Ziegfreid Morissey Flameño

1. **Answer:**

|  |
| --- |
| Collections – Based on my understanding, this is the class that consists of methods that manipulates collections. Such methods are “sort” or “reverse”.    Collection – This is the root interface in the *collection hierarchy*. A collection represents a group of objects. These collections may allow duplicate elements and others contain unique elements Some are ordered and others unordered.  List – This is an ordered collection (also known as a sequence). Lists allow access to its elements by their integer index (position in the list), and it also allows search for elements in the list.  Set – This is collection that contains unique elements. Sets contain no pair of elements e1 and e2 such that e1.equals(e2), and at most one null element.  Map – This is an object that maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value.  ArrayList - This list is used when adding/removing elements at the back as it performs better than LinkedList. This implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to Vector, except that it is unsynchronized.)  LinkedList - This list is used when adding/removing elements in the middle or adding elements at the front as it performs better than ArrayList. Doubly-linked list implementation of the List and Deque interfaces. Implements all optional list operations, and permits all elements (including null).  Hash Map – This implementation provides all of the optional map operations, and permits null values and the null key. (The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls.) This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time  Tree Map - This map is sorted according to the natural ordering of its keys, or by a Comparator provided at map creation time, depending on which constructor is used. This implementation provides guaranteed log(n) time cost for the containsKey, get, put and remove operations. |

2. **Answer**

1. Ignore

|  |
| --- |
| Hello  Java  Learn  World  Hello  Java  Learn  World  Hello  Java  Learn  World |



|  |
| --- |
| List list = new ArrayList(); // change code here  List list = new LinkedList(); // <- changed code  ArrayList and LinkedList differ on performance. ArrayList performs better when adding or removing at the end of the List. While LinkedList performs better when adding or removing at the beginning or in the middle of the list. Additionally, based on my research, ArrayList is better at searching (get method) than LinkedList |

4.

|  |
| --- |
| List list = new ArrayList(); // change code here  List list = new Vector(); // <- changed code  According to my research, Vectors are synchronized. Any method that touches the Vector's contents is thread-safe. ArrayList, on the other hand, is unsynchronized, making them, therefore, not thread-safe. With that difference in mind, using synchronization will incur a performance hit. So if you don't need a thread-safe collection, use the ArrayList. |

3. **Answer:**

|  |
| --- |
| Hello  Learn |

4. **Answer:**

|  |
| --- |
| 3. Compile and run well, and output 3 |

5. **Answer:**

|  |
| --- |
| See codes folder |