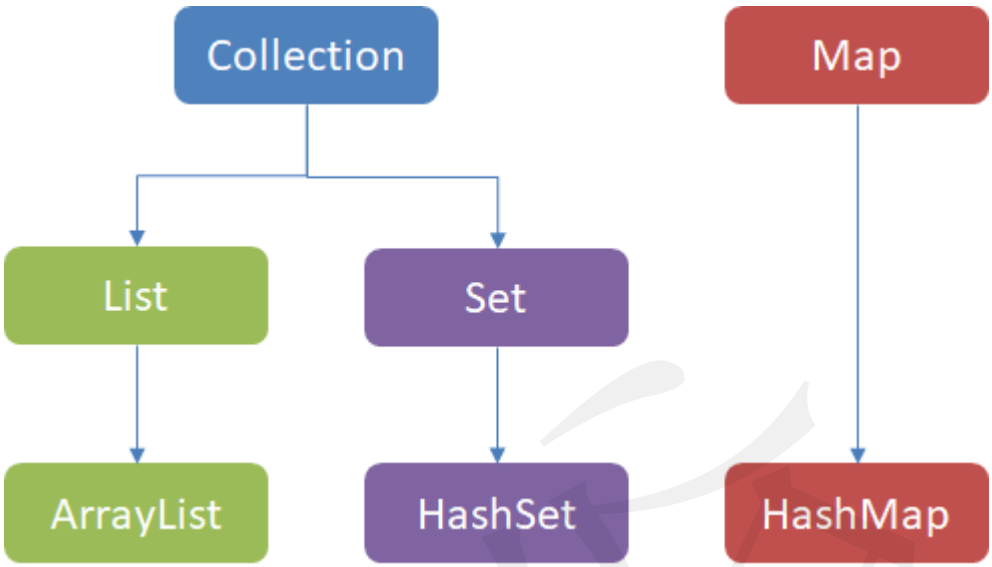
```
sort(List<T>,Comparator<T>)
    根据比较器规则对列表进行排序
```

```
max(Collection<T>)
    返回集合的最大元素
```

```
reverse(List<T>)
    反转List集合元素
```

```
shuffle(List<T>)
    根据自然顺序返回最大元素
```

Set集合的特点和应用



Set集合的特点

特点：

不可重复、无序

应用：

```
Set<T> set = new HashSet<>();
```

案例：Set集合的简单使用

需求：向Set集合中添加五个元素，并遍历打印

分析：

1. 向集合中添加元素的方法为：add()
2. 遍历集合的方式：迭代器
3. 获取集合中元素个数的方法：size()

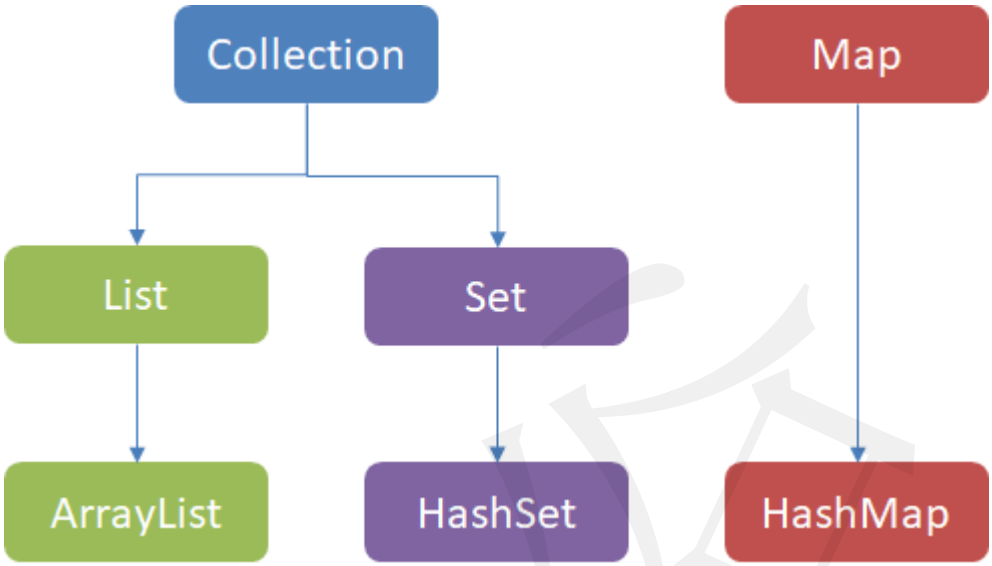
步骤：

1. 创建集合对象：

   ```
   Set<Student> set = new HashSet<>();
   ```

2. 分别创建五个Student对象
3. 使用add方法将Student对象添加到集合中
4. 使用迭代器遍历集合并打印，注意添加顺序和打印顺序是否相同

Map集合的特点和应用



Map集合的特点

特点：

双列集合，元素由键值对（Entry）构成： key -- value key不可以重复，value可以重复

应用：

```
Map<T1, T2> map = new HashMap<>();
```

案例：Map集合的简单使用

需求：向Map集合中添加三个元素，并遍历打印

分析：

1. 向map集合中添加元素的方法为：put()
2. 遍历集合的方式：

   获取所有的key：keySet()

   遍历keySet，通过key获取value：get()
3. 遍历keySet的方法：iterator()

步骤：

1. 创建集合对象：

   ```
   Map<Integer, Student> map = new HashMap<>();
   ```
2. 分别创建三个Student对象
3. 使用put方法将Student对象添加到集合中

4. 获取所有的key，并使用迭代器遍历
5. 通过key分别获取对应的value并打印

案例：模拟斗地主发牌

案例：模拟斗地主发牌

需求：

使用一个集合对象存储一副扑克牌，将所有扑克牌的顺序打乱，然后分发给用集合表示的三个玩家和底牌，并打印玩家和底牌的集合内容

步骤：

1. 买一副扑克牌 将花色和数字分别进行组合，生成所有的普通牌 手动添加“大王”、“小王”

```
    // 创建数字数组:
    String[] nums = {"3","4","5","6","7","8","9","10","J","Q","K","A","2"};
    // 创建花色数组:
    String[] colors = {"方片","梅花","红桃","黑桃"};
    // 将每一个花色分别和数字进行拼接
    colors[i].concat(nums[j]);
```

2. 洗牌

   使用Collections工具类的shuffle()方法打乱牌的顺序

```
    /*
    使用Collections工具类的shuffle()方法打乱牌堆集合的顺序
    分别创建三个玩家集合对象和底牌集合对象
    */
```

3. 发牌

   遍历牌堆，将每一张牌分发到三个玩家集合中 留三张作为底牌

```
    /*
    遍历牌堆的每一张牌，分别将索引对3取模的值为0，1，2的牌存放到三个玩家集合中，将最后三张存放到底牌集合中
    */
```

4. 看牌

   分别打印每个玩家集合的内容

```
    // 将玩家集合中的牌按自然顺序进行排序
    Collections.sort(List, Comparator);
    // 打印玩家集合中的牌
    // 重复上面的操作打印所有玩家的牌和底牌
```