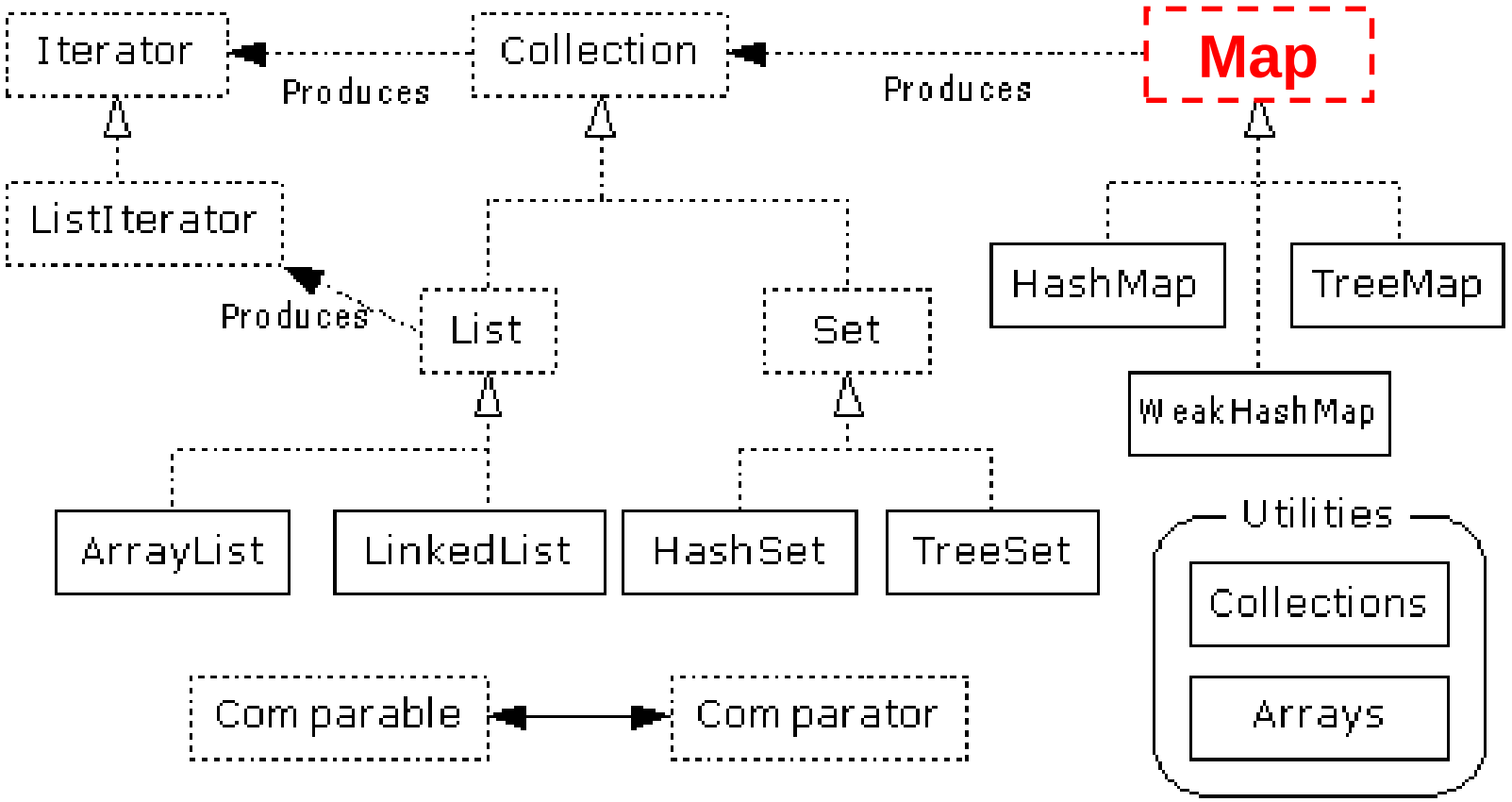
Java & JEE Training

Day 20 – Collections - Maps

MindsMapped Consulting

Map Interface Context



Map Interface

- Stores key/value pairs
- Maps from the key to the value
- Keys are unique
  - a single key only appears once in the Map
  - a key can map to only one value
- Values do not have to be unique

Map methods

Method	Description
Object put(Object key, Object value)	It is used to insert an entry in this map.
void putAll(Map map)	It is used to insert the specified map in this map.
Object remove(Object key)	It is used to delete an entry for the specified key.
Object get(Object key)	It is used to return the value for the specified key.
pboolean containsKey(Object key)	It is used to search the specified key from this map.
Set keySet()	It is used to return the Set view containing all the keys.
Set entrySet()	It is used to return the Set view containing all the keys and values.

Map views

- A means of iterating over the keys and values in a Map
- **`Set keySet()`**
  - returns the Set of keys contained in the Map
- **`Collection values()`**
  - returns the Collection of values contained in the Map. This Collection is not a Set, as multiple keys can map to the same value.
- **`Set entrySet()`**
  - returns the Set of key-value pairs contained in the Map. The Map interface provides a small nested interface called Map.Entry that is the type of the elements in this Set.

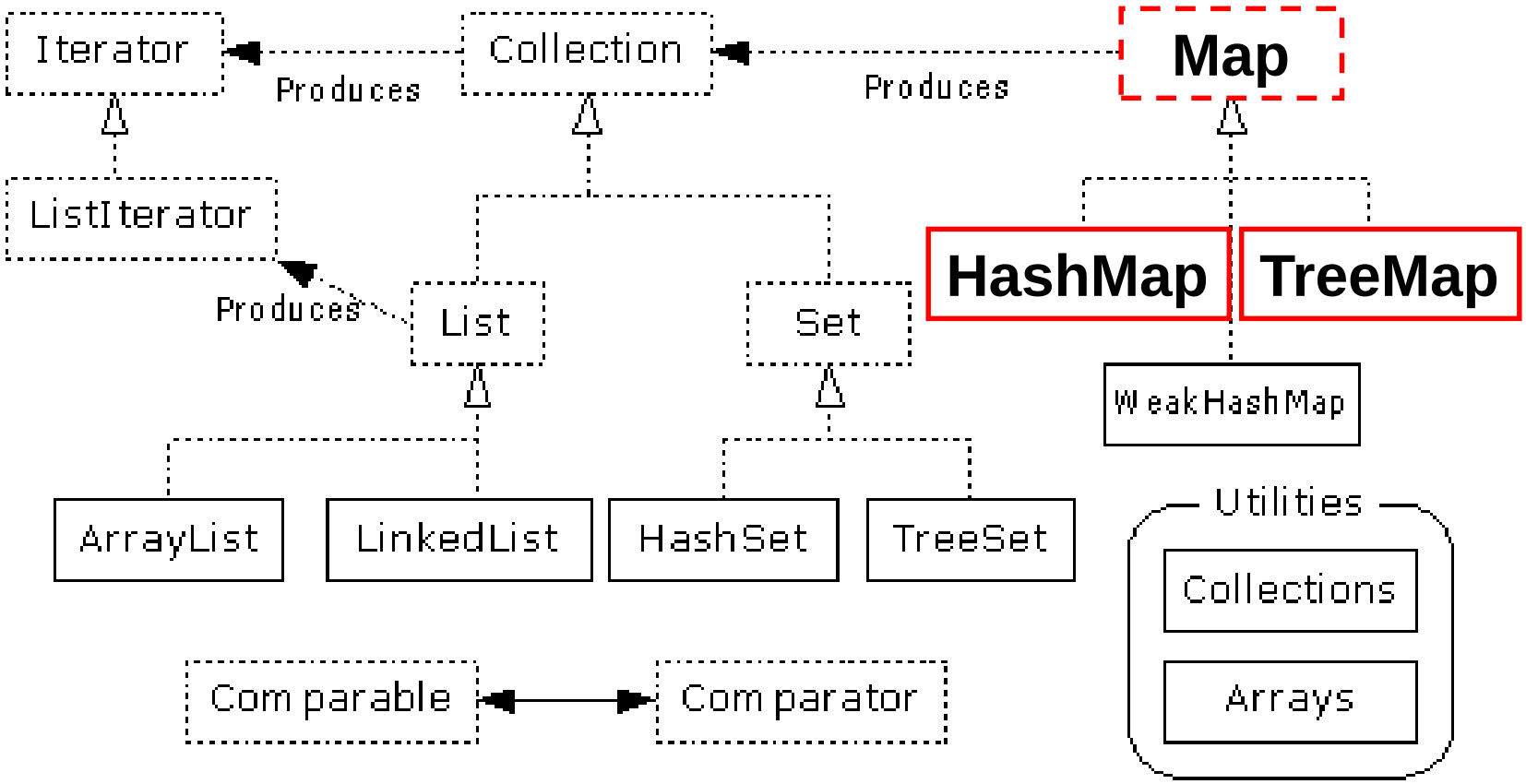
Map.Entry interface Example

```
import java.util.*;
class MapInterfaceExample{
 public static void main(String args[]){
  Map<Integer,String> map=new HashMap<Integer,String>();
  map.put(100,"Amit");
  map.put(101,"Vijay");
  map.put(102,"Rahul");
  for(Map.Entry m:map.entrySet()){
   System.out.println(m.getKey()+" "+m.getValue());
  }
 }
}
```

Exercise..

- Create a Map with 5 values…
- Iterate through and print
  - the keys,
  - the values,
  - both the keys and the values of the Map

HashMap and TreeMap Context



HashMap and TreeMap

- HashMap
  - The keys are a set - unique, unordered
  - Fast

- TreeMap
  - The keys are a set - unique, ordered
  - Same options for ordering as a TreeSet
    - *Natural order (Comparable, compareTo(Object))*
    - *Special order (Comparator, compare(Object, Object))*

HashMap

- A HashMap contains values based on the key.
- It contains only unique elements.
- It may have one null key and multiple null values.
- It maintains no order.

HashMap Constructors

Constructor	Description
HashMap()	It is used to construct a default HashMap.
HashMap(Map m)	It is used to initializes the hash map by using the elements of the given Map object m.
HashMap(int capacity)	It is used to initializes the capacity of the hash map to the given integer value, capacity.
HashMap(int capacity, float fillRatio)	It is used to initialize both the capacity and fill ratio of the hash map by using its arguments.

HashMap methods

Method	Description
void clear()	It is used to remove all of the mappings from this map.
boolean containsKey(Object key)	It is used to return true if this map contains a mapping for the specified key.
boolean containsValue(Object value)	It is used to return true if this map maps one or more keys to the specified value.
boolean isEmpty()	It is used to return true if this map contains no key-value mappings.
Object clone()	It is used to return a shallow copy of this HashMap instance: the keys and values themselves are not cloned.
Set entrySet()	It is used to return a collection view of the mappings contained in this map.
Set keySet()	It is used to return a set view of the keys contained in this map.
Object put(Object key, Object value)	It is used to associate the specified value with the specified key in this map.
int size()	It is used to return the number of key-value mappings in this map.
Collection values()	It is used to return a collection view of the values contained in this map.

HashMap Example: remove()

```
import java.util.*;
public class HashMapExample {
   public static void main(String args[]) {
   // create and populate hash map
   HashMap<Integer, String> map = new HashMap<Integer, String>();
   map.put(101,"Let us C");
   map.put(102, "Operating System");
   map.put(103, "Data Communication and Networking");
   System.out.println("Values before remove: "+ map);
   // Remove value for key 102
   map.remove(102);
   System.out.println("Values after remove: "+ map);
   }
}
```

Difference between HashSet and HashMap

- Think…

HashMap 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 MapExample {
public static void main(String[] args) {
   //Creating map of Books
   Map<Integer,Book> map=new HashMap<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);
   }
}
}
```

Exercise..

- What is a TreeMap? Implement the book example for TreeMap.
- What is a LinkedHashMap?
- What is a HashTable? How is it different from a HashMap?
- What is Comparable and Comparator interfaces in Java? How are they different? Write programs utilizing both to test their usage.

Bulk Operations

- In addition to the basic operations, a Collection may provide "bulk" operations

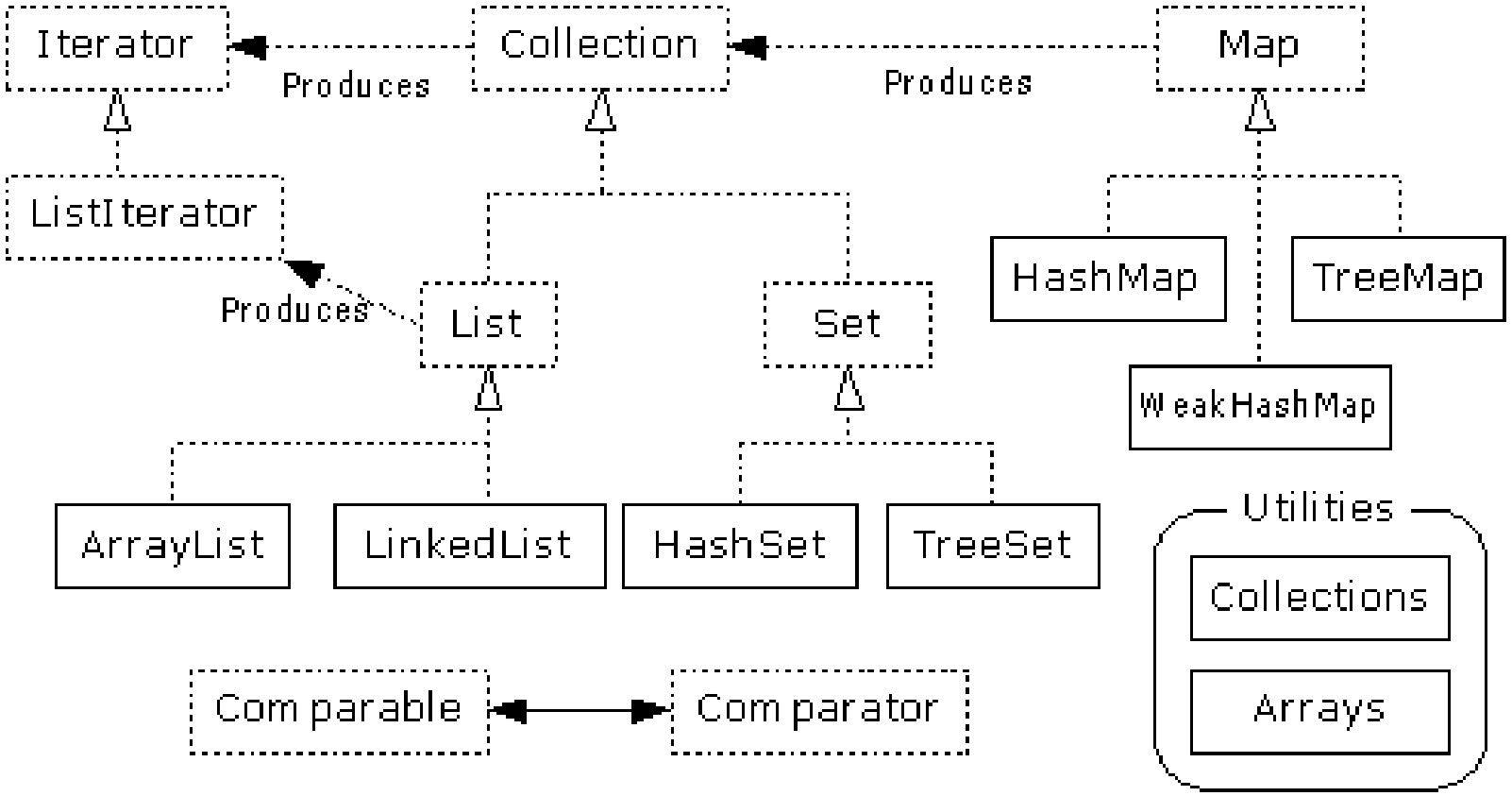
```
boolean containsAll(Collection c);
boolean addAll(Collection c);    // Optional
boolean removeAll(Collection c); // Optional
boolean retainAll(Collection c); // Optional
void clear();                    // Optional
Object[] toArray();
Object[] toArray(Object a[]);

A = {1,2,3,4} B={2,3,4,5}
addAll() -> A U B = {1,2,3,4,5} UNION
A = STUDENTS LEARNING JAVA; B = STUDENTS LEARNING .NET
A U B  = ALL STUDENTS LEARNING EITHER JAVA OR .NET OR BOTH
A INTERSECTION B =  ALL STUDENTS LEARNING BOTH JAVA AND .NET
removeAll() -> A-B = {1}, B-A={5}..
A IS ALL EMPLOYEES, B IS EMPLOYEES SERVING NOTICE PERIOD
A-B = EMPLOYEES NOT SERVING NOTICE PERIOD
retainAll() -> A INTERSECTION B = {2,3,4}
```

HashMap vs TreeMap

- HashMap can contain one null key. TreeMap cannot contain a null key.
- HashMap does not maintain any order, whereas TreeMap maintains ascending order.

Utilities Context



Utilities

- The Collections class provides a number of static methods for fundamental algorithms
- Most operate on Lists, some on all Collections
  - Sort, Search, Shuffle
  - Reverse, fill, copy
  - Min, max
- Wrappers
  - synchronized Collections, Lists, Sets, etc
  - unmodifiable Collections, Lists, Sets, etc

Properties class

- Located in java.util package
- Special case of Hashtable
  - Keys and values are Strings
  - Tables can be saved to/loaded from file