At this lab section, we will learn how to implement hash tables in Java using OOP principles.

Hash Tables

Asst. Prof. Dr. Feriştah DALKILIÇ
Res. Asst. Fatih DİCLE

PART 1 – Hash Tables

Hash tables are an efficient method of storing a small number, n of integers from a large range

U= {0, …… , k}.

Hash function which is used to map a given value with a particular key for storing and faster access of elements. Access of elements becomes very fast if we know the key of the desired data. The efficiency of mapping depends on the efficiency of the hash function used.

It becomes a data structure in which insertion and search operations are very fast irrespective of the size of the data.

Hashing is a technique to convert a range of key values into a range of indexes of an array. Let's assume we have an array called "A" and hash table like the following.

A = [85, 12, 21, 43, 14, 30]

0	30
1	21
2	12
3	43
4	14
5	85
6	

We can design a hash function as follows.

h(x) = x mod 10

h(85) = 5 h(12) = 2 h(21) = 1

h(43) = 3 h(14) = 4 h(30) = 0

Exercise - 1

At this section, we will create a hash table and design a hash function. A new element **will not be inserted** when a collision occurs. Suppose we have an array called "B" as follows.

B = [90, 27, 19, 16, 13, 77, 43, 35, 53, 28]

Step – 1

We have a hash table of size 11.

0	77
1	
2	90
3	
4	
5	27
6	28
7	
8	19
9	53
10	43

Step – 2

You will create a hash function that performs mod by table size, calculate hash indexes of items and insert items into the hash table if no collision occurs.

Your Answer	
h(x) = x % 11 h(90) = 2 h(27) = 5 h(19) = 8 h(16) = 5 h(13) = 2 h(77) = 0 h(43) = 10 h(35) = 2 h(53) = 9 h(28) = 6	0 77 1 2 90 3 4 5 27 6 28 7 8 19 9 53 10 43

PART 2 – Implementation in Java

At this section, we will implement a hash table in Java in two different ways. Firstly, Java's HashMap class will be used, then an array-based hash table technique will be applied.

The main methods of the HashMap class are as shown in the table.

Method	**Description**
put (Object key, Object value)	Associates the specified value with the specified key in this map
get(Object key)	Returns the value to which the specified key is mapped in this identity hash map
remove(Object key)	Removes the mapping for this key from this map if present
keySet()	Returns a set view of the keys contained in this map
values()	Returns a collection view of the values contained in this map

Exercise – 2

In this exercise, you are expected to experiment HashMap class in *java.util* library

Step – 1

Create a new Java Project. Add a class with the name "Test.java" and paste following code. This example matches the countries and their capitals by using `java.util.HashMap`.

Test. Java

```
import java.util.HashMap;

public class Test {

      public static void main(String[] args)
      {
            // Create a HashMap object called capitalCities
            HashMap<String, String> capitalCities = new
HashMap<String, String>();

            // Add keys and values (Country, City)
            capitalCities.put("Turkey", "Ankara");
            capitalCities.put("England", "London");
            capitalCities.put("Germany", "Berlin");
            capitalCities.put("Norway", "Oslo");
            capitalCities.put("USA", "Washington DC");

            System.out.println(capitalCities);

            // Get value
            System.out.println(capitalCities.get("Turkey"));

            // Remove item by key
            System.out.println(capitalCities.remove("Germany"));

            System.out.println(capitalCities);

            // Print keys
            for (String i : capitalCities.keySet()) {
              System.out.println(i);
            }

            // Print values
            for (String i : capitalCities.values()) {
              System.out.println(i);
            }
      }// end of main
}// end of class
```

Step – 2

Paste your output.

Your Output
{USA=Washington DC, Turkey=Ankara, Norway=Oslo, England=London, Germany=Berlin} Ankara

```
Berlin
{USA=Washington DC, Turkey=Ankara, Norway=Oslo, England=London}
USA
Turkey
Norway
England
Washington DC
Ankara
Oslo
London
```

Exercise - 3

In this exercise, you are expected to use `java.util.HashMap` for storing student records.

Step – 1

Create a HashMap object and insert students and grades as key-value pairs.

Student	Grade
Arda	60
Duygu	75
Ceren	65
Berk	80
Kaan	95
Utku	50

Step – 2

Print the students and grades inserted to HashMap.

Step – 3

Print the student with highest grade in HashMap.

Step – 4

Print the grade average in HashMap.

Step – 5

Paste your code and output.

Your Code

```
import java.util.HashMap;
public class Main {

   public static void main(String[] args)
   {
      // Create a HashMap object called capitalCities
      HashMap<String, Integer> capitalCities = new HashMap<String, Integer>();
      // Add keys and values (Country, City)
      capitalCities.put("Arda",60);
      capitalCities.put("Duygu",75);
      capitalCities.put("Ceren",65);
      capitalCities.put("Berk",80);
```

```
        capitalCities.put("Kaan",95);
        capitalCities.put("Utku",50);
        System.out.println(capitalCities);
        int maxStudentGrade = 0;
        int gradesSum = 0;
        for (String student: capitalCities.keySet()) {
            int grade = capitalCities.get(student);
            if (grade > maxStudentGrade) {
                maxStudentGrade = grade;
            }
            gradesSum += grade;
        }
        // Print the student with the highest grade in HashMap.
        System.out.println(maxStudentGrade);
        // Print the grade average in HashMap.
        System.out.println((float)gradesSum /
                (float)capitalCities.size());
    }// end of main
}// end of class
```

Your Output
{Kaan=95, Arda=60, Utku=50, Duygu=75, Berk=80, Ceren=65}
95
70.833336

Exercise - 4

You are given a simple implementation of an array-based hash table with missing parts. You will fill in the required fields and test your algorithm.

Step – 1

Create a class called HashEntry to store key and value.

HashEntry. java

```
public class HashEntry {

    private int key;
    private int value;

    HashEntry(int key, int value) {
        this.key = key;
        this.value = value;
    }

    public int getKey() {
        return key;
    }

    public int getValue() {
        return value;
    }
}
```

Step – 2

Create HashTable class and implement the hash function, put and get methods.

HashTable. java

```
public class HashTable {

    private final static int TABLE_SIZE = 128;

    HashEntry[] table;

    public HashTable() {
        table = new HashEntry[TABLE_SIZE];
        for (int i = 0; i < TABLE_SIZE; i++)
            table[i] = null;
    }

    public int hashFunction(int key) {

            // Create a hash function that performs mod by table size

    }

    public int get(int key) {

             int hash = // Calculate hash value

        if (table[hash] == null)
            return -1;
        else
            return table[hash].getValue();

    }

    public void put(int key, int value) {

        int hash = // Calculate hash value

        // Print "There is a collision !" message to indicate collision and do not insert item

        table[hash] = new HashEntry(key, value);

    }

}
```

Step – 3

Create a class with the name “Test.java” and paste the following code to evaluate your hash table.

Test. java

```
public class Test {

        public static void main(String[] args)
        {
            HashTable hashTable = new HashTable();

                    hashTable.put(201790002, 60);
```

```
                hashTable.put(201690002, 80);
                hashTable.put(201693673, 75);
                hashTable.put(201690002, 85);
                hashTable.put(201690002, 70);


                System.out.println(hashTable.get(201690002));
                System.out.println(hashTable.get(201693673));
        }

}
```

Step – 3

Paste your code and output.

Your HashTable. java

```
public class HashTable {

    private final static int TABLE_SIZE = 128;

    HashEntry[] table;

    public HashTable() {
        table = new HashEntry[TABLE_SIZE];
        for (int i = 0; i < TABLE_SIZE; i++)
            table[i] = null;
    }

    public int hashFunction(int key) {
        return key%TABLE_SIZE;
    }

    public int get(int key) {

        int hash = hashFunction(key);// Calculate hash value

        if (table[hash] == null)
            return -1;
        else
            return table[hash].getValue();

    }

    public void put(int key, int value) {

        int hash = hashFunction(key);
        if (table[hash]!=null)
            System.out.println("Collision appeared");
        else{
            table[hash] = new HashEntry(key,value);
        }

    }

}
```

Your Output
Collision appeared Collision appeared 80 75