Lab 5

CST 338

Part 1 – The ArrayList Class

Define a class called `ArrayListDemo` which creates an `ArrayList` of Strings called `myList`. The beginning of the program is based on the `ArrayList` lecture (Lecture 16), slides 28–30.

ArrayListDemo Run

```
Input an entry:
hw1
More items? (y/n)
y
Input an entry:
hw2
More items? (y/n)
n
The list contains:
hw1
hw2
An item to search:
hw2
hw2 is in the list
An item to remove:
hw1
The list contains:
hw2
```

ArrayListDemo Code Excerpts

```java
//display
System.out.println("The list contains:");
for (String entry : myList) {
  System.out.println(entry);
}
```

```
//search
System.out.println("An item to search: ");
answer = keyboard.nextLine();
if (myList.contains(answer)) {
  System.out.printf("%s is in the list%n", answer);
} else {
  System.out.printf("%s is not in the list%n", answer);
}

//remove
System.out.print("An item to remove: ");
answer = keyboard.nextLine( );
if (myList.remove(answer)) {
  System.out.printf("%s is removed from the list.%n", answer);
} else {
  System.out.printf("%s was not in the list.%n", answer);
}
```

Object Parameter Type

- Note that some methods of the `ArrayList` class have a formal parameter of type `Object`.
- For these methods, the `equals()` method may be very important.

Example – Toy Class

A class with a number (`int`) and a name (`String`).

```
public class Toy {
  private int number;
  private String name;

  public Toy(int number, String name) {
    this.number = number;
    this.name = name;
  }
}
```

Driver Program – Execution Result?

ArrayList’s `contains()` method uses `equals()` to check if two objects are the same.

```java
public static void main(String[] args) {
  ArrayList<Toy> list = new ArrayList<Toy>();

  Toy tom100 = new Toy(100, "Tom");
  Toy tom200 = new Toy(200, "Tom");

  list.add(tom100);
  list.add(tom200);

  Toy tom100_2 = new Toy(100, "Tom");

  if (list.contains(tom100_2)){
    System.out.println("Found");
  } else {
    System.out.println("Not found");
  }
}
```

equals() method in ArrayList

- The `equals()` method is used to check for equality.
- A proper implementation of `equals()` is important when you use the `ArrayList` class.

`equals()` method – Overrides Object's `equals()` method

```java
@Override
public boolean equals(Object obj) {
  // checks if the passed object is of type Toy
  if (obj instanceof Toy) {
    // compares the two Toy objects
    // (casts the passed in object as Toy)
    Toy other = (Toy) obj;
    return ((this.number == other.number) &&
            (this.name.equals(other.name)));
  } else {
    return false;
  }
}
```

equals() in ArrayList

- References
  - How to Write an Equality Method in Java
  - Java Collections - hashCode() and equals()

Exercise

- Assume that you are developing an `Employee` class with `idNum`, `firstName`, and `lastName`.
  - Now, you want to keep the `Employee` objects in an `ArrayList`.
  - Develop the correct `equals()` method of the `Employee` class.

Lab Sample Run

```
Input a number:
100
Input a name:
Tom
More items? (y/n)
y
Input a number:
200
Input a name:
John
More items? (y/n)
n
The list contains:
100 Tom
200 John
Type the number and name to search
Number:
100
Name:
Tom
100 Tom is in the list
More search? (y/n)
n
Number:
100
```

```
Name:
Tom
100 Tom is removed from the list
The list contains:
200 John
More remove? (y/n)
n
```

Part 2 – The HashMap Class

Map

A map is a collection that contains **values** that are associated with **keys**.

Example

- Student's ID (= key) and the student's academic record (=value)

Sample Usage of HashMap

- Assume that you want to store a student's information for a registration system.
  - We probably have a student's ID (=key) and the student's other information, such as a name, degree, address, etc. (=value)
- Another example could be a bank application with many accounts.
  - Key: an account number
  - Value: account's other information for the account number.

Hash Table

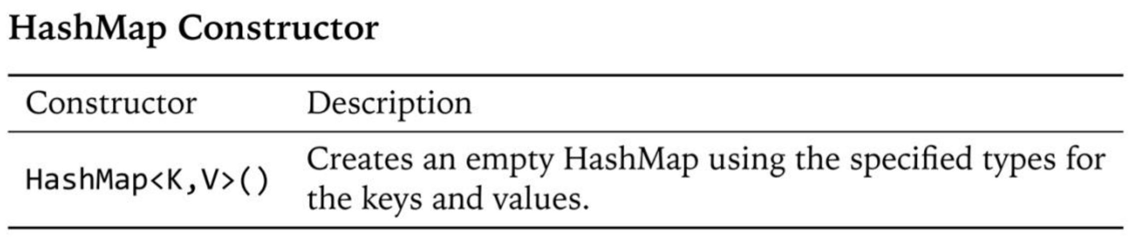
The *hash table* is a data structure to find a value very quickly for a key. See the Wikipedia article on Hash table (or look elsewhere) for more information.

HashMap Class

Typically, the key is a simple type such as a `String` or `Integer` and the value is a user-defined class such as an `Account` class.

- If you want to use your own class as a key, your class must override the `hashCode()` and `equals()` methods inherited from `Object`.

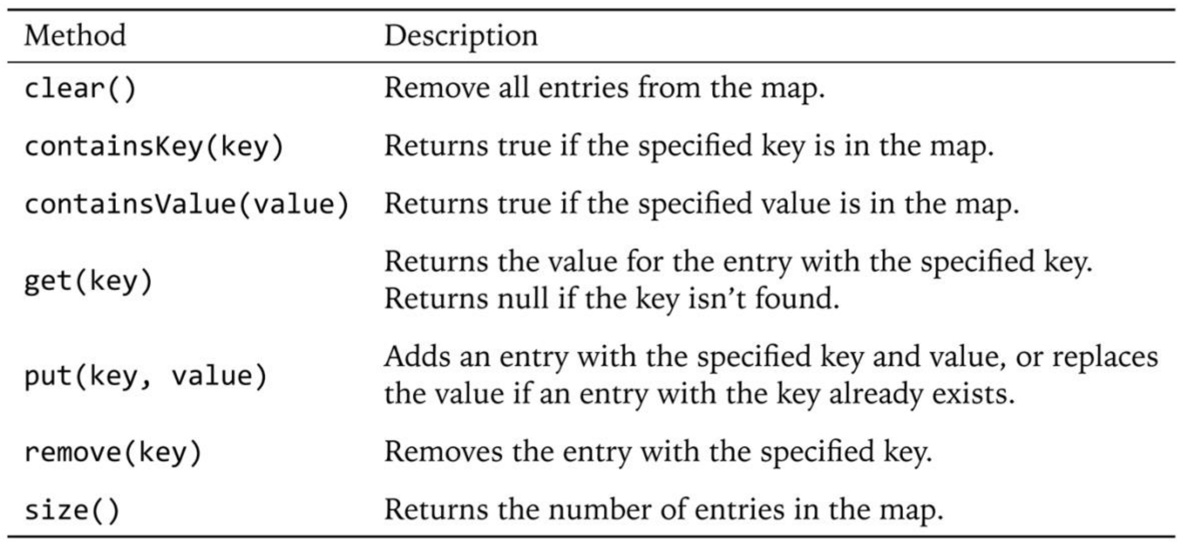
HashMap Constructor



HashMap Constructor

Constructor	Description
`HashMap<K,V>()`	Creates an empty HashMap using the specified types for the keys and values.

Some methods of the HashMap Class



Method	Description
`clear()`	Remove all entries from the map.
`containsKey(key)`	Returns true if the specified key is in the map.
`containsValue(value)`	Returns true if the specified value is in the map.
`get(key)`	Returns the value for the entry with the specified key. Returns null if the key isn't found.
`put(key, value)`	Adds an entry with the specified key and value, or replaces the value if an entry with the key already exists.
`remove(key)`	Removes the entry with the specified key.
`size()`	Returns the number of entries in the map.

Code that uses a hash map

```
HashMap<String, String> courses = new HashMap<String, String>();

courses.put("cst231", "Intro to Programming Language in C++");
courses.put("cst238", "Intro to Data Structures");
courses.put("cst300", "Prosem");
courses.put("cst338", "Software Design");
courses.put("cst438", "Software Engineering");
```

```
System.out.println("courses");
System.out.println("\nValue of CST338 is " + courses.get("cst338"));
```

Resulting output

```
{cst438=Software Engineering, cst231=Introduction to
Programming Language in C++, cst338=Software Design,
cst300=Prosem, cst238=Intro to Data Structures}

Value of CST338 is Software Design
```

Sample Program – Toy2 Class

```
public class Toy2 {
  private String name;
  private double score;

  public Toy2 (String n, double s) {
    name = n;
    score = s;
  }

  @Override
  public String toString() {
    return "[name: " + name + ", score:" + score + "]";
  }
}
```

Sample hash map program with Toy2 class

```
import java.util.HashMap;
public class MapSample2 {
  public static void main(String[] args) {
    HashMap<Integer, Toy2> toyBox = new HashMap<Integer, Toy2>();
    toyBox.put(100, new Toy2("Tom", 90.5));
    toyBox.put(200, new Toy2("Alice", 100));
    toyBox.put(750, new Toy2("Monica", 88.25));
    // print all entries
    System.out.println(toyBox);
    // remove an entry
    toyBox.remove(100);
    // add a new entry
    toyBox.put(400, new Toy2("Oprah", 68.75));
    // replace an entry
    toyBox.put(750, new Toy2("Susan", 99.90));
    // look up a value
    System.out.println("Value of key 750: " + toyBox.get(750));
    // print all entries
    System.out.println(toyBox);
  }
}
```

Execution Result

```
{100=[name: Tom, score:90.5], 750=[name: Monica,
score:88.25], 200=[name: Alice, score:100.0]}}

Value of key 750: [name: Susan, score:99.9]

{750=[name: Susan, score:99.9], 200=[name: Alice,
score:100.0], 400=[name: Oprah, score:68.75]}
```

Lab Exercise

```
1. Input a key: 100
2. Input a name: Tom
3. Input a number: 89.5
4. More items?(Y/N) y
5. Input a key: 200
6. Input a name: John
7. Input a number: 90.5
8. More items?(Y/N) y
9. Input a key: 100
10. Input a name: Alice
11. Input a number: 91.5
12. Input error! Key 100 already exists.
13. More items?(Y/N) n
14. A key to search: 100
15. The key 100 has the value Tom 89.5 in the HashMap
```

```
16. More search?(Y/N) y
17. A key to search: 150
18. The key 150 doesn’t exist at the HashMap
19. More search?(Y/N) n
```

20. A key to remove: **200**

21. The value John 90.5 of key 150 is removed from the HashMap

Reference

How HashMap Works In Java

Please submit your code for Parts 1 and 2 on iLearn.