Ziegfreid Morissey Flameño

W2D1 Homework

1 What’s character of these Collections, Collection, List, Set and Map? ArrayList and LinkedList? HashMap and TreeMap?

**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. （List）Read the codes

**import java.util.\*;**

**public class Test {**

**public static void main(String args[]) {**

**List list = new ArrayList();**

**list.add("Hello");**

**list.add("World");**

**list.add(1, "Learn");**

**list.add(1, "Java");**

**printList(list);**

**}**

**public static void printList(List list) {**

**// 1**

**for (int i = 0; i < list.size(); i++) {**

**System.out.println(list.get(i));**

**}**

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

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

**}**

**Iterator itor = list.iterator();**

**while (itor.hasNext()) {**

**System.out.println(itor.next());**

**}**

**}**

**}**

Requirement:

1. Complete the codes at //1, and need to print out all the elements of the list.

**Answer: (ignore)**

|  |
| --- |
| System.out.println(list); |

1. Write the output of the code.

**Answer:**

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

1. Where and how to modify if change Arraylist with LinkedList? What’s the difference between ArrayList and LinkedList?

**Answer:**

|  |
| --- |
| 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 |

1. Where and how to modify if change Arraylist with Vector? What’s the difference between ArrayList and Vector?

**Answer:**

|  |
| --- |
| 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. （List）Write the output of the program.

**import** java.util.\*;

**public** **class** TestList {

**public** **static** **void** main(String args[]) {

List list = **new** ArrayList();

list.add("Hello");

list.add("World");

list.add("Hello");

list.add("Learn");

list.remove("Hello");

list.remove(0);

**for**(**int** i = 0; i < list.size(); i++) {

System.***out***.println(list.get(i));

}

}

}

**Answer:**

|  |
| --- |
| Hello  Learn |

4. Select the right one?

**import** java.util.\*;

**public** **class** TestListSet {

**public** **static** **void** main(String args[]) {

List list = **new** ArrayList();

list.add("Hello");

list.add("Learn");

list.add("Hello");

list.add("Welcome");

Set set = **new** HashSet();

set.addAll(list);

System.***out***.println(set.size());

}

}

1. Compile with error
2. Compile correctly, but throw exception when running.
3. Compile and run well, and output 3
4. Compile and run well, and output 4

**Answer:**

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

5 (List, Map)

**public** **class** Worker {

**private** **int** age;

**private** String name;

**private** **double** salary;

**public** Worker() {

}

**public** Worker (String name, **int** age, **double** salary) {

**this**.name = name;

**this**.age = age;

**this**.salary = salary;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

**public** **void** work() {

System.***out***.println(name + "is working");

}

}

Please finish the requirement:

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.

**Answer:**

|  |
| --- |
| Please see codes folder |