# CIS 143 Lab 8: Sets, maps, hashes (45 points)

Please leave the lab questions/instructions/rubrics/etc. in place. Just paste your screenshots and code below the rubric.

Purpose/knowledge/skills: Sets, maps, iterators, and hashes are core computer science concepts. You’ll work with all four in this lab.

## Task 1: Work with sets in Java (10 points)

Please enter the following code into a Java file. Remember to type instead of copy-and-paste; Word loves to mangle source code.

// Student name today’s date

import java.util.\*;

public class SetDemo {

public static void main(String args[]) {

int count[] = {34, 22,10,60,30,22};

Set<Integer> unsortedSet = new HashSet<Integer>();

try {

for(int i = 0; i < count.length; i++) {

unsortedSet.add(count[i]);

}

System.out.println("The unsorted set is: " + unsortedSet);

TreeSet sortedSet = new TreeSet<Integer>(unsortedSet);

System.out.println("The sorted set is: " + sortedSet);

System.out.println("The First element of the set is: "+ (Integer)sortedSet.first());

System.out.println("The last element of the set is: "+ (Integer)sortedSet.last());

}

catch(Exception e) {}

}

}

Thanks to <https://www.tutorialspoint.com/java/java_set_interface.htm> .

Run the program to confirm it works as expected. Then, expand the program to do the following steps in order. **For each test, display the result of the test to the console.**

1.1. Test to see if the unsorted set contains the value 20.

1.2. Test to see if the sorted set contains the value 7.

1.3. Remove the element 20 from the unsorted set.

1.4. Test to see if the unsorted set still contains the value 20.

1.5. Add the value **3** to the sorted set.

1.6. Test to see if the sorted test contains the value **3**.

1.7. Display the unsorted set.

1.8. Display the sorted set. (The sorted set should now have two more values than the unsorted set.)

1.9. Add all values from the sorted set to the unsorted set by calling unsortedSet.addAll(sortedSet).

1.10. Display both sets and confirm they have the same contents, just in a different order.

Sample program output follows.  
**Your code must use the values listed above, even if they differ from the sample output.**

Text, letter

Description automatically generated

Rubric:

* Student name and today’s date is a comment in the first line of the programs: -5 points if fails
* Screenshot and program code: -5 points if fails
* Program uses values from the problem statement, **not** the sample output: -10 points if fails
* Ten test/steps: 1 point each

Please paste a screenshot of a successful program run, and copy-and-paste the source code from your .java file, here.

|  |
| --- |
|  |
| // Kai Gillespie 20240306  **import** java.util.\*;  **public** **class** SetDemo {  **public** **static** **void** main(String args[]) {  // Array of integers to be added to the set  **int** count[] = {34, 22,10,60,30,22};    // Creating an unsorted set using HashSet  Set<Integer> unsortedSet = **new** HashSet<Integer>();  **try** {  // Adding elements from the array to the unsorted set  **for**(**int** i = 0; i < count.length; i++) {  unsortedSet.add(count[i]);  }  // Printing the unsorted set  System.***out***.println("The unsorted set is: " + unsortedSet);      // Creating a sorted set from the unsorted set using TreeSet  TreeSet<Integer> sortedSet = **new** TreeSet<Integer>(unsortedSet);    // Printing the sorted set  System.***out***.println("The sorted set is: " + sortedSet);      // Printing the first and last element of the sorted set  System.***out***.println("The First element of the set is: " +  (Integer)sortedSet.first());  System.***out***.println("The last element of the set is: " +  (Integer)sortedSet.last());    // 1.1 Check if the unsorted set contains the value 20  **boolean** contains20Before = unsortedSet.contains(20);  System.***out***.println("1.1 Unsorted set contains 20: " + contains20Before);    // 1.2 Checking if the sorted set contains the value 7  **boolean** contains7 = sortedSet.contains(7);  System.***out***.println("1.2 Sorted set contains 7: " + contains7);    // 1.3 Remove the element 20 from the unsorted set  **boolean** isRemoved = unsortedSet.remove(20);  System.***out***.println("1.3 Element 20 removed from unsorted set: " + isRemoved);  // 1.4 Test to see if the unsorted set still contains the value 20  **boolean** contains20After = unsortedSet.contains(20);  System.***out***.println("1.4 Unsorted set contains 20 after removal: " + contains20After);  // 1.5 Add the value 3 to the sorted set  **boolean** isAdded = sortedSet.add(3);  System.***out***.println("1.5 Element 3 added to sorted set: " + isAdded);  // 1.6 Test to see if the sorted set contains the value 3  **boolean** contains3 = sortedSet.contains(3);  System.***out***.println("1.6 Sorted set contains 3: " + contains3);  // 1.7 Display the unsorted set  System.***out***.println("1.7 Unsorted set: " + unsortedSet);  // 1.8 Display the sorted set  System.***out***.println("1.8 Sorted set: " + sortedSet);  // 1.9 Add all values from the sorted set to the unsorted set  unsortedSet.addAll(sortedSet);  // 1.10 Display both sets and confirm they have the same contents  //Display  System.***out***.println("1.10 Unsorted set after adding all from sorted set: " + unsortedSet);  System.***out***.println("1.10 Sorted set after adding all from sorted set: " + sortedSet);    //Confirm  **boolean** sameContents = unsortedSet.containsAll(sortedSet) && sortedSet.containsAll(unsortedSet);  System.***out***.println("1.10 Do both sets have the same contents? " + sameContents);    }  **catch**(Exception e) {}  }  } |

## Task 2: Work with maps in Java (10 points)

Please run the following code in Java, and then answer the following questions. Remember to type instead of copy-and-paste; Word loves to mangle source code.

// Student name today’s date

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

**public** **class** MapTest {

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

Map<Integer, String> map = **new** HashMap<Integer, String>();

map.put(2, "Marty");

map.put(30, "Louann");

map.put(27, "Nguyen");

map.put(15, "Moshe");

map.put(84, "Larry");

map.put(2, "Ed");

map.put(2350, "Orlando");

map.remove(8);

map.put(5, "Moshe");

map.remove(84);

map.put(17, "Steve");

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

}

}

2.1. What keys are in this map after the program runs?

|  |
| --- |
| 2, 30, 27, 15, 2350, 5, and 17 |

2.2. What values are contained in the map after the program runs?

|  |
| --- |
| "Ed", "Louann", "Nguyen", "Moshe", "Orlando", "Moshe", and "Steve".  "Moshe" appears twice because it's associated with two different keys, 15 and 5. |

2.3. There are two “put” commands with key 2. Why is there only one value for key 2 stored in the map?

|  |
| --- |
| In a HashMap, each key can be associated with exactly one value. When the put method is called with a key already in the map, the new value replaces the old value for that key. |

2.4. Are the values in a HashMap stored in sorted order?

|  |
| --- |
| No, the values in a HashMap are not stored in sorted order. |

2.5. When you issue a “put” command with a key that is already in the HashMap, Java returns the old value. Do you have to do anything with this returned value?  
(Hint: is the return value from map.put(7, "Ed"); assigned or output in the above program?)

|  |
| --- |
| No, you don’t have to do anything with the returned value if it’s not needed for anything. |

Rubric:

* Student name and today’s date is a comment in the first line of the programs: -5 points if fails
* Screenshot and program code: -5 points if fails
* Five questions: 10 points total, 2 point each

Please paste a screenshot of a successful program run, and copy-and-paste the source code from your .java file, here.

|  |
| --- |
|  |
| // Kai Gillespie 20240306  **import** java.util.\*;  **public** **class** MapTest {  **public** **static** **void** main(String args[]) {  Map<Integer, String> map = **new** HashMap<Integer, String>();  map.put(2, "Marty");  map.put(30, "Louann");  map.put(27, "Nguyen");  map.put(15, "Moshe");  map.put(84, "Larry");  map.put(2, "Ed");  map.put(2350, "Orlando");  map.remove(8);  map.put(5, "Moshe");  map.remove(84);  map.put(17, "Steve");  System.***out***.println(map);  }  } |

## Task 3: Answer questions about iterators (5 points total, 1 point per answer)

3.1. In your own words, describe what each iterator method does:

|  |
| --- |
| hasNext(): Returns true if the iteration has more elements.  next(): Retrieves the next element in the iteration.  remove (): Removes the last element that was returned by the next() method. |

3.2. If you have a Collection variable myCollection, containing strings, how would you use a for loop to display the contents of myCollection?

Hint: for (String word: myCollection) { // your code here }

|  |
| --- |
| // Kai Gillespie 20240306  for (String word : myCollection) {  System.out.println(word);  } |

3.3. Since an ArrayList, Linked List, HashSet, and TreeSet all implement the Collection interface, the methods in this section work for variables of any of those data types. Which of the following does that demonstrate?

**Inheritance** Subnetting Encapsulation

(The other two are networking terms. X-D )

## Task 4: Work with hashes (10 points)

Hash a list of employee name/number combinations.

Hashing function h1(x) = the **last** three digits of the employee number

Compression function h2(x) = the value mod 10  
 (Hint: this is the last digit of the value coming in.)

Index into the array: h2(h1(x))

|  |  |  |
| --- | --- | --- |
| Key and value | Hash code  h1(x)  last three digits | Array index  h2(h1(x))  h1(x) % 10 |
| **Haen324350** | 350 | 0 |
| **Gurj313235** | 235 | 5 |
| **Durd313553** | 553 | 3 |
| **Putt125432** | 432 | 2 |
| **Grog131354** | 354 | 4 |
| **Rami967999** | 999 | 9 |

When complete, please put the values into the array according to the hash index

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Value | Haen324350 |  | Putt125432 | Durd313553 | Grog131354 | Gurj313235 |  |  |  | Rami967999 |

Rubric:

* H1(x): 4 points
* H2(h1(x)): 4 points
* Array values: 2 points

## Task 5: Work with hash collisions (10 points)

Hash a list of employee name/number combinations.

Hashing function h1(x) = the **first** three digits of the employee number

Scaling function h2(x) = the value mod 10   
 The easiest way to calculate this: it’s the last digit of the value coming in

Index into the array: h2(h1(x))

|  |  |  |
| --- | --- | --- |
| Key and value | Hash code  h1(x)  first three digits | Array index  h2(h1(x))  h1(x) % 10 |
| **Haen324350** | 324 | 4 |
| **Gurj313235** | 313 | 3 |
| **Durd313553** | 313 | 3 |
| **Putt125432** | 125 | 5 |
| **Grog131354** | 131 | 1 |
| **Rami967999** | 967 | 7 |

When complete, please put the values into the array according to the hash index.   
In case of collisions, use **linear probing**.

1. Haen324350 assigned to index 4.
2. Gurj313235 assigned to index 3.
3. Durd313553 results in a collision with Gurj313235 and moves to the next available index (5).
4. Putt125432 results in a collision with Putt125432 and moves to the next available index (6).
5. Grog131354 assigned to index 1.
6. Rami967999 assigned to index 7.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Value |  | Grog131354 |  | Gurj313235 | Haen324350 | Durd313553 | Putt125432 | Rami967999 |  |  |

Rubric:

* H1(x): 3 points
* H2(h1(x)): 3 points
* Array values: 4 points