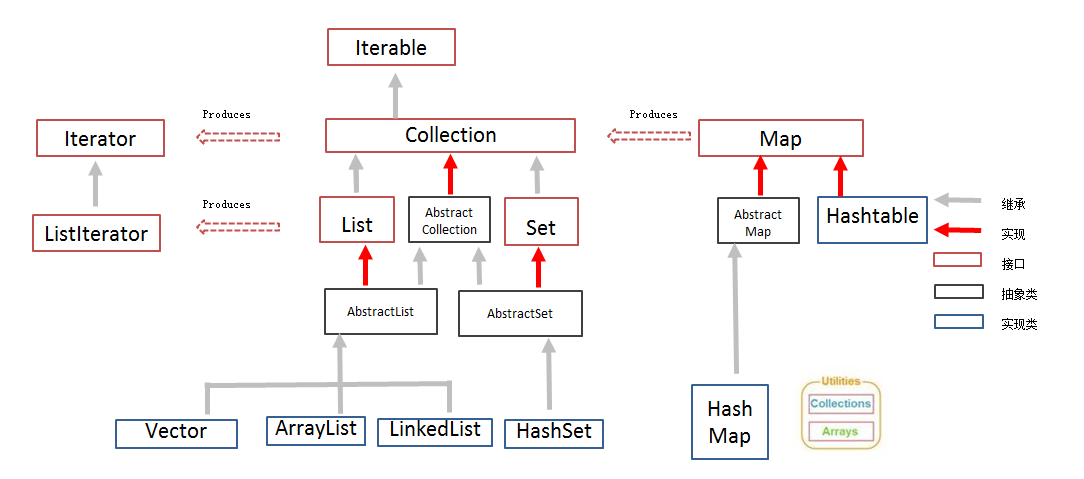
# 总结

## 集合继承体系的面向对象思想



* 接口：用来明确所有集合中该具有的功能，相当于在定义集合功能标准；
* 抽象类：把多个集合中功能实现方式相同的方法，抽取到抽象类实现，具体集合不再遍写，继承使用即可；
* 具体类：继承抽象类，实现接口，重写所有抽象方法，达到具备指定功能的集合。每个具体集合类，根据自身的数据存储结构方式，对接口中的功能方法，进行不同方式的实现。

## 知识点总结

* Collection:集合的顶层接口，定义了集合中通用的功能方法

常用方法：

* + boolean add(Object e) 把给定的对象添加到当前集合中
  + void clear() 清空集合中所有的元素
  + boolean remove(Object o) 把给定的对象在当前集合中删除
  + boolean contains(Object o) 判断当前集合中是否包含给定的对象
  + boolean isEmpty() 判断当前集合是否为空
  + Iterator iterator() 迭代器，用来遍历集合中的元素的
  + int size() 返回集合中元素的个数
  + Object[] toArray() 把集合中的元素，存储到数组中
* Iterator: 迭代器
  + Object next()返回迭代的下一个元素
  + boolean hasNext()如果仍有元素可以迭代，则返回 true。
* 增强for

简化数组和Collection集合的遍历

格式：

for(元素数据类型 变量 : 数组或者Collection集合) {

使用变量即可，该变量就是元素

}

好处：简化遍历

* 泛型： 用来约束数据的数据类型
  + 泛型的格式：

<数据类型>

泛型可以使用在 类，接口，方法上

* + 泛型的好处

A:将运行期遇到的问题转移到了编译期

B:省去了类型强转的麻烦

* + 泛型高级(通配符):
    - 泛型通配符<?>

泛型类型不确定时，可以通过通配符<?>表示。但是一旦使用泛型的通配符后，只能使用Object类中的共性方法，集合中元素自身方法无法使用。

* + - <? extends E>

向上限定，E及其子类

? extends Animal

? 代表的就是Animal类本身或者是Animal类的子类

* + - <? super E>

向下限定，E及其父类

? super Animal

? 代表的就是Animal类本身或者是Animal类的父类

## 知识点总结

* List与Set集合的区别？

List:

它是一个有序的集合(元素存与取的顺序相同)

它可以存储重复的元素

Set:

它是一个无序的集合(元素存与取的顺序可能不同)

它不能存储重复的元素

* List集合中的特有方法
  + void add(int index, Object element) 将指定的元素，添加到该集合中的指定位置上
  + Object get(int index)返回集合中指定位置的元素。
  + Object remove(int index) 移除列表中指定位置的元素, 返回的是被移除的元素
  + Object set(int index, Object element)用指定元素替换集合中指定位置的元素,返回值的更新前的元素
* **ArrayList**:

底层数据结构是数组，查询快，增删慢

* LinkedList:

底层数据结构是链表，查询慢，增删快

* **HashSet**:

元素唯一，不能重复

底层结构是 哈希表结构

元素的存与取的顺序不能保证一致

如何保证元素的唯一的？

重写hashCode() 与 equals()方法

* LinkedHashSet:

元素唯一不能重复

底层结构是 哈希表结构 + 链表结构

元素的存与取的顺序一致

## 知识点总结

* Map集合:

map集合中的元素都是成对出现，成对存储的

map集合中的元素都是以一对键和值的形式组成存在的，称为键值对，理解为夫妻对

map集合中的键不能重复存储，值可以重复

map集合中的每一个键 对应着一个值

* + 方法：

V put(K key, V value) 把指定的键与指定的值添加到Map集合中

V remove(Object key) 把指定的键 所对应的键值对元素 在Map集合中删除，返回被删除元素的值

Set<Map.Entry<K,V>> entrySet() 获取到Map集合中所有的键值对对象的集合(Set集合)

V get(Object key) 根据指定的键，在Map集合中获取对应的值

Set<K> keySet() 获取Map集合中所有的键，存储到Set集合中

* Map集合遍历的两种方式
  + 方式1：根据键找值的方式

//a, 获取到Map集合中所有的键，返回对应的Set集合

//b, 遍历键的集合，获取到每一个键

//c, 通过键，找到对应的值

//获取到Map集合中所有的键，返回对应的Set集合

Set<String> keys = map.keySet();

//遍历键的集合，获取到每一个键

for (String key : keys) {

//通过键，找到对应的值

Student s = map.get(key);

System.out.println( key + "..." + s.getName() + "..." + s.getAge() );

}

* + 方式2：根据键值对对象找键和值的方式

//a, 获取Map集合中所有的键值对元素,返回对应的Set集合

//b, 遍历键值对元素集合，获取到每一个键值对元素对象

//c, 通过键值对元素对象，获取对应的键，和对应的值

//获取Map集合中所有的键值对元素,返回对应的Set集合

Set< Map.Entry<String, Student>> entrySet = map.entrySet();

//遍历键值对元素集合，获取到每一个键值对元素对象

for (Map.Entry<String, Student> entry : entrySet) {

//通过键值对元素对象，获取对应的键，和对应的值

//找键

String key = entry.getKey();

//找值

Student s = entry.getValue();

//打印

System.out.println( key+"..."+s.getName()+"..."+s.getAge() );

}

* HashMap:
  + 特点：

是Map集合的子集合

底层采用哈希表结构

HashMap集合中的key不能重复，通过重写hashCode() 与 equals()方法来保证键的唯一。

不能保证元素存与取的顺序完全一致

* LinkedHashMap:
  + 特点：

是HashMap集合的子集合

底层采用哈希表+链表结构

LinkedHashMap集合中的key不能重复，通过重写hashCode() 与 equals()方法来保证键的唯一。

* Collections中的方法：

public static <T> void sort(List<T> list) 排序

public static void shuffle(List<?> list) 集合中的元素存储位置随机打乱