Hyperion Development

Task: HashMaps and Sets

www.hyperiondev.com

Introduction

Collecting information

In this task you are going to learn about the collections framework in java. This is essential component which gives you access to multiple libraries and data structures to enhance the functionality of your program. Within this framework, we'll focus on learning about the structure and implementations of HashMaps and sets, whilst analysing what makes them similar as well as unique. You've previously dealt with ArrayLists and LinkedLists so the structure and operations may seem similar to an extent. This kind of analysis is key to have the most optimised program with the best code implementation. Let's get started!

-The Hyperion Team

Start collecting

Once again we'll be looking at the java.util package. This important package contains a large assortment of classes and interfaces that support a broad range of functionality. For example, java.util has classes that generate pseudorandom numbers, manage date and time, observe events, manipulate sets of bits, tokenize strings, and handle formatted data. The java.util package also contains one of Java's most powerful subsystems: the Collections Framework. The Collections Framework is a sophisticated hierarchy of interfaces and classes that provide state-of-the-art technology for managing groups of objects. It merits close attention by all programmers.

Now that you are familiar with the collection interface, you are ready to examine the standard classes that implement them. Some of the classes provide full implementations that can be used as-is. Others are abstract, providing skeletal implementations that are used as starting points for creating concrete collections.

HashMaps

Arrays and Lists store elements as ordered collections, with each element given an integer index. HashMap is used for storing data collections as key and value pairs. One object is used as a key (index) to another object (the value). The put, remove, and get methods are used to add, delete, and access values in the HashMap:

```
import java.util.HashMap;
public class MyClass {
  public static void main(String[ ] args) {
    HashMap<String, Integer> points = new HashMap<String, Integer>();
```

Above we created a HashMap with Strings as its keys and Integers as its values.

Note: *Use the get method and the corresponding key to access the HashMap elements.*

A HashMap cannot contain duplicate keys. Adding a new item with a key that already exists overwrites the old element. The HashMap class provides containsKey and containsValue methods that determine the presence of a specified key or value. If you try to get a value that is not present your map, it returns the value of null.

Note: null *is a special type that represents the absence of a value.*

HashMap Class Methods

1. **void clear():** It removes all the key and value pairs from the specified Map.

2. **Object clone():** It returns a copy of all the mappings of a map and is used for cloning them onto another map.

3. **boolean containsKey(Object key):** It is a boolean function which returns true or false based on whether the specified key is found in the map.

4. **boolean containsValue(Object Value):** Similar to containsKey() method, however it looks for the specified value instead of the key.

5. **Value get(Object key):** It returns the value for the specified key.

6. **boolean isEmpty():** It checks whether the map is empty. If there are no key-value mapping present in the map then this function returns true, else false.

7. **Set keySet():** It returns the Set of the keys fetched from the map.

8. **value put(Key k, Value v):** Inserts key value mapping into the map.

9. **int size():** Returns the size of the map – the number of key-value mappings.

10. **Collection values():** It returns a collection of values of map.

11. **Value remove(Object key):** It removes the key-value pair for the specified key.

12. **void putAll(Map m):** Copies all the elements of a map to the another specified map.

Sets

A Set is a collection that cannot contain duplicate elements. It models the mathematical set abstraction. One of the implementations of the Set is the HashSet class:

```
import java.util.HashSet;

public class MyClass {
  public static void main(String[ ] args) {
    HashSet<String> set = new HashSet<String>();
```

Note: *You can use the size() method to get the number of elements in the HashSet.*

HashSet Class Methods

1. **boolean add(Object o):** Adds the specified element to this set if it is not already present.

2. **void clear():** Removes all of the elements from this set.

3. **Object clone():** Returns a shallow copy of this HashSet instance: the elements themselves are not cloned.

4. **boolean contains(Object o):** Returns true if this set contains the specified element

5. **boolean isEmpty():** Returns true if this set contains no elements.

6. **Iterator iterator():** Returns an iterator over the elements in this set.

7. **boolean remove(Object o):** Removes the specified element from this set if it is present.

8. **int size():** Returns the number of elements in this set (its cardinality).

Some unique features of the HashSet class

1. HashSet doesn't maintain any order, the elements would be returned in any random order.

2. HashSet doesn't allow duplicates. If you try to add a duplicate element in HashSet, the old value would be overwritten.

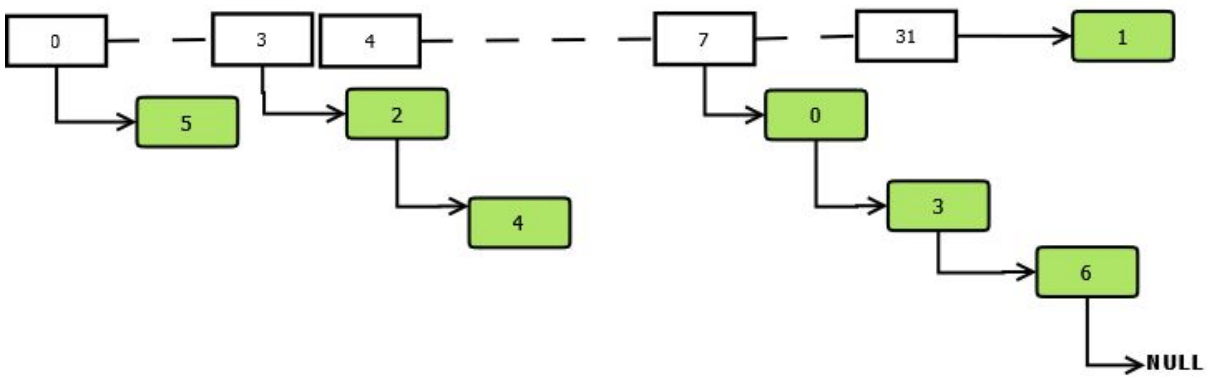
3. HashSet allows null values, however if you insert more than one nulls it would still return only one null value.

4. HashSet is non-synchronized.

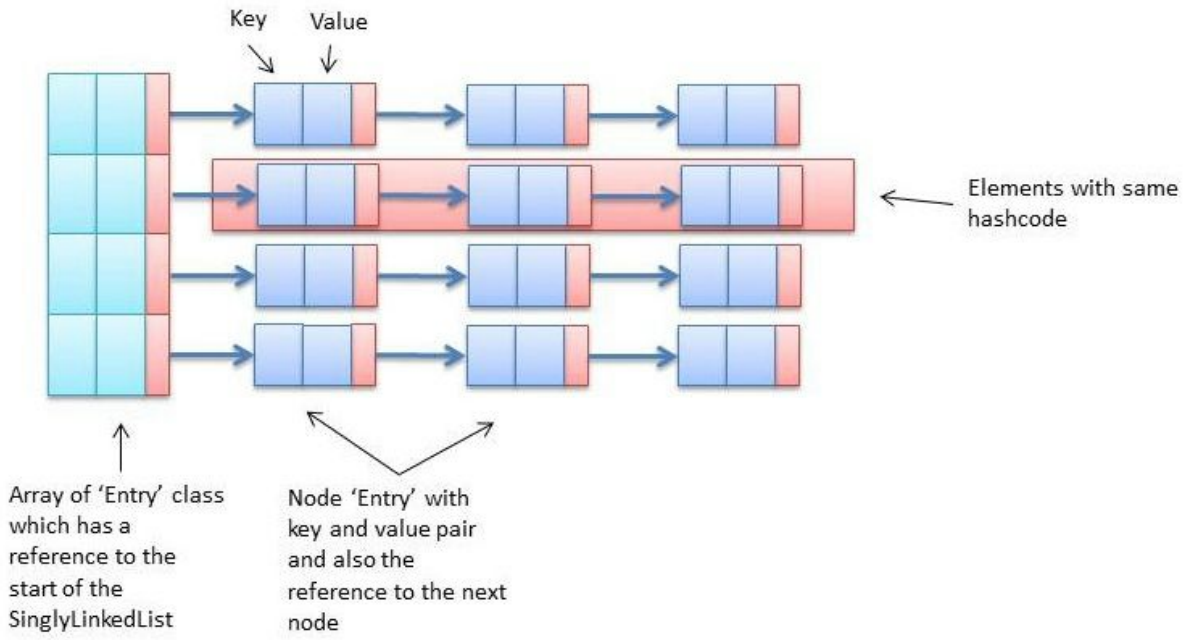
5. The iterator returned by this class is fail-fast which means iterator would throw ConcurrentModificationException if HashSet has been modified after creation of iterator, by any means except iterator's own remove method.

Data structure of the HashMap vs. HashSet

HashMap:



HashSet:



Instructions

First read example.java, open it using jGRASP (Right click the file and select 'Open with jGRASP").

- In this folder there is a file called example.java
- Open this folder using the JGRASP program. You can either do this by right clicking on the example.java file, going to the 'Open with' menu, and selecting JGrasp.exe. Alternatively, you can run the JGRASP program on your computer, go to the top left corner of the program, select File->Open and navigate to example.java on your hard drive and double click on it to open it.
- Once example.java is open in JGRASP please read all of its content very carefully. This will help you understand the basic structure of a Java program.
- There is a compulsory task at the end of the example.java document. The instructions of what to do to complete the task can be found here. You must complete this exercise to continue onto the next task of this course.

Compulsory Task 1

Follow these steps:

- Create a new java file called Restaurant.java
- Import the java.util.HashMap package.
- Create a public class called 'Restaurant'.'
- Within this class, create a main method.
- Within this method, create a HashMap called "RestaurantMenu' with Strings for keys and Integers for values.
- Add 10 meal elements to the menu, with their prices mapped as the values.
- Print out the keys to the user (picture a menu).
- Present the user with the option to order, or pay.
- If order, present the list and option again, whilst summing the prices of what they order (note the user can only order one item at a time).
- Print out the total of their order.
- Compile, save and run your file.

Compulsory Task 2

Follow these steps:

- Create a new java file called Buffet.java
- Import the java.util.HashSet package.
- Create a public class called 'Buffet'.
- Within this class, create a main method.
- Ask a user to input what they would eat at a buffet.
- Append the item to the HashSet.
- Keep asking and appending items, until the user enters "nothing".
- Print out the size of the HashSet.
- Print out the HashSet (notice what happened to any repeated entries?).
- Compile, save and run your file.

Things to look out for

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **jGRASP or Java** may not be installed correctly.
2. If you are not using Windows, please ask your tutor for alternative instructions.

Still need help?

Just write your queries in your comments.txt file and your tutor will respond.

Task Statistics

Last update to task: 04/02/2016.
Authors: Jared Ping.
Main trainer: Umar Randeree.
Task Feedback link: [Hyperion Development Feedback.]