### Java第一阶段—DAY16-JAVA案例

1. 三国时期，刘备派关羽守樊城，张飞守新野，赵子龙守徐州，他自己坐镇荆州。使用HashMap存储相关数据，练习相关API。

public class Demo07HashMap {  
  
 public static void main(String[] args) {  
*// 三国时期，刘备派关羽守樊城，张飞守新野，赵子龙守徐州，他自己坐镇荆州。  
 //创建 map对象* HashMap<String, String> map = new HashMap<String, String>();  
  
 *//添加元素到集合* map.put("关羽", "樊城");  
 map.put("张飞", "新野");  
 map.put("赵子龙", "徐州");  
 map.put("刘备", "荆州");  
 System.out.println(map);  
  
*// 删除赵子龙* System.out.println(map.remove("赵子龙"));  
 System.out.println(map);  
  
 *// 想要查看刘备守哪座城？* System.out.println(map.get("刘备"));  
 }  
}

**class** Hero{  
  
 **private** String name ;  
  
 **public** Hero(String name) {  
 **this**.name = name;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (!(o **instanceof** Hero)) **return false**;  
 Hero hero = (Hero) o;  
 **return** Objects.equals(name, hero.name);  
 }  
  
 @Override  
 **public int** hashCode() {  
 **return** Objects.hash(name);  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Hero{"** +  
 **"name='"** + name + **'\''** +  
 **'}'**;  
 }  
}  
  
**public class** Demo08HashMap {  
  
 **public static void** main(String[] args) {  
 *//创建 map对象* HashMap<Hero, String> map = **new** HashMap<Hero, String>();  
  
 *//添加元素到集合* map.put(**new** Hero(**"关羽"**), **"樊城"**);  
 map.put(**new** Hero(**"张飞"**), **"新野"**);  
 map.put(**new** Hero(**"赵子龙"**), **"徐州"**);  
 map.put(**new** Hero(**"刘备"**), **"荆州"**);  
 map.put(**new** Hero(**"刘备"**), **"荆州"**);  
 System.out.println(map);  
  
 }  
}

**public class** Demo09HashMap {  
  
 **public static void** main(String[] args) {  
 *//创建 map对象* HashMap<String, String> map = **new** HashMap<String, String>();  
  
 *//添加元素到集合* map.put(**"关羽"**, **"樊城"**);  
 map.put(**"张飞"**, **"新野"**);  
 map.put(**"赵子龙"**, **"徐州"**);  
 map.put(**"刘备"**, **"荆州"**);  
  
  
*//方式1* Set<String> keys = map.keySet();  
 **for** (String name : keys) {  
 System.out.println(name + **" "** + map.get(name));  
 }  
  
 Collection<String> values = map.values();  
 **for** (String value : map.values()) {  
 System.out.println(**"Value = "** + value);  
 }  
  
*// 方式2* Iterator<Map.Entry<String, String>> entries = map.entrySet().iterator();  
 **while** (entries.hasNext()) {  
 Map.Entry<String, String> entry = entries.next();  
 System.out.println(**"Key = "** + entry.getKey() + **", Value = "** + entry.getValue());  
 }  
 System.out.println(**"----------"**);  
*// 方式3 推荐* **for**(Map.Entry<String, String> entry : map.entrySet()) {  
 System.out.println(**"key = "** + entry.getKey() + **", value = "** + entry.getValue());  
 }  
  
  
 }  
}

1. 仿照ArrayList自定义泛型类。

class MyGeneric<E> {  
 private E e ;  
  
 public void set(E e){  
 this.e = e ;  
 }  
  
 public E get(){  
 return e ;  
 }  
  
}  
  
  
public class Demo12Generic {  
  
 public static void main(String[] args) {  
 *// 泛型类确定为String* MyGeneric<String> g1 = new MyGeneric<String>();  
 g1.set("blb");  
 System.out.println(g1.get());  
 *// 泛型类确定为Integer* MyGeneric<Integer> g2 = new MyGeneric<Integer>();  
 g2.set(9);  
 System.out.println(g2.get());  
  
 }  
}

1. 测试Collections的常用API

public class Demo10Collections {  
  
 @Test  
 public void testBinarySearch() {  
 ArrayList<Integer> list = new ArrayList<Integer>();  
 Collections.addAll(list,1,2,3,4,5,6,7,8,9);  
 System.out.println(Collections.binarySearch(list, 4));  
 }  
  
 @Test  
 public void testReplaceAll() {  
 ArrayList<String> list = new ArrayList<String>();  
 Collections.addAll(list,"关羽","张飞","赵子龙","刘备");  
 Collections.replaceAll(list,"刘备","马超");  
 System.out.println(list);  
 }  
  
 @Test  
 public void testSwap() {  
 ArrayList<String> list = new ArrayList<String>();  
 Collections.addAll(list,"关羽","张飞","赵子龙","刘备");  
*// 往集合中的元素进行互换* Collections.swap(list,0,2);  
 System.out.println(list);  
 }  
  
 @Test  
 public void testSort() {  
 ArrayList<Integer> list = new ArrayList<Integer>();  
  
*// 往集合中添加元素* Collections.addAll(list,3,9,6,4,1,5);  
 System.out.println(list);  
  
*// 用默认的比较器对集合中的元素排序* Collections.sort(list);  
 System.out.println(list);  
  
 Collections.sort(list, new Comparator<Integer>() {  
 @Override  
 public int compare(Integer o1, Integer o2) {  
 return o2-o1;  
 }  
 });  
 System.out.println(list);  
 }  
  
 @Test  
 public void testMinAndMax() {  
 ArrayList<Integer> list = new ArrayList<Integer>();  
 Collections.addAll(list,3,9,6,4,1,5);  
 System.out.println(Collections.min(list));  
 System.out.println(Collections.max(list));  
 }  
  
 @Test  
 public void testReverse() {  
 ArrayList<String> list = new ArrayList<String>();  
 Collections.addAll(list,"关羽","张飞","赵子龙","刘备");  
*// 往集合中所有元素反转* Collections.reverse(list);  
 System.out.println(list);  
 }  
  
  
 @Test  
 public void testShuffle() {  
 ArrayList<String> list = new ArrayList<String>();  
*// 往集合中添加元素* Collections.addAll(list,"关羽","张飞","赵子龙","刘备");  
 System.out.println(list);  
  
*// 随机打乱这个集合中的元素* Collections.shuffle(list);  
 System.out.println(list);  
 }  
  
 @Test  
 public void testAddAll() {  
 ArrayList<String> list = new ArrayList<String>();  
*// 往集合中添加元素* Collections.addAll(list,"关羽","张飞","赵子龙","刘备");  
 System.out.println(list);  
 }  
  
}