|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ordering | Random access | Key-value pairs | Allows duplicates | Allows null values | Thread safe | Blocking operations |
| ArrayList | Yes | Yes | No | Yes | Yes | No |  |
| LinkedList | Yes | No | No | Yes | Yes | No |  |
| HashSet | No | No | Yes | No | Yes | No |  |
| TreeSet | Yes | No | Yes | No | Yes | No |  |
| LinkedHashSet | Yes | No | Yes | No | Yes | No |  |
| HashMap | No | No | Yes | No | Yes | No |  |
| TreeMap | Yes | No | Yes | No | Yes | No |  |
| LinkedHashMap | Yes | No | Yes | No | Yes | No |  |
| PriorityQueue | Yes | No | No | Yes | No | No |  |
| Vector | Yes | Yes | No | Yes | Yes | Yes |  |
| Stack | Yes | Yes | No | Yes | Yes | Yes |  |
| HashTable | No | No | Yes | Yes | No | Yes |  |

6.3

|  |  |  |
| --- | --- | --- |
|  | 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>(); | + |  |

6.4

|  |  |  |
| --- | --- | --- |
|  | Основная функциональность | Примеры типичного использования |
| Set | Set обладает такой же функциональностью, как и Collection.  void clear():  boolean add(Object item)  boolean contains (Object obj);  boolean isEmpty();  boolean remove(Object obj);  int size(); | Каждый элемент хранится только в одном экземпляре, различные реализации Set используют разный порядок хранения элементов.  Чаще всего используется для тестирования принадлежности, чтобы пользователь мог узнать присутствует ли объект в множестве. |
| List | Содержит все методы интерфейса Collection и добавляет следующие:  void add(int index, Object item);  E get(int index);  E set(int index, Object item);  int indexOf(Object obj);  int lastIndexOf(Object obj);  List<E> subList(int from, int to); | Гарантирует определенный порядок следования элементов. Используется как динамически расширяемый массив:  ArrayList оптимизирован для произвольного доступа к элементам, но с относительно медленными операциями вставки/удаления элементов в середине списка.  Контейнер LinkedList, оптимизирован для последовательного доступа, с быстрыми операциями вставки/удаления в середине списка. Произвольный доступ к элементам выполняется относительно медленно. |
| Queue | Содержит все методы интерфейса Collection и добавляет следующие:  E element();  E poll();  E peek();  boolean offer(Object obj);  E remove(); | Реализует очередь: элементы добавляются только с одного конца, а извлекаются с другого. Используются для надежного перемещения объектов из одной области программы в другую. |
| Map | Содержит все методы интерфейса Collection и добавляет следующие:  void clear():  boolean containsKey(Object k);  boolean containsValue(Object k);  Set<Map.Entry<K,V>> entrySet();  boolean equals(Object obj);  boolean isEmpty();  V get(Object k);  V put(K k, V v);  Set<K> keyset();  Collection<V> values();  Void putAll(Map<? Extends K, ?extends V> map);  V remove(Object k);  int size(); | В контейнере Map хранятся два объекта: ключ и связанное с ним значение. Ключи должны быть уникальными, а значения могут дублироваться. Используется для поиска объектов по ключу. |