Домашнее задание (Collection)

Задание 1. Классы коллекций

Collection	Ordering	Random Access	Key-Value Pairs	Allows Duplicate	Allows null	Thread Safe	Blocking Operations
ArrayList	Yes	Yes	No	Yes	Yes	No	No
CopyOn Write ArrayList	Yes	Yes	No	Yes	Yes	Yes	No
LinkedList	Yes	No	No	Yes	Yes	No	No
HashSet	No	No	No	No	Yes	No	No
Concurrent SkipList Set	Yes	No	No	No	No	Yes	No
CopyOn Write ArraySet	Yes	Yes	No	No	Yes	Yes	No
TreeSet	Yes	No	No	No	No	No	No
Linked HashSet	Yes	No	No	No	Yes	No	No
HashMap	No	Yes	Yes	No	Yes	No	No
Concurrent HashMap	No	Yes	Yes	No	No	Yes	No
Concurrent SkipList Map	Yes	Yes	Yes	No	No	Yes	No
TreeMap	Yes	Yes	Yes	No	No	No	No
Linked HashMap	Yes	No	Yes	No	No	No	No
WeakHash Map	No	Yes	Yes	No	Yes	No	No
HashTable	No	Yes	Yes	No	No	Yes	No
Properties	No	Yes	Yes	No	No	Yes	No
Stack	Yes	Yes	No	Yes	Yes	No	No
Vector	Yes	Yes	No	Yes	Yes	No	No
Array Deque	Yes	Yes	No	Yes	No	No	No
Concurrent Linked Queue	Yes	No	No	Yes	No	Yes	No
Array Blocking	Yes	No	No	Yes	No	Yes	Yes

Queue							
Linked Blocking Queue	Yes	No	No	Yes	No	Yes	Yes
Linked Transfer Queue	Yes	No	No	Yes	No	Yes	Yes
Priority Blocking Deque	Yes	No	No	Yes	No	Yes	Yes
Linked Blocking Queue	Yes	No	No	Yes	No	Yes	Yes
SynchronousQueue	Yes	No	No	Yes	No	Yes	Yes
Delay Queue	Yes	No	No	Yes	No	Yes	Yes

Задание 3. Ссылки на коллекции

Определена иерархия классов

```
class MedicalStaff{}
class Doctor extends MedicalStaff{}
class Nurse extends MedicalStaff{}
class HeadDoctor extends Doctor{}
```

Укажите корректные и некорректные операторы. Дайте ответу пояснение.

	correct	not correct
`Doctor doctor1 = new Doctor();`	•	
`Doctor doctor2 = new MedicalStaff();`		•
`Doctor doctor3 = new HeadDoctor();`	•	
`Object object1 = new HeadDoctor();`	•	
`HeadDoctor doctor5 = new Object();`		•
`Doctor doctor6  = new Nurse();`		•
`Nurse nurse = new Doctor();`		•
`Object object2 = new Nurse();`	•	
`List<Doctor> list1= new ArrayList<Doctor>();`	•	
`List<MedicalStaff> list2 = new ArrayList<Doctor>();`		•
`List<Doctor> list3 = new ArrayList<MedicalStaff>();`		•
`List<Object> list4 = new ArrayList<Doctor>();`		•
`List<Object> list5 = new ArrayList<Object>();`	•	

Задание 4. Применение коллекций

Заполните таблицу.

	Основная функциональность	Примеры типичного использования
Set	Позволяет хранить множество значений, гарантируя их уникальность. Не гарантирует порядок.	Set<String> set = new HashSet<>(); set.add(«string1»); set.add(«string2»); set.add(«string1»); for (String s : set) System.out.println(s);
List	Является тем же массивом, но в отличии от него динамически расширяется. Хранит любые объекты, в том числе null. Как и в массиве, каждый элемент имеет свой порядковый номер (индекс).	List<String> list = new ArrayList<>(); list.add(«string1»); list.add(«string2»); list.set(1, «string1»); for (String s : list) System.out.println(s);

Queue	Коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки. Упорядочивают элементы в FIFO (First-In-First-Out), но не обязательно.	Queue<String> queue = new PriorityQueue<>();
		queue.add("1");
		queue.add("2");
		queue.add("3");
		for (Object s : queue.toArray())
		System.out.println(s);

;Map	Карта значений. Позволяет хранит информацию в виде «ключ — значение». Ключ при этом должен быть уникальным, значение — нет.	Map<String, String> map = new HashMap<>(); map.put(«1», «string1»); map.put(«2», «string1»); map.put(«1», «ssss»); for (Map.Entry<String, String> entry : map.entrySet()) System.out.println(«Key: » + entry.getKey() + « Value: » + entry.getValue());