Topics to cover:

1. Hashtables in Java

1. Hashtables

Java Hashtable class

Java Hashtable class implements a hashtable, which maps keys to values. It inherits Dictionary class and implements the Map interface.

- A Hashtable is an array of a list. Each list is known as a bucket. The position of the bucket is identified by calling the hashcode() method. A Hashtable contains values based on the key.
- Java Hashtable class contains unique elements.
- Java Hashtable class doesn't allow null key or value.
- Java Hashtable class is synchronized.
- The initial default capacity of Hashtable class is 11 whereas loadFactor is 0.75.

Hashtable class declaration

Let's see the declaration for java.util.Hashtable class.

```
public class Hashtable<K,V> extends Dictionary<K,V> implements Map<K,V>,
```

Hashtable class Parameters

Let's see the Parameters for java.util.Hashtable class.

- **K**: It is the type of keys maintained by this map.
- **V**: It is the type of mapped values.

Java Hashtable Example

```
import java.util.*;
class Hashtable1{
 public static void main(String args[]){
  Hashtable<Integer,String> ht=new Hashtable<Integer,String>();

  ht.put(100,"Amit");
  ht.put(102,"Ravi");
  ht.put(101,"Vijay");
  ht.put(103,"Rahul");

  for(Map.Entry m:hm.entrySet()){
   System.out.println(m.getKey()+" "+m.getValue());
  }
 }
}
```

Output:

```
103 Rahul
102 Ravi
101 Vijay
100 Amit
```

Java Hashtable Example: remove()

```
import java.util.*;

public class Hashtable2 {

  public static void main(String args[]) {

 Hashtable<Integer,String> map=new Hashtable<Integer,String>();

   map.put(100,"Amit");

   map.put(102,"Ravi");

   map.put(101,"Vijay");

   map.put(103,"Rahul");

   System.out.println("Before remove: "+ map);

    // Remove value for key 102

    map.remove(102);

    System.out.println("After remove: "+ map);

  }
}
```

Output:

```
Before remove: {103=Rahul, 102=Ravi, 101=Vijay, 100=Amit}
After remove: {103=Rahul, 101=Vijay, 100=Amit}
```

Java Hashtable Example: getOrDefault()

```
import java.util.*;
class Hashtable3{
 public static void main(String args[]){
   Hashtable<Integer,String> map=new Hashtable<Integer,String>();
    map.put(100,"Amit");
    map.put(102,"Ravi");
    map.put(101,"Vijay");
    map.put(103,"Rahul");
    //Here, we specify the if and else statement as arguments of the method
    System.out.println(map.getOrDefault(101, "Not Found"));
    System.out.println(map.getOrDefault(105, "Not Found"));
 }
}
```

Output:

```
Vijay
```

Java Hashtable Example: putIfAbsent()

```
import java.util.*;

class Hashtable4{

 public static void main(String args[]){

    Hashtable<Integer,String> map=new Hashtable<Integer,String>();

     map.put(100,"Amit");

     map.put(102,"Ravi");

     map.put(101,"Vijay");

     map.put(103,"Rahul");

     System.out.println("Initial Map: "+map);

     //Inserts, as the specified pair is unique

     map.putIfAbsent(104,"Gaurav");

     System.out.println("Updated Map: "+map);

     //Returns the current value, as the specified pair already exist

     map.putIfAbsent(101,"Vijay");

     System.out.println("Updated Map: "+map);

 }
```

```
    }
```

Output:

```
Initial Map: {103=Rahul, 102=Ravi, 101=Vijay, 100=Amit}
Updated Map: {104=Gaurav, 103=Rahul, 102=Ravi, 101=Vijay, 100=Amit}
Updated Map: {104=Gaurav, 103=Rahul, 102=Ravi, 101=Vijay, 100=Amit}
```

Java Hashtable Example: Book

```
    import java.util.*;

    class Book {

    int id;

    String name,author,publisher;

    int quantity;

    public Book(int id, String name, String author, String publisher, int quantity) {

       this.id = id;

       this.name = name;

       this.author = author;

       this.publisher = publisher;

       this.quantity = quantity;

    }
```

```
}

public class HashtableExample {

public static void main(String[] args) {

   //Creating map of Books

   Map<Integer,Book> map=new Hashtable<Integer,Book>();

   //Creating Books

   Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);

   Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc
Graw Hill",4);

   Book b3=new Book(103,"Operating System","Galvin","Wiley",6);

   //Adding Books to map

   map.put(1,b1);

   map.put(2,b2);

   map.put(3,b3);

   //Traversing map

   for(Map.Entry<Integer, Book> entry:map.entrySet()){

      int key=entry.getKey();

      Book b=entry.getValue();

      System.out.println(key+" Details:");
```

```
        System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+"
"+b.quantity);
    }
}
}
```

Output:

```
3 Details:
103 Operating System Galvin Wiley 6
2 Details:
102 Data Communications & Networking Forouzan Mc Graw Hill 4
1 Details:
101 Let us C Yashwant Kanetkar BPB 8
```

Difference between HashMap and Hashtable

HashMap and Hashtable both are used to store data in key and value form. Both are using hashing technique to store unique keys.

But there are many differences between HashMap and Hashtable classes that are given below.

HashMap	**Hashtable**
1) HashMap is **non synchronized**. It is not-thread safe and can't be shared between	Hashtable is **synchronized**. It is thread-safe and can be shared with many threads.

many threads without proper synchronization code.	
2) HashMap **allows one null key and multiple null values**.	Hashtable **doesn't allow any null key or value**.
3) HashMap is a **new class introduced in JDK 1.2**.	Hashtable is a **legacy class**.
4) HashMap is **fast**.	Hashtable is **slow**.