Lecture#7

Object Oriented Programming (JAVA)

Dr. Abu Nowshed Chy

Department of Computer Science and Engineering

University of Chittagong

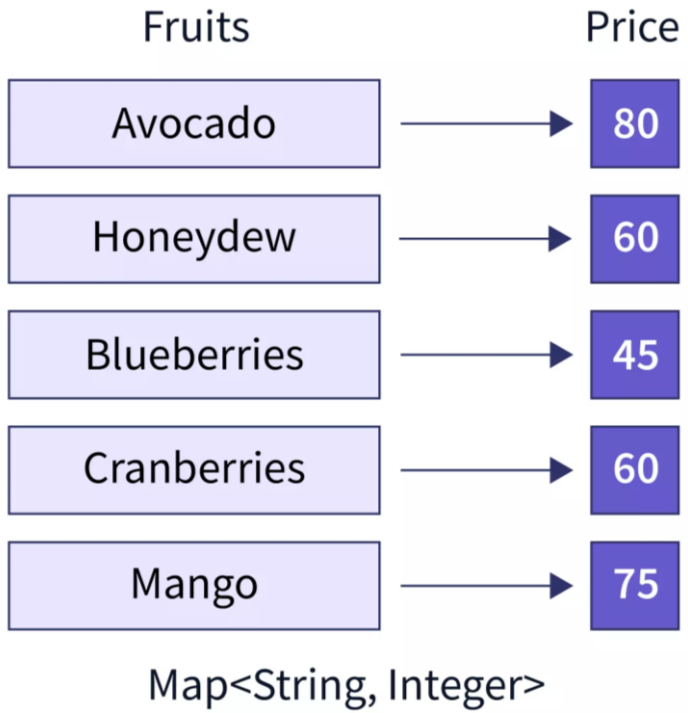
February 01, 2024

Faculty Profile

Map

- **Map**: Holds a set of unique *keys* and a collection of *values*, each associated with one value.
  - a.k.a. "dictionary", "associative array", "hash"

- Basic map operations:
  - **put**(*key*, *value*): Adds a mapping from a key to a value.
  - **get**(*key*): Retrieves the value mapped to the key.
  - **remove**(*key*): Removes the given key and its mapped value.

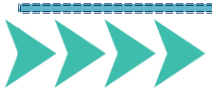


`myMap.get("Mango")` returns 75

Map

- Map is implemented by the HashMap and TreeMap classes
  - HashMap: implemented using an array called a "hash table"; extremely fast: O(1) ; keys are stored in unpredictable order
  - TreeMap: implemented as a linked "binary tree" structure; very fast: O(log N) ; keys are stored in sorted order

```
// maps from String keys to Integer values
Map<String, Integer> votes = new HashMap<String, Integer>();

// maps from String keys to Integer values
Map<String, Integer> votes = new TreeMap<String, Integer>();
```



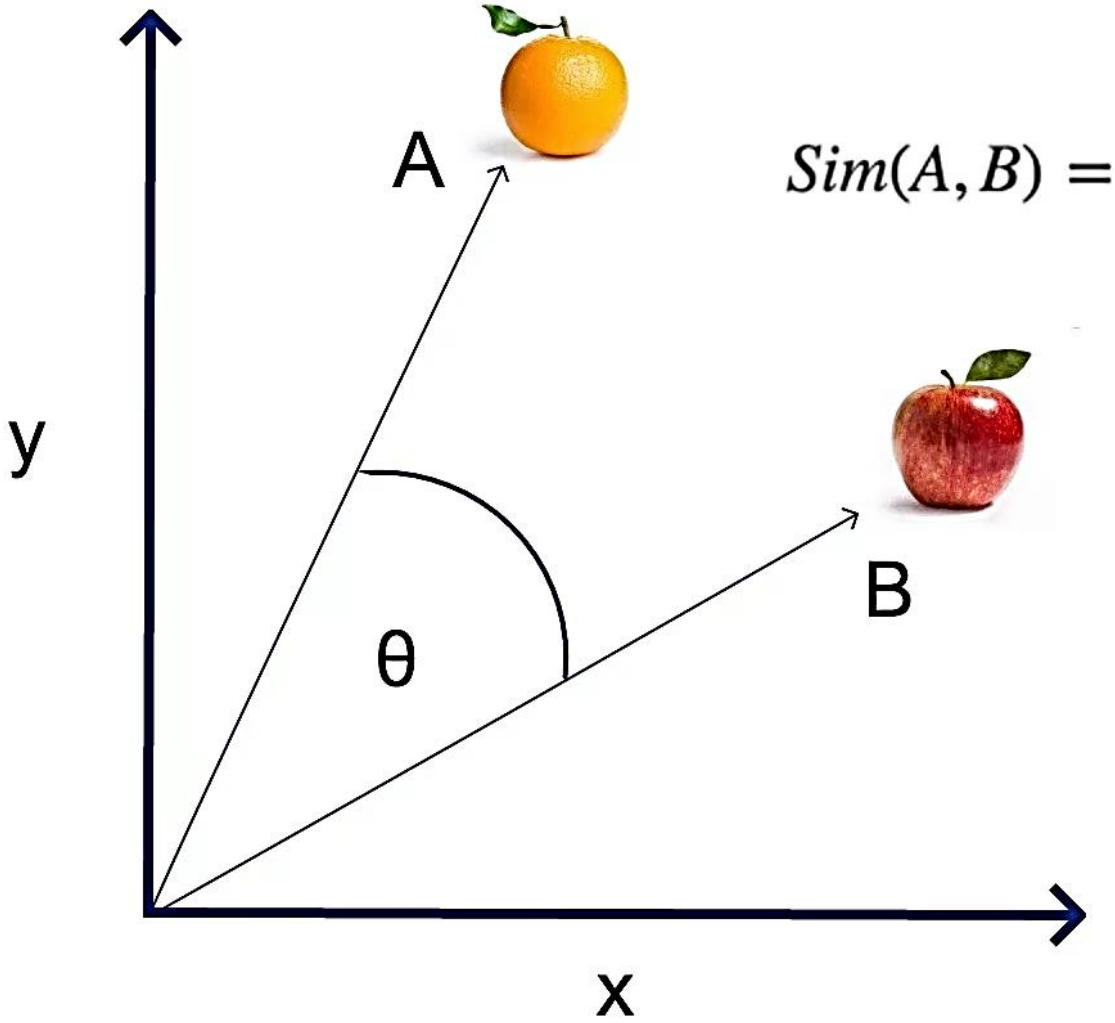
Map

`put(`**key, value**`)`	adds a mapping from the given key to the given value; if the key already exists, replaces its value with the given one
`get(`**key**`)`	returns the value mapped to the given key (`null` if not found)
`containsKey(`**key**`)`	returns `true` if the map contains a mapping for the given key
`remove(`**key**`)`	removes any existing mapping for the given key
`clear()`	removes all key/value pairs from the map
`size()`	returns the number of key/value pairs in the map
`isEmpty()`	returns `true` if the map's size is 0
`toString()`	returns a string such as `"{a=90, d=60, c=70}"`
`keySet()`	returns a set of all keys in the map
`values()`	returns a collection of all values in the map
`putAll(`**map**`)`	adds all key/value pairs from the given map to this map
`equals(`**map**`)`	returns `true` if given map has the same mappings as this one

Map Example

```
package Similarity;

import java.util.HashMap;
import java.util.Map;

public class MapSampleCode {
    public static void main(String[] args) {
        Map<String,Integer> sampleMap=new HashMap<String,Integer>();
        sampleMap.put("Std1", 60);
        sampleMap.put("Std2", 70);
        sampleMap.put("Std3", 80);
        sampleMap.put("Std4", 90);

        System.out.println(sampleMap);

        System.out.println(sampleMap.get("Std1"));

        for(String mapKey:sampleMap.keySet()){
            System.out.println(mapKey);
        }
        for(String mapKey:sampleMap.keySet()){
            System.out.println(sampleMap.get(mapKey));
        }
    }
}
```

Cosine Similarity



$$Sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

Cosine Similarity

Document 1 = 'the best data science course'

Document 2 = 'data science is popular'

Cosine Similarity

$$D1 = [1, 1, 1, 1, 1, 0, 0]$$

$$D2 = [0, 0, 1, 1, 0, 1, 1]$$

$$D1 \cdot D2 = 1 \times 0 + 1 \times 0 + 1 \times 1 + 1 \times 1 + 1 \times 0 + 0 \times 1 + 0 \times 1 = 2$$

$$\|D1\| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2} = \sqrt{5}$$

$$\|D2\| = \sqrt{0^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 1^2} = \sqrt{4}$$

Cosine Similarity

$$similarity(D1, D2) = \frac{D1 \cdot D2}{\|D1\|\|D2\|} = \frac{2}{\sqrt{5}\sqrt{4}} = \frac{2}{\sqrt{20}} = 0.44721$$

The angle between the vectors is calculated as:

$$cos(\theta) = 0.44721$$

$$\theta = \arccos(0.44721) = 63.435$$

Cosine Similarity

```
public class SimilarityEstimation {
    public static void main(String[] args) {
        String testString1 = "the best data science course in the Engg course";

        Map<String,Integer> word_freq=new HashMap<String,Integer>();
        Integer count = null;

        String delims = " ";
        String[] token=testString1.split(delims);
        String word;

        for(int i=0; i<token.length; i++){
          word = token[i];
          count = word_freq.get(word);

          if (count == null){
              count = 1;
              }
          else {
              count = count + 1;
              }
          word_freq.put(word, count);
        }
        System.out.println(word_freq);
    }
}
```

Cosine Similarity

```
public class SimilarityEstimation2 {
    public static void main(String[] args) {
        String testString1 = "the best data science course in the Engg course";
        SimilarityDescriptor similarityObj = new SimilarityDescriptor();
        Map<String,Integer> word_freq = similarityObj.wordFreqEstimation(testString1);
        System.out.println(word_freq);
    }
}
```

Word Frequency Estimation

```
class SimilarityDescriptor {
    Map<String,Integer> wordFreqEstimation(String getString){
        Map<String,Integer> word_freq=new HashMap<String,Integer>();
        Integer count = null;

        String delims = " ";
        String[] token=getString.split(delims);
        String word;

        for(int i=0; i<token.length; i++){
          word = token[i];
          count = word_freq.get(word);

          if (count == null){
              count = 1;
              }
          else {
              count = count + 1;
              }
          word_freq.put(word, count);
        }
        return word_freq;
    }
}
```

Cosine Similarity Estimation

```
Double cosineSimilarity(String text1, String tex2){
    Map<String, Integer> docfreq1 = wordFreqEstimation(text1);
    Map<String, Integer> docfreq2 = wordFreqEstimation(tex2);

    Double cosine_similarity;
    double mul=0.0f;
    double fr1=0.0f;
    double fr2=0.0f;

    for(String key1:docfreq1.keySet()){
      fr1=fr1+Math.pow(docfreq1.get(key1),2);
    }
    for(String key2:docfreq2.keySet()){
      fr2=fr2+Math.pow(docfreq2.get(key2),2);
    }

    for(String key1:docfreq1.keySet()){
        if (docfreq2.containsKey(key1)){
            mul=mul+docfreq1.get(key1)*docfreq2.get(key1);
          }
    }
    cosine_similarity=(Double) (mul/Math.sqrt(fr1*fr2));
    return cosine_similarity;
}
```

Map Practice Problem#1

Write a Java method that takes two binary string as input and estimate their simple matching coefficient (SMC) score as follows:

$\mathbf{x} = 1\,0\,0\,0\,0\,0\,0\,0\,0\,0$

$\mathbf{y} = 0\,0\,0\,0\,0\,0\,1\,0\,0\,1$

$f_{01} = 2$ (the number of attributes where p was 0 and q was 1)

$f_{10} = 1$ (the number of attributes where p was 1 and q was 0)

$f_{00} = 7$ (the number of attributes where p was 0 and q was 0)

$f_{11} = 0$ (the number of attributes where p was 1 and q was 1)

$$\text{SMC} = (f_{11} + f_{00}) / (f_{01} + f_{10} + f_{11} + f_{00})$$
$$= (0+7) / (2+1+0+7) = 0.7$$

Map Practice Problem#2

Consider the following two documents:

doc_1 = "Data is the new oil of the digital economy"

doc_2 = "Data is a new oil"

The Jaccard similarity of these two documents is estimated as follows:

$$J(doc_1, doc_2) = \frac{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cap \{'data', 'is', 'a', 'new', 'oil'\}}{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cup \{'data', 'is', 'a', 'new', 'oil'\}}$$

$$= \frac{\{'data', 'is', 'new', 'oil'\}}{\{'data', 'a', 'of', 'is', 'economy', 'the', 'new', 'digital', 'oil'\}}$$

$$= \frac{4}{9} = 0.444$$

Write a Java method that takes two string as input and return their Jaccard similarity Score.


Thank You!