# Ссылки

Ознакомится с

<http://docs.oracle.com/javase/tutorial/java/generics/>

<http://howtodoinjava.com/2014/07/24/java-generics-what-is-pecs-producer-extends-consumer-super/>

# Youtube

Посмотреть доклад 'Неочевидные Дженерики' c JeeConf. <https://www.youtube.com/watch?v=H5WlE8BK5sI>

# Задания

Ссылка на примеры из класса: <https://github.com/sbtjavaschool/generics>

1. **Параметризовать CountMap (из репозитория выше) и реализовать его.**

**public interface** CountMap {  
 *// добавляет элемент в этот контейнер.* **void** add(Object o);  
  
 *//Возвращает количество добавлений данного элемента* **int** getCount(Object o);  
  
 *//Удаляет элемент и контейнера и возвращает количество его добавлений(до удаления)* **int** remove(Object o);  
  
 *//количество разных элементов* **int** size();  
  
 *//Добавить все элементы из source в текущий контейнер,   
 // при совпадении ключей, суммировать значения* **void** addAll(CountMap source);  
  
 *//Вернуть java.util.Map. ключ - добавленный элемент,   
 // значение - количество его добавлений* Map toMap();  
  
 *//Тот же самый контракт как и toMap(), только всю информацию записать в destination* **void** toMap(Map destination);  
}

пример использования:

CountMap<Integer> map = **new** CountMapImpl<>();  
  
map.add(10);  
map.add(10);  
map.add(5);  
map.add(6);  
map.add(5);  
map.add(10);  
  
*// int count = map.getCout(5); // 2  
// int count = map.getCout(6); // 1  
// int count = map.getCout(10); // 3*

1. **Параметризовать методы, используя правило PECS, и реализовать их.**

**public class** CollectionUtils {  
 **public static**<T> **void** addAll(List<? **extends** T> source, List<? **super** T> destination) {  
 destination.addAll(source);  
 }  
  
 **public static** List newArrayList() { }  
  
 **public static int** indexOf(List source, Object o) { }  
  
 **public static** List limit(List source, **int** size) { }  
  
 **public static void** add(List source, Object o) { }  
  
 **public static void** removeAll(List removeFrom, List c2) { }  
  
 **public static boolean** containsAll(List c1, List c2) { }  
  
 **public static boolean** containsAny(List c1, List c2) { }  
  
 **public static** List range(List list, Object min, Object max) { }  
  
 **public static** List range(List list, Object min, Object max, Comparator comparator) { }  
}

Пояснения к некоторым методам.

*//true если первый лист содержит все элементы второго***public static boolean** containsAll(List c1, List c2) {  
  
}  
  
  
*//true если первый лист содержит хотя-бы 1 второго***public static boolean** containsAny(List c1, List c2) {  
  
}   
  
  
*//Возвращает лист, содержащий элементы из входного листа в диапазоне от min до max.   
// Элементы сравнивать через Comparable.  
// Прмер range(Arrays.asList(8,1,3,5,6, 4), 3, 6) вернет {3,4,5,6}***public static** List range(List list, Object min, Object max) {  
}  
  
  
*//Возвращает лист, содержащий элементы из входного листа в диапазоне от min до max.   
// Элементы сравнивать через Comparable.*

*// Прмер range(Arrays.asList(8,1,3,5,6, 4), 3, 6) вернет {3,4,5,6}***public static** List range(List list, Object min, Object max, Comparator comparator) {  
  
}