**1.)**

**Collection** is an interface; it is the root of all Java collection classes. We do not instantiate a collection directly but a subtype of a Collection (List, Set, and Map)

List is an ordered collection, an ArrayList is an implementation of the List interface which is optimized for retrieval/ get operations, it is a dynamically resizing array, inserting an element in the middle of an ArrayList is slow because it shifts /re indexes all the elements above.

LinkedList is faster in insertion/removal operations.

Set – is the interface of the collection for unique elements, its implementations are TreeSet, HashSet, and LinkedHashSet, TreeSet is sorted in natural order, LinkedHashSet retains the order of insertion while HashSet does not retain order.

Map – is the interface of collections which map keys to values, each key can only map to one value, implementations are HashMap – is not guaranteed to be sorted, TreeMap – sorted by keys, and LinkedHashMap – retains insertion order,

**2.)**

1.)

**for** (Object o : list) {

System.***out***.println(o);

}

2.)

Hello

Java

Learn

World

3.) List list = **new** LinkedList();

ArrayList is a resizable array, as more elements are added to an ArrayList its size grows dynamically. Its elements can be accessed using the get and set methods. Their main difference is in the implementation; operations on the LinkedList will traverse the list from beginning or end whichever is closed to the given index.

4.)

Vector list = **new** Vector();

Vector is synchronized, if one thread is working on vector, no other thread can access it. ArrayList is not synchronized which means multiple threads can work on ArrayList at the same.

**3.)**

**Hello**

**Learn**

The **public boolean remove(Object o) {**

of ArrayList removed the first occurrence of “Hello”, then, the remove(0) removed the “World”

**4.)**

**Compile and run well, and output 3**

**5.)**

1. To create a List and add three workers, and their information shown like this:

|  |  |  |
| --- | --- | --- |
| Name | Age | Salary |
| Simon | 20 | 10000 |
| Jame | 25 | 13000 |
| Alex | 22 | 12000 |

1. Add one worker before Jame ( Steven, 24, 15000)
2. Remove the worker Alex’s information
3. Go through the list using for statement and print out all the worker’s information.
4. Go through the list using Iterator statement to call all the worker’s method work.
5. Over write the equals method for the class Worker. New equals method return true only if the workers’ name, age and salary are the same at the same time.
6. **Sort the all the workers from high to low by salary** and print out the all the workers information with the format “Name: “ + name + “ Salary: “ + salary.
7. Add a id to Worker class, and save the above data to workMap. Map<String, Worker > ( Worker ID, Worker) . **At least three ways t**o go through the workMap, to print out all the workder’s information with Worker id and all other information like “Worker Id: “ + “Name: “ + name + “Age: “ + age + “ Salary: “ + salary.

**Please see**

**Worker.java and WorkerTest.java**