JAVA 达摩班

Generics & Collection

Generics 概念及详解

1

理解泛型Generics

JDK 5引入泛型的概念，作为一个语言特性，它允许定义通配的，多样的类型和方法。

LinkedList<E>	泛型 generic type
E	形式参数 formal type
LinkedList<Integer>	参数化类型 parameterized types
Integer	实际参数 actual type

理解泛型Generics

为什么要引入泛型?

泛型提供编译时类型检测 (compile-time type checking)，防止ClassCastException异常发生，它是Collection中常见的异常。

泛型增加代码鲁棒性，它可以在编译时发现bug

理解泛型Generics

```
List list = new ArrayList();
list.add("abc");
list.add(new Integer(5));

for(Object obj : list){
   String str=(String) obj;
}
```

类型转换导致运行时ClassCastException

```
List<String> list1 = new ArrayList();
list1.add("abc");
list1.add(new Integer(5));

for(String str : list1){
   …
}
```

直接报编译时错误，不会发生类型转换，避免运行时异常

泛型类型

1. 泛型类 / 泛型接口 （Generic Class/Interface）

泛型定义在类或者接口上

2. 泛型方法 （Generic Method）

泛型定义在方法上，而不是整个类

3. 有界泛型参数（Generics Bounded Type Parameters）

用于限制对象类型数量

<T extends A & B & C>

<T extends Comparable<T>>

泛型类型

4. 泛型子类（Generic Classes and Subtype）

泛型子类通过extend父类或者implement接口实现
只要不改变类型参数，泛型子类的关系就会一直保留

假如定义了 interface MyList<E,T> extends List<E>{}
那么List<String>的子类可以是 MyList<String,Object>，也可以是MyList<String,Integer>

5. 泛型通配符 （Generics Wildcards）

问号（？）用于定义泛型通配符，代表未知类型
通配符可以用于参数类型，字段类型，变量或者返回值，但不能用于调用泛型方法或实例化类。

泛型类型

5.1. 通配符上限 （Generics Upper bound Wildcards）

List<? extend Number>，Interger，Double等子类型都是合法的

5.2. 通配符下限 （Generics Lower bound Wildcards）

List<? super Integer>，Number，Object等父类型都是合法的

5.3. 无界通配符下限 （Generics Unbounded Wildcards）

List<?> list，接受任何类型，等价于<? extends Object>

泛型类型

6. 泛型类型擦除（Generics Type Erasure）

泛型擦除是编译器的工作

它会移除所有泛型类型检查代码（字节码），加入必要的类型转换。

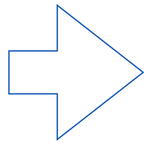
它会保证不增加新的类（泛型实参），从而提高运行时效率

```
public class Test<T extends Comparable<T>> {

    private T data;
    private Test<T> next;

    public Test(T d, Test<T> n) {
        this.data = d;
        this.next = n;
    }

    public T getData() { return this.data; }
}
```



```
public class Test {

    private Comparable data;
    private Test next;

    public Node(Comparable d, Test n) {
        this.data = d;
        this.next = n;
    }

    public Comparable getData() { return data; }
}
```

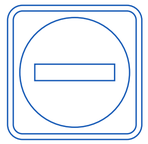
泛型类型

泛型命名规范（Naming convention）

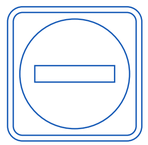
E – Element 元素，主要用在集合类中，e.g. ArrayList
K – Key 关键字，用在Map中
N – Number 数字
T – Type 类型
V – Value 值，用在Map中
S,U,V etc. – 2nd, 3rd, 4th 表示第二个，三个，四个类型

```
List<? extends Integer> intList = new ArrayList<>();
List<? extends Number>  numList = intList;
```

```
List<Number> numbers = new ArrayList<Integer>
```



```
List<Integer>[] array = new ArrayList<Integer>[10]
```



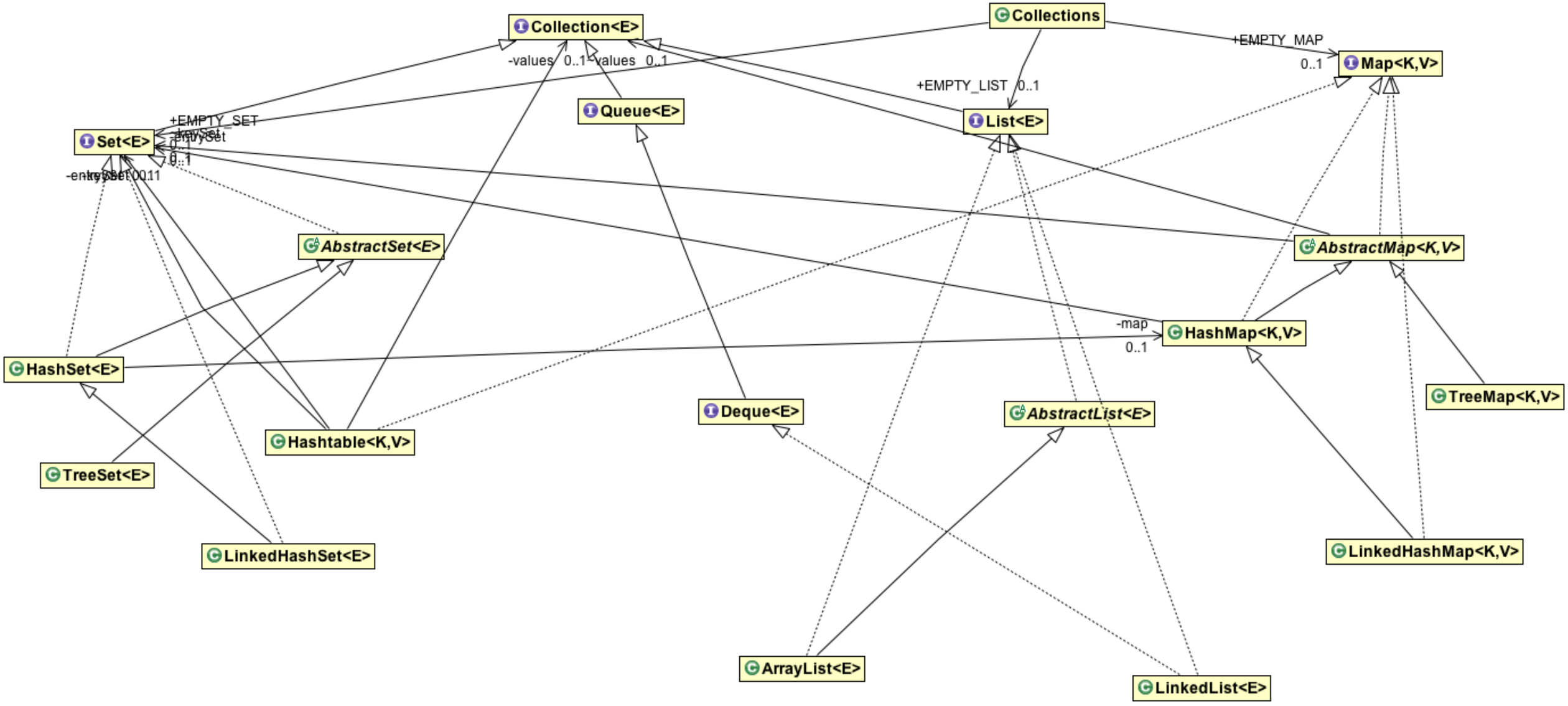
Collection 概念及综述

2

理解集合Collections

集合是一个java自带的框架，用于存储和操作一组对象
集合可以看作是一个容器

集合包括三个部分：
- 接口：抽象数据类型，如 java.util.Collection，java.util.Map…
- 实现类：ArrayList, LinkedList, HashMap, TreeMap, HashSet…
- 算法：searching, sorting and shuffling


Collection<E>
Collections
Map<K,V>
Queue<E>
List<E>
Set<E>
AbstractSet<E>
AbstractMap<K,V>
HashMap<K,V>
HashSet<E>
Deque<E>
AbstractList<E>
TreeMap<K,V>
Hashtable<K,V>
TreeSet<E>
LinkedHashSet<E>
LinkedHashMap<K,V>
ArrayList<E>
LinkedList<E>
+EMPTY_MAP
0..1
+EMPTY_LIST
0..1
-values
0..1
-values
0..1
+EMPTY_SET
-map
0..1

集合接口

Collections Interfaces

1. 集合Collection

处于collection顶层，提供了很多抽象方法，size, isEmpty, contains, add, remove, iterator…

2. 迭代器Iterator

3. 组合Set

不能包含重复元素，无序

4. 列表List

能包含重复元素，有序

集合接口

Collections Interfaces

5. 映射Map

key/value对，key唯一

6. 队列Queue

队列存放等待处理的元素，顺序为FIFO（first-in，first-out）

7. 双向队列Dequeue

和Queue不同的是，它接受双向数据处理，全称double ended queue，

集合类

Collections Class

Collection	Ordering	Random Access	Key-Value	Duplicate Elements	Null Element	Thread
ArrayList	Yes	Yes	No	Yes	Yes	No
LinkedList	Yes	No	No	Yes	Yes	No
HashSet	No	No	No	No	Yes	No
TreeSet	Yes	No	No	No	No	No
HashMap	No	Yes	Yes	No	Yes	No
TreeMap	Yes	Yes	Yes	No	No	No
Vector	Yes	Yes	No	Yes	Yes	Yes
Hashtable	No	Yes	Yes	No	No	Yes
Properties	No	Yes	Yes	No	No	Yes
Stack	Yes	No	No	Yes	Yes	Yes
CopyOnWriteArrayList	Yes	Yes	No	Yes	Yes	Yes
ConcurrentHashMap	No	Yes	Yes	No	Yes	Yes
CopyOnWriteArraySet	No	No	No	No	Yes	Yes

集合算法

Collections Algorithm

1. 排序Sort

提供两种操作：自然排序和比较器Comparator

2. 洗牌Shuffle

在保证每个元素平等的原则下随机打散列表

3. 搜索Search

基于有序列表的二进制搜索算法，也包含两种形式：自然排序和比较器。

4. 组合Composition

frequency：计算特定元素在特定集合中出现的次数

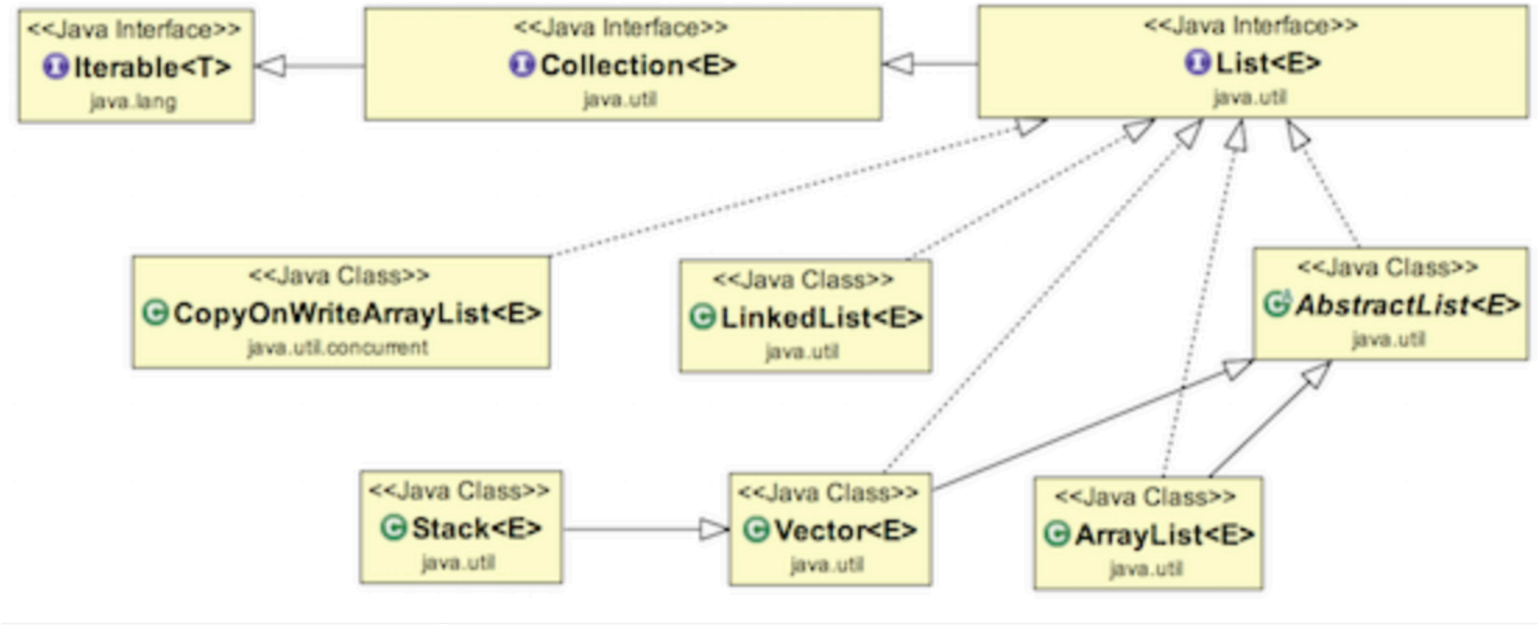
disjoint：判断是否两个集合有重合（有相同的元素）

常用集合类介绍和应用

3

理解列表List

列表是有序集合，它是继承与Collection的接口。
列表可以控制位置，可以在位置处插入元素，基于搜索搜索元素



List方法

1. **int size()**
2. **boolean isEmpty()**
3. **boolean contains(Object o)**
4. **Iterator<E> iterator()**
5. **Object[] toArray()**
6. **boolean add(E e)**: 在列表末尾添加元素
7. **boolean remove(Object o)**: 删除指定元素
8. **boolean retainAll(Collection<?> c)**: 保留包含在集合c中的元素
9. **void clear()**
10. **E get(int index)**
11. **E set(int index, E element)**
12. **ListIterator<E> listIterator()**
13. **List<E> subList(int fromIndex, int toIndex)**

理解ArrayList

最常用的集合类之一

java.util.ArrayList 实现接口 java.util.List interface

ArrayList不是线程安全，性能更好，除此之外和Vector一样
ArrayList可以包含重复元素，允许null
ArrayList是有序的，按序操作，可通过index访问元素
ArrayList默认10个元素数组，通过ensureCapacity(int minCapacity)
ArrayList Iterator是fail-fast，创建iterator后可执行初add/remove外，不可随意修改元素。

推荐写法：**List<String> list = new ArrayList<>()**

理解Map

Map表示key-value键值对集合

Map提供三种视图：keys，values，key-value mapping
Map key必须唯一，但value可以重复
Map不保证顺序，依赖于实现。HashMap不保证顺序，TreeMap保证顺序
大部分Map类都是基于AbstractMap抽象类
Map使用在key上的hashCode和equals方法操作get和put操作

Map方法

1. **int size()**
2. **boolean isEmpty()**
3. **boolean containsValue(Object value)**
4. **V get(Object key)**
5. **V put(K key, V value)**
6. **V remove(Object key)**
7. **void putAll(Map<? extends K, ? extends V> m)**: Copies all of the mappings from the specified map to this map.
8. **void clear()**
9. **Set<K> keySet()** keySet和Map是关联的
10. **Collection<V> values()**
11. **Set<Map.Entry<K, V>> entrySet()** entrySet和Map关联，任何修改都会在各自集合上体现

理解HashMap

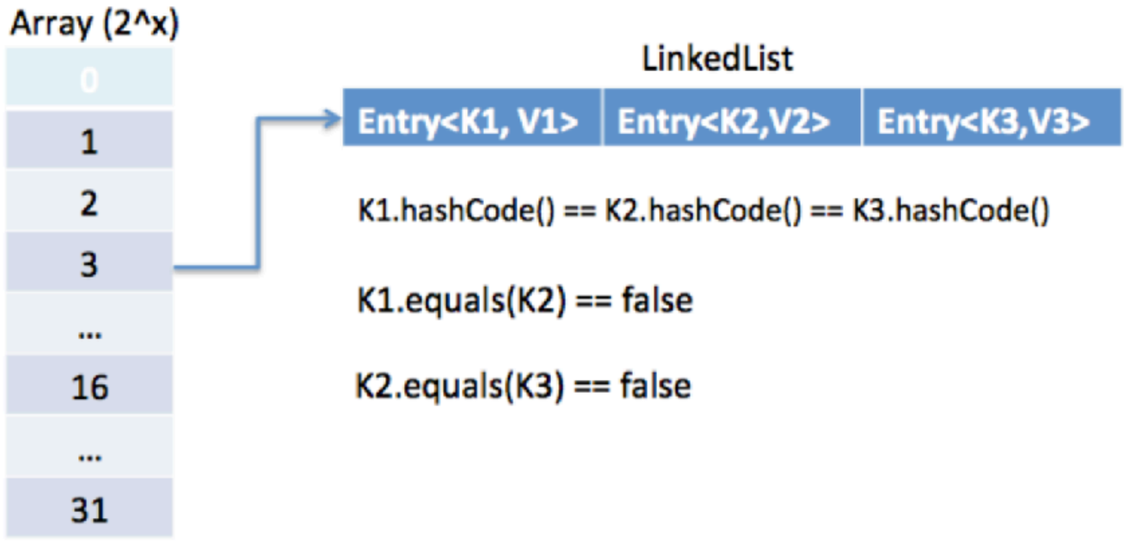
最常用的集合类之一

HashMap 基于hash表，继承自 AbstractMap，它实现接口 Map

HashMap允许null key和null value
HashMap是无序的
HashMap除了非同步和允许null key/value，它和HashTable是一样的
HashMap使用链表实现map （entry）的存储，叫做bucket或者bin。默认箱空间是16，权值2
HashMap非线程安全，对于多线程环境应该ConcurrentHashMap方法。

HashMap 工作原理

HashMap使用内部类Node<K,V>存储键值对，同时HashMap基于hash算法，并使用hashCode和equals方法保证key的唯一性



HashMap PUT

HashMap使用链表存储hash值相同的键值对，叫做桶bucket或垃圾箱bin。当调用put方法，它首先通过hashCode找到bucket。然后再用hashCode检测是否链表不存在，如果存在继续比较（目的是找位置），就使用equals方法比较key，如果相同就覆写，否则创建新的键值对加入改链表（bucket）。如果key不存在，键值对插入bucket

HashMap GET

HashMap使用hashCode首先找到bucket，然后继续使用hashCode和equals遍历链表找到元素，如果找到返回，如果找不到，返回null

作业

1. 查找并阅读 HashSet 资料

2. 实现一副扑克洗牌和发牌程序，要求扑克52张牌，可以发4张牌

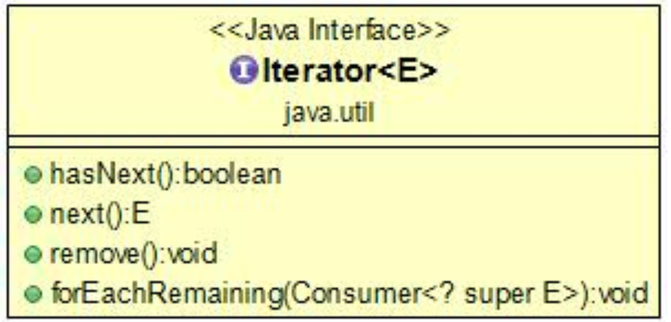
3. 使用HashMap重写杂货店程序

理解迭代器Iterator

Java光标（Cursor）是一种迭代器，用于顺序的迭代，遍历或者搜索一个集合。

四种光标：
1. Enumeration
2. Iterator
3. ListIterator
4. Spliterator (Java8)

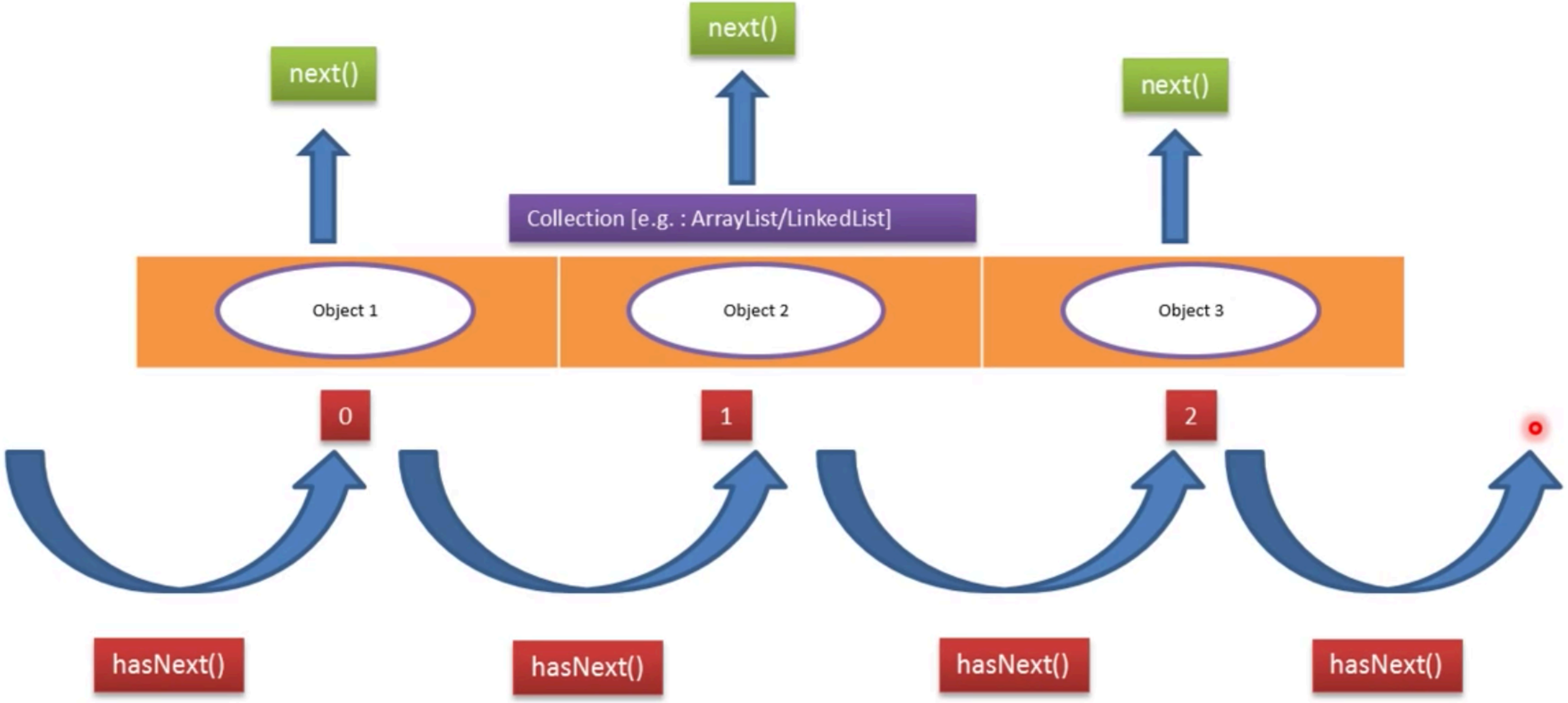
Iterator是一个Collection框架的接口，用于便利集合元素。

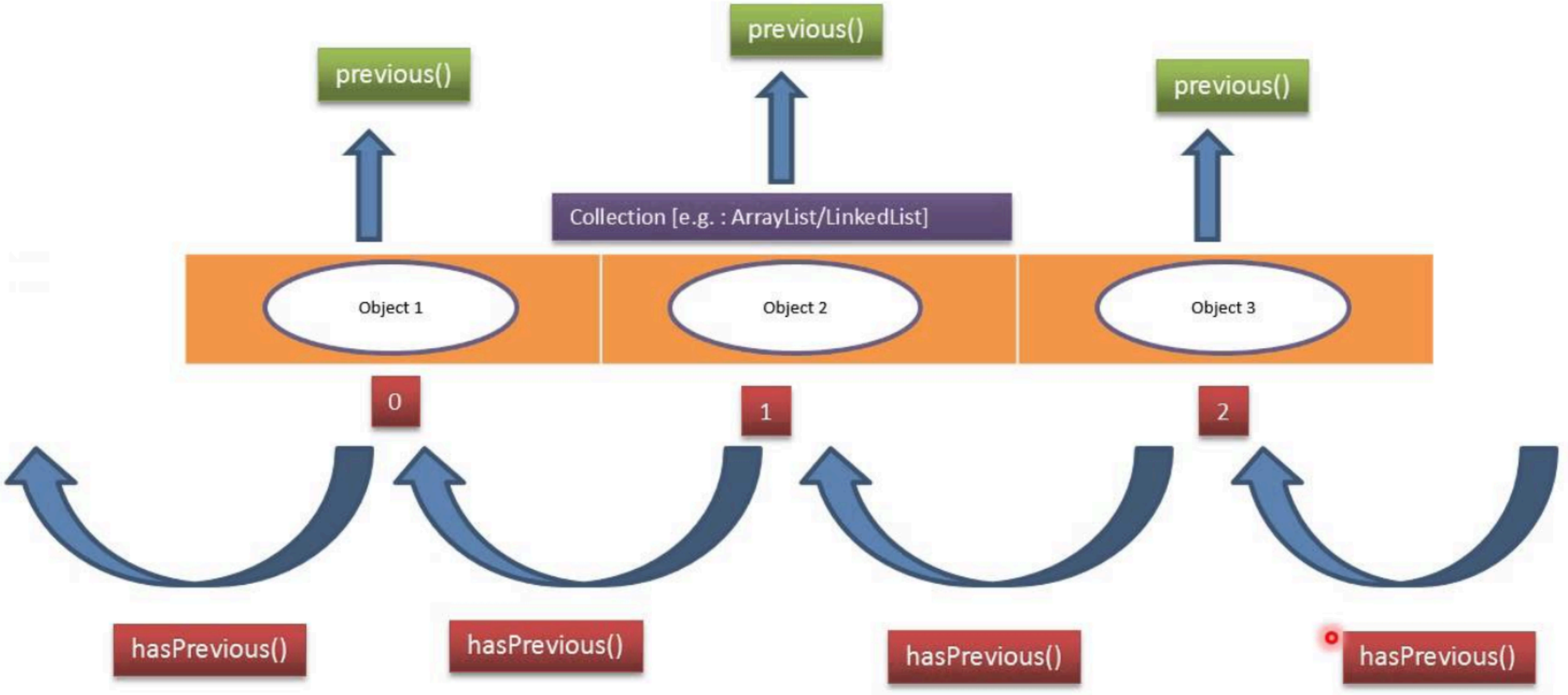


理解迭代器Iterator

- 用于一个一个按顺序遍历集合
- Java 1.2开始支持，任何集合类都实现了Iterator接口，也叫通用光标
- 支持读和删操作
- 相比Enumeration接口, Iterator更简单，可以完全替代

缺点：
不支持增和更新
Iterator仅支持“前进”，不能“倒退”
仅支持顺序便利，不支持平行，不支持大数据（Spliterator）


next()
next()
next()
Collection [e.g. : ArrayList/LinkedList]
Object 1
Object 2
Object 3
0
1
2
hasNext()
hasNext()
hasNext()
hasNext()


previous()
previous()
previous()
Collection [e.g. : ArrayList/LinkedList]
Object 1
Object 2
Object 3
0
1
2
hasPrevious()
hasPrevious()
hasPrevious()
hasPrevious()

Thanks!

Any questions?