**Task1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ordering | Random  Access | Key-Value  Pairs | Allows  Duplicates | Allows Null Values | Thread  Safe | Blocking  Operations |
| [**ArrayList**](http://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html) | + | + | - | + | + | - | - |
| [**Vector**](http://docs.oracle.com/javase/8/docs/api/java/util/Vector.html) | + | + | - | + | + | + | + |
| [**Stack**](http://docs.oracle.com/javase/8/docs/api/java/util/Stack.html) | + | + | - | + | + | +- | + |
| [**LinkedList**](http://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html) | + | + | - | + | + | - | - |
| [**HashSet**](http://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html) | - | - | - | - | + | - | - |
| [**LinkedHashSet**](http://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashSet.html) | + | - | - | - | + | - | - |
| [**TreeSet**](http://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html) | + | - | - | - | + | - | - |
| [**PriorityQueue**](http://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html) | + | - | - | + | - | - | - |
| [**ArrayDeque**](http://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html) | + | - | - | + | - | - | - |
| [**Hashtable**](http://docs.oracle.com/javase/8/docs/api/java/util/Hashtable.html) | - | by key+ | + | Key- Value+ | - | + | + |
| [**HashMap**](http://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html) | - | by key+ | + | Key- Value+ | + | - | - |
| [**LinkedHashMap**](http://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html) | + | by key+ | + | Key- Value+ | + | - | - |
| [**TreeMap**](http://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html) | + | by key+ | + | Key- Value+ | + | - | - |

Не уверена насчёт последней колонки

**Task2** реализован в предыдущем модуле.

Как ведет себя map-коллекция если в неё добавить элемент с ключом, который уже присутствует?

Заменит старое значение по этому ключу на новое

**Task3**

**class MedicalStaff{}**

**class Doctor extends MedicalStaff{}**

**class Nurse extends MedicalStaff{}**

**class HeadDoctor extends Doctor{}**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Doctor doctor1 = new Doctor();** | **+** |  |
| **Doctor doctor2 = new MedicalStaff();** | **-** | **Справа может оказаться Nurse** |
| **Doctor doctor3 = new HeadDoctor();** | **+** | **HeadDoctor потомок Doctor** |
| **Object object1 = new HeadDoctor();** | **+** | **Object - предок всех классов** |
| **HeadDoctor doctor5 = new Object();** | **-** | **Справа может быть что угодно** |
| **Doctor doctor6 = new Nurse();** | **-** |  |
| **Nurse nurse = new Doctor();** | **-** |  |
| **Object object2 = new Nurse();** | **+** | **Object - предок всех классов** |
|  |  |  |
| **List<Doctor> list1= new ArrayList<Doctor>();** | **+** | **ArrayList – реализация интерфейса List** |
| **List< MedicalStaff > list1= new ArrayList<Doctor>();** | **-** | ArrayList <OtherClass> не наследник AbstractList <Object> |
| **List<Doctor> list1= new ArrayList< MedicalStaff >();** | **-** |
| **List< Object> list1= new ArrayList<Doctor>();** | **-** |
| **List< Object> list1= new ArrayList<Object>();** | **+** | **ArrayList – реализация интерфейса List** |

**Task4**

|  |  |  |
| --- | --- | --- |
|  | Основная функциональность | Примеры типичного использования |
| Set | Добавление элементов не допускает их дублирования | Моделирование математического понятия множества |
| List | Произвольный доступ к элементам по индексу | Хранение упорядоченного массива данных переменной длинны |
| Queue | Вставка и удаление объектов отвечающая очереди FIFO | Накопление и обработка последовательно происходящих событий |
| Map | Работа с данными вида ключ-значение. Ключи уникальны | Работа с таблицами данных. Поиск данных в таблице по идентификатору |