Java-集合类框架

- Java 集合框架主要包括两种类型的容器，
  - 一种是集合（Collection），存储一个元素集合；
  - 另一种是图（Map），存储键/值对映射。
- Collection 接口又有 3 种子类型，List、Set 和 Queue，再下面是一些抽象类，最后是具体实现类，常用的有 ArrayList、LinkedList、HashSet、LinkedHashSet、HashMap 等等。
- 集合框架是一个用来代表和操纵集合的统一架构。所有的集合框架都包含如下内容：
  - 接口：是代表集合的抽象数据类型。例如 Collection、List、Set、Map 等。之所以定义多个接口，是为了以不同的方式操作集合对象
  - 实现（类）：是集合接口的具体实现。从本质上讲，它们是可重复使用的数据结构，例如：ArrayList、LinkedList、HashSet、HashMap。
  - 算法：是实现集合接口的对象里的方法执行的一些有用的计算，例如：搜索和排序，这些算法实现了多态，那是因为相同的方法可以在相似的接口上有着不同的实现。

集合类接口-Collection

- 集合类基本接口，说明对象按照“集合”的方式放置在一起时，所应当具有的结构特征属性和带有共性的操作方法。
- 子接口包含List和Set。
  - List规定实现其的集合类元素具有可以控制的顺序。
  - Set不能包含重复的元素。
- 接口定义的方法
  - boolean add(Object c) : 向集合类中添加一个新元素。
  - boolean addAll(Collection c): 将c所引用集合对象中所有对象加入现有集合中
  - boolean remove(Object o): 删除一个指定对象的引用
  - Object[] toArray(): 将集合中所有元素以数组对象引用形式返回。

常用集合类

- ArrayList和LinkedList
  - LinkedList
    - 该类实现了List接口，允许有null（空）元素。主要用于创建链表数据结构，该类没有同步方法，如果多个线程同时访问一个List，则必须自己实现访问同步，解决方法就是在创建List时候构造一个同步的List。例如：
    - `List list=Collections.synchronizedList(newLinkedList(...));`
    - LinkedList 查找效率低。
  - ArrayList

    - 该类也是实现了List的接口，实现了可变大小的数组，随机访问和遍历元素时，提供更好的性能。该类也是非同步的,在多线程的情况下不要使用。
    - ArrayList 增长当前长度的50%，插入删除效率低。
- HashSet和HashMap

ArrayList

```
public class RunoobTest {
    public static void main(String[] args) {
        ArrayList<String> sites = new ArrayList<String>();
        sites.add("Google");
        sites.add("Runoob");
        sites.add("Taobao");
        sites.add("Weibo");
                System.out.println(sites.get(1));  // 访问第二个元素
                sites.set(2, "Wiki"); // 第一个参数为索引位置，第二个为要修改的值
                sites.remove(3); // 删除第四个元素
                System.out.println(sites);

                System.out.println(sites.size());
                for (int i = 0; i < sites.size(); i++) {
                        System.out.println(sites.get(i));
                }
                for (String i : sites) {
            System.out.println(i);
        }
    }
}
```

LinkedList

- 上述ArrayList的方法都拥有。同时拥有以下方法，访问更高效
    - addFirst()
    - addLast()
    - removeFirst()
    - removeLast()
    - getFirst()
    - getLast()

HashSet

- 添加元素 `add(Object)`
- 判断元素是否存在 `contains(Object)`
- 删除元素 `remove()`

- 删除所有元素 `clear()`
- 计算大小 `size()`
- 迭代所有元素 `for(Object i : hashSet)`

HashMap

```
public class RunoobTest {
    public static void main(String[] args) {
        // 创建 HashMap 对象 Sites
        HashMap<Integer, String> Sites = new HashMap<Integer, String>();
        // 添加键值对
        Sites.put(1, "Google");
        Sites.put(2, "Runoob");
        Sites.put(3, "Taobao");
        Sites.put(4, "Zhihu");
        System.out.println(Sites);
        System.out.println(Sites.get(3)); //使用 get(key) 方法来获取 key 对应的
value
        Sites.remove(4);
        Sites.clear();
        // 输出 key 和 value
        for (Integer i : Sites.keySet()) {
            System.out.println("key: " + i + " value: " + Sites.get(i));
        }
        // 返回所有 value 值
        for(String value: Sites.values()) {
          // 输出每一个value
          System.out.print(value + ", ");
    }
}
```