Class Libraries

CSE 142, Summer 2002

Computer Programming 1

http://www.cs.washington.edu/education/courses/142/02su/

Readings and References

- Reading

- Other References
  - » The Java tutorial
  - » http://java.sun.com/docs/books/tutorial/

Java fundamentals

- Object oriented programming
  - » classes and objects
  - » interfaces and inheritance
  - » constructors, methods, variables
- The Java language
  - » types, expressions
  - » control flow
  - » exceptions
- Development tools
  - » editors, compiler, Java virtual machine

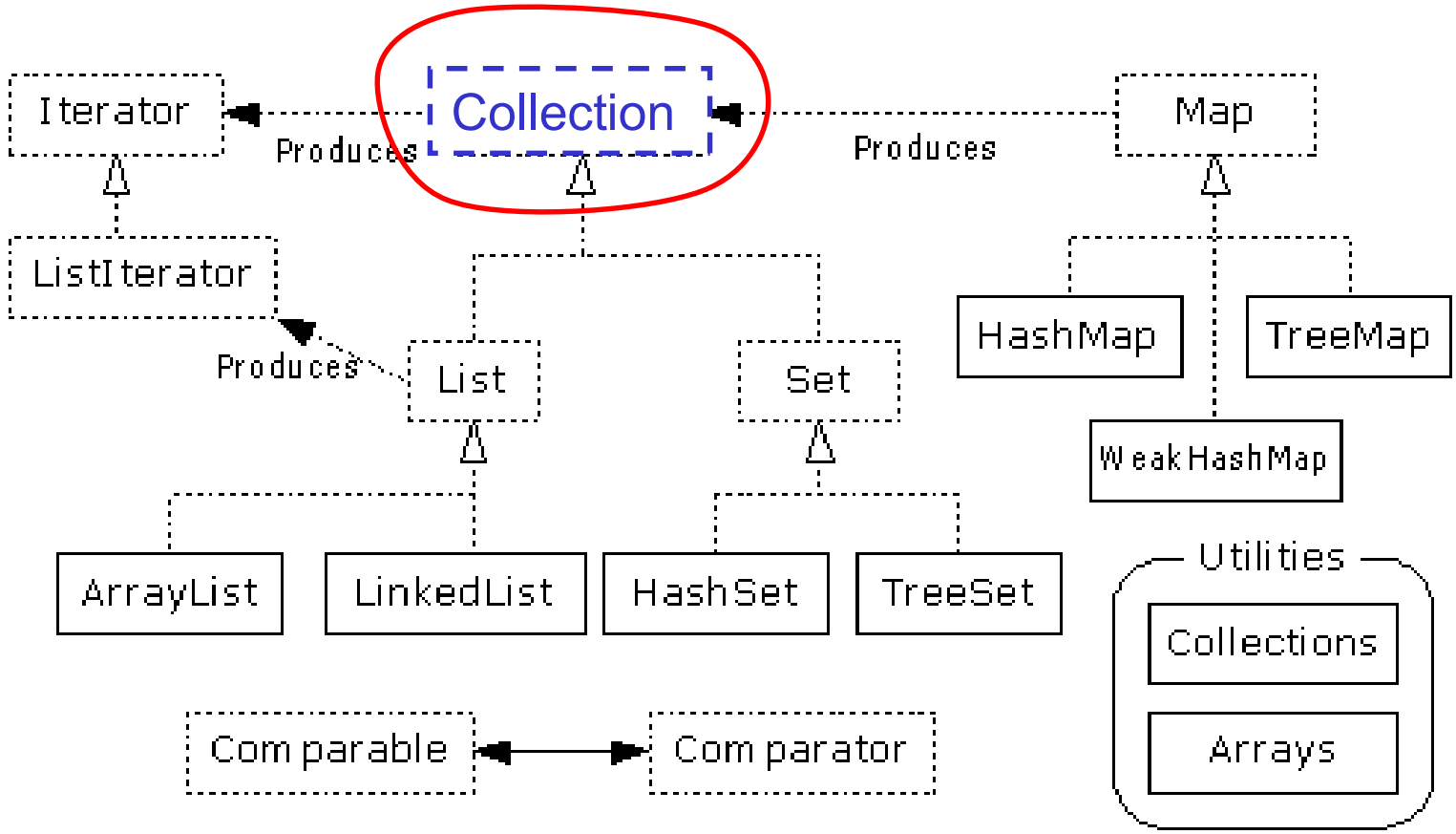
Java class libraries and data structures

- ArrayLists
  - » representative of the many Collection types
- Arrays
  - » can hold primitive types directly
- Input and Output
  - » we just looked at the tip of the iceberg with this package

There's more, much more ...

- Essential Classes
  - » Collections and Internationalization
- Advanced GUI Building
  - » Swing, 2D Graphics, Sound, and JavaBeans
- Networking and Connectivity
  - » JDBC, RMI, IDL, Servlets, and Security
- Packaging
  - » JAR and the Extension Mechanism
- Advanced Language Topics
  - » Java Native Interface and Reflection

Collection interface



- Interfaces, Implementations, and Algorithms
- From Thinking in Java, page 462

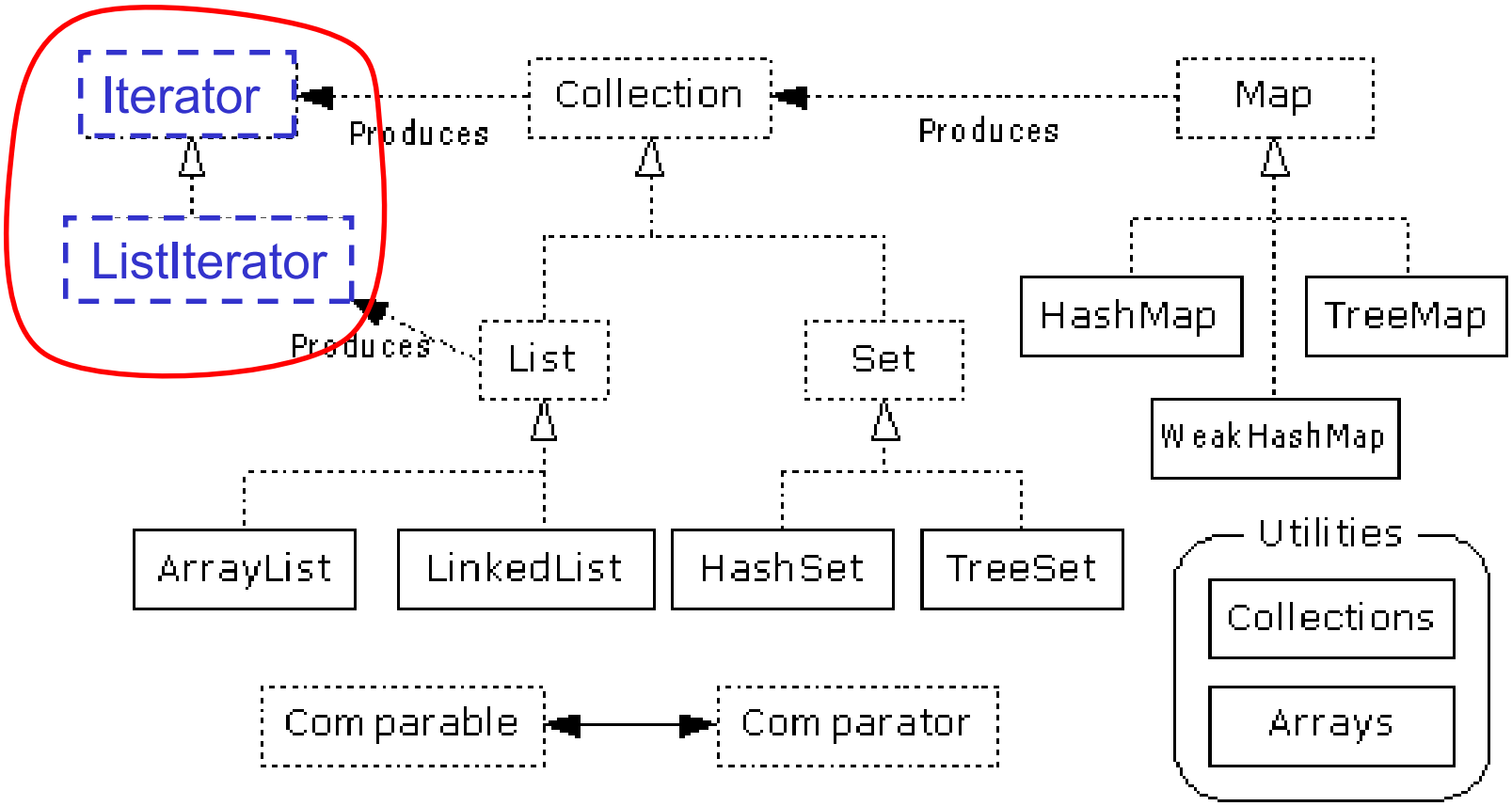
java.util.Collection Interface

- Collection is the root interface in the collection hierarchy
  - » A collection represents a group of objects (the elements of the collection)
  - » Some collections allow duplicate elements and others do not
  - » Some collections are ordered and others are unordered

Collection interface methods

- Defines two fundamental methods
  - » boolean add(Object o)
  - » Iterator iterator()
- These two methods are enough to define the basic behavior of a collection
- An Iterator lets you step through the elements in the Collection without knowing the index

Iterator interface



Iterator Interface

- Defines three fundamental methods
  - » Object next()
  - » boolean hasNext()
  - » void remove() (optional operation)
- These three methods provide access to the contents of the collection
- An Iterator knows position within collection
- Each call to **`next()`** gets the next element from the collection

Iterator Position with next()

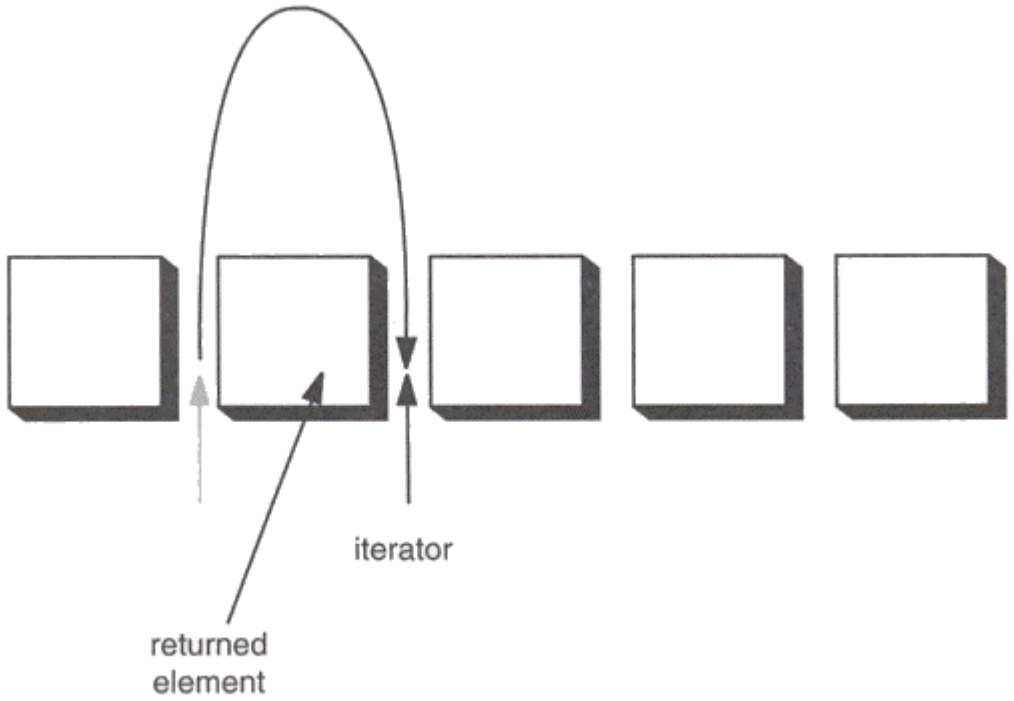


Figure 2–3: Advancing an iterator

Example - SimpleCollection

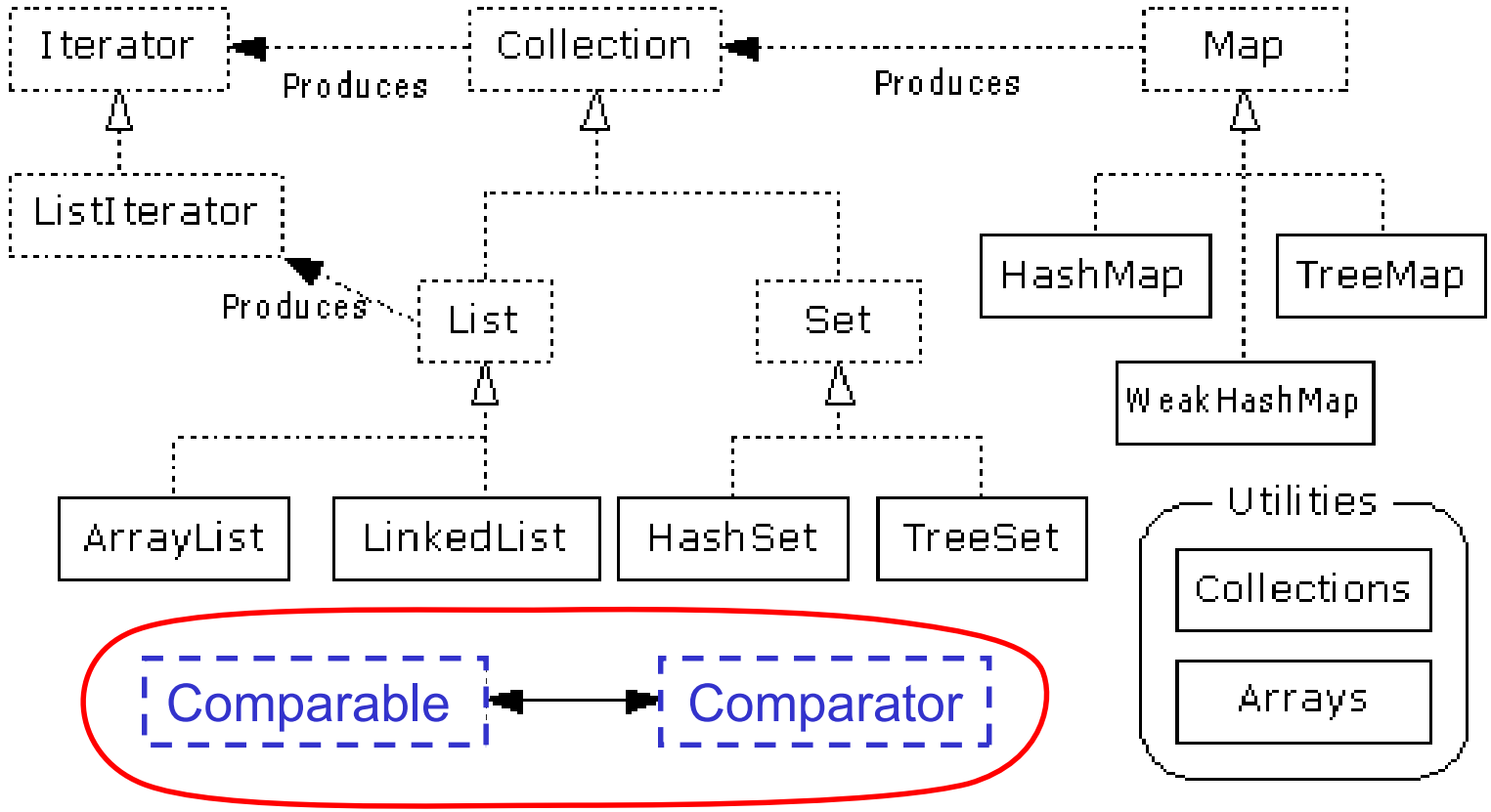
```
public void processCollection(Collection c) {

   System.out.println(c.getClass().getName());

   for (int i=9; i >= 0; i--) {
      c.add(i + " * " + i + " = "+i*i);
   }

   Iterator iter = c.iterator();
   while (iter.hasNext()) {
      System.out.println(iter.next());
   }
}
```

Comparable and Comparator interfaces



Comparable and Comparator interfaces

- java.lang.Comparable
  - » Imposes a total ordering on the objects of each class that implements it
  - » This ordering is referred to as the class's natural ordering, and the class's compareTo() method is referred to as its natural comparison method
- java.util.Comparator
  - » Defines a comparison function, which imposes a total ordering on some collection of objects
  - » Comparators can be passed to a sort method (such as Collections.sort) to allow precise control over the sort order
  - » Comparators can also be used to control the order of certain data structures (such as TreeSet or TreeMap)

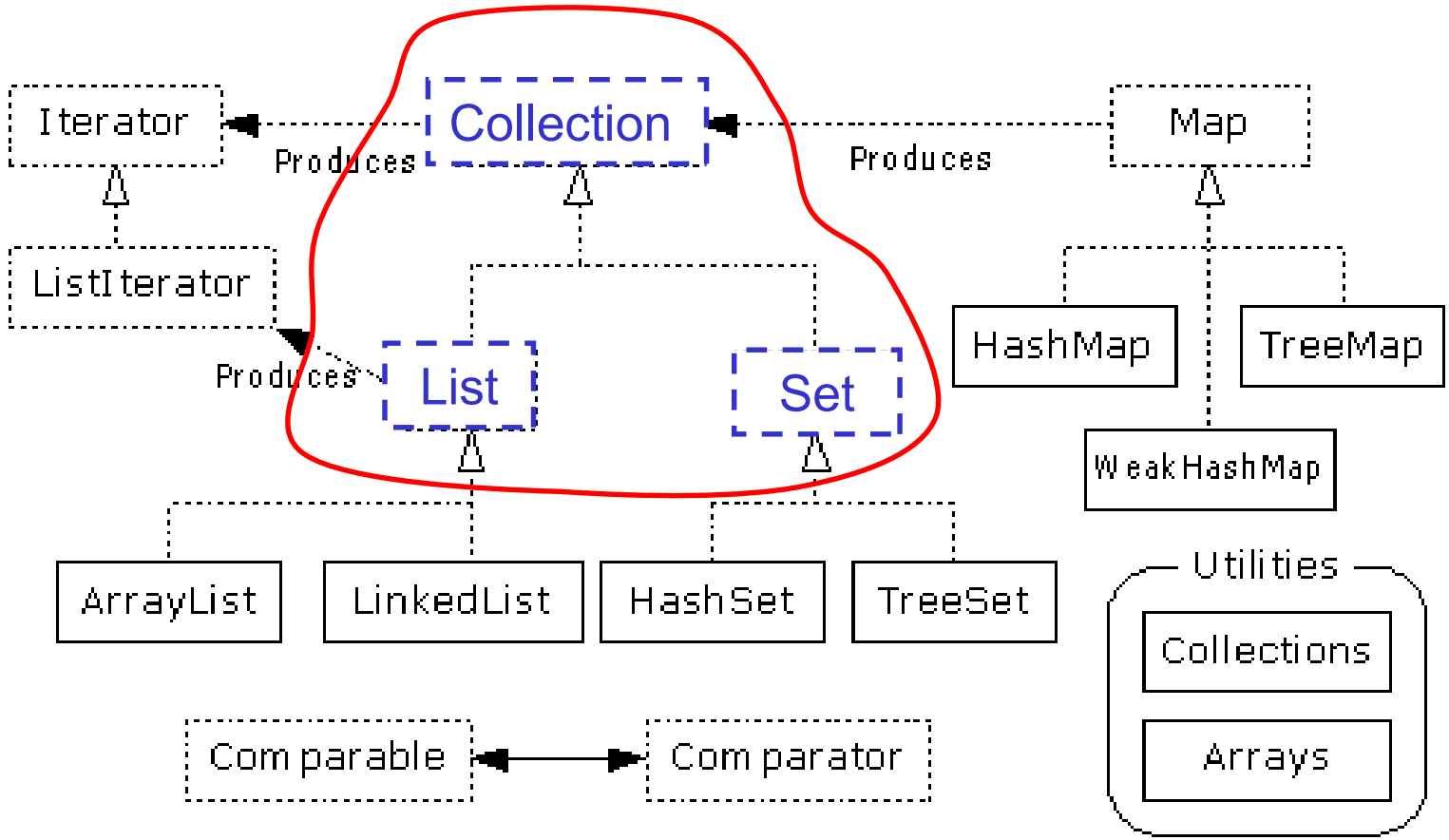
Comparable Example

```
public class Vehicle implements Comparable, Cloneable {
    ...
    /**
    * Compares this object with the specified object for order, based on the
    * vehicle identification number.
    * @param o the other Vehicle object
    * @return a negative integer, zero, or a positive integer as this
    * object is less than, equal to, or greater than the specified object.
    */
    public int compareTo(Object o) {
        Vehicle other = (Vehicle)o;
        if (this.vin < other.vin) {
            return -1;
        } else if (this.vin > other.vin) {
            return +1;
        } else {
            return 0;
        }
    }
```

Comparator example

```
import java.util.*;
import hw6.*;
public class CompareByRoute implements Comparator {
    /**
    * This method compares two TransitBus objects based on their route
    * number.  This is the only method required by the Comparator interface.
    * @param oA the first object to be compared.  Must be a TransitBus.
    * @param oB the second object to be compared.  Must be a TransitBus.
    * @return a negative integer, zero, or a positive integer as the route
    * number of the first argument is less than, equal to, or greater than
    * the route number of the second argument.
    */
    public int compare(Object oA, Object oB) {
        int rA = ((TransitBus)oA).getRoute();
        int rB = ((TransitBus)oB).getRoute();
        if (rA < rB) {
            return -1;
        } else if (rA > rB) {
            return +1;
        } else {
            return 0;
        }
    }
}
```

List and Set interfaces

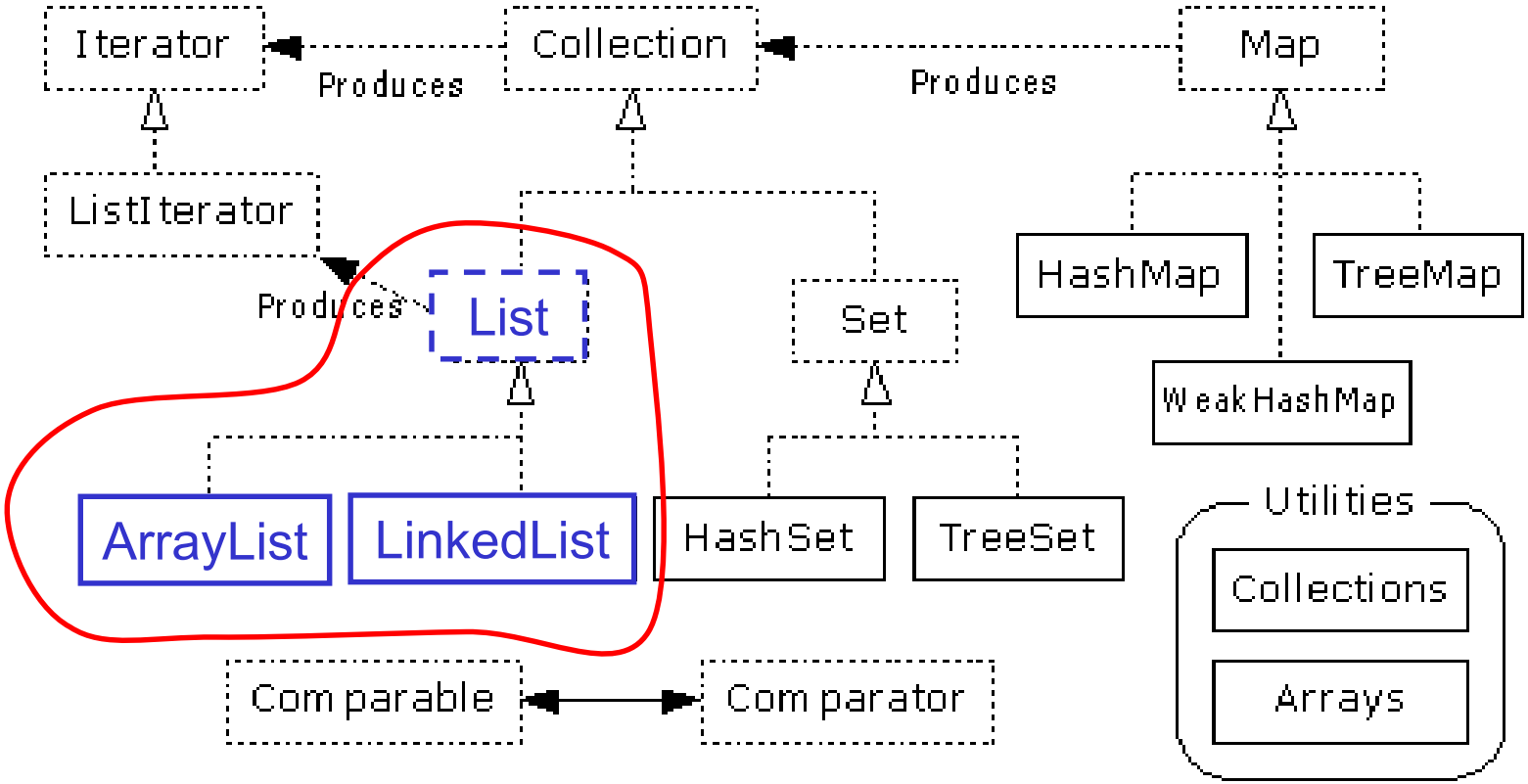


List and Set

- **`public interface List extends Collection`**
  - » An ordered collection (also known as a *sequence*)
  - » User can store and access elements by their integer index and search for elements in the list
  - » Lists typically allow duplicate elements

- **`public interface Set extends Collection`**
  - » A collection that contains no duplicate elements and at most one null element
  - » Models the mathematical *set* abstraction

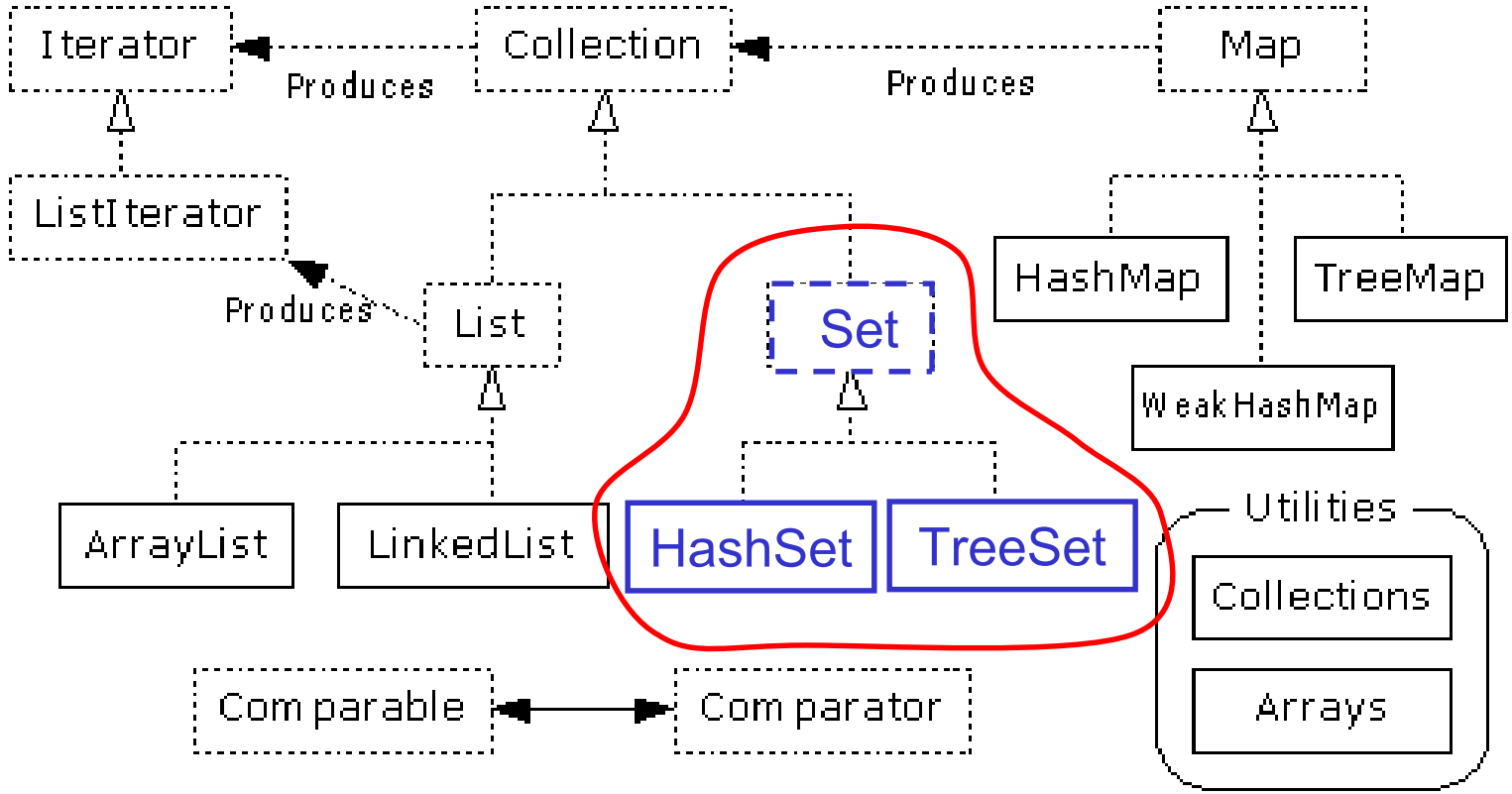
Concrete classes that implement List



ArrayList and LinkedList

- ArrayList
  - » fast access to any element in the List by index
  - » implemented with an array of Objects, ie, Object[]
  - » automatically increases array size when needed
  - » add at the end is fast, but add in the front requires copying the entire array to make room
- LinkedList
  - » fast insert and delete at any point in a list
  - » slow if you want to access elements by index

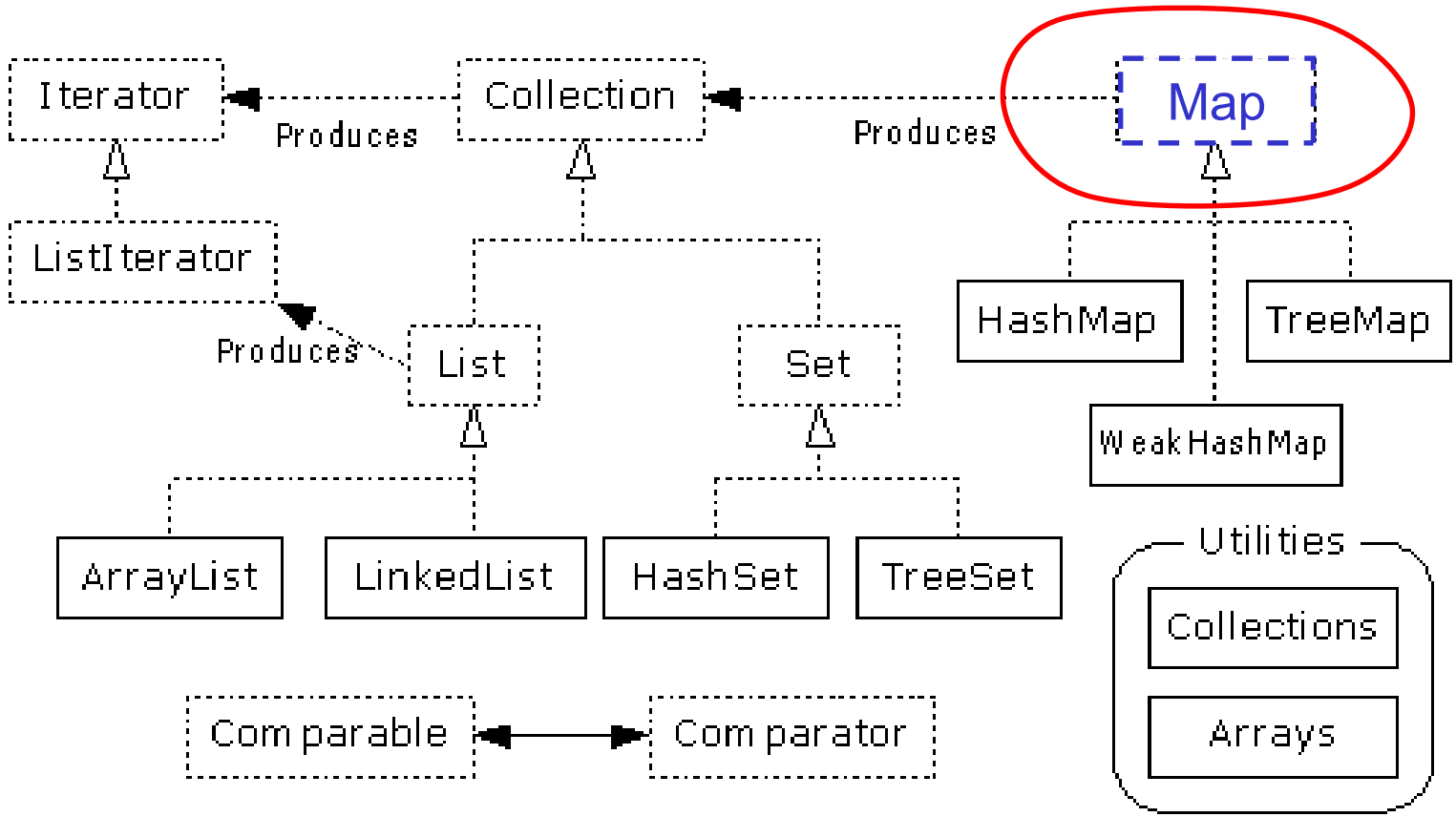
Concrete classes that implement Set



HashSet and TreeSet

- HashSet
  - » Like all Collections, a HashSet stores objects
  - » This class offers constant time performance for the basic operations (add, remove, contains and size)
  - » No guarantee as to the order of the elements
- TreeSet
  - » Guarantees that the set will be sorted in ascending element order

Map interface

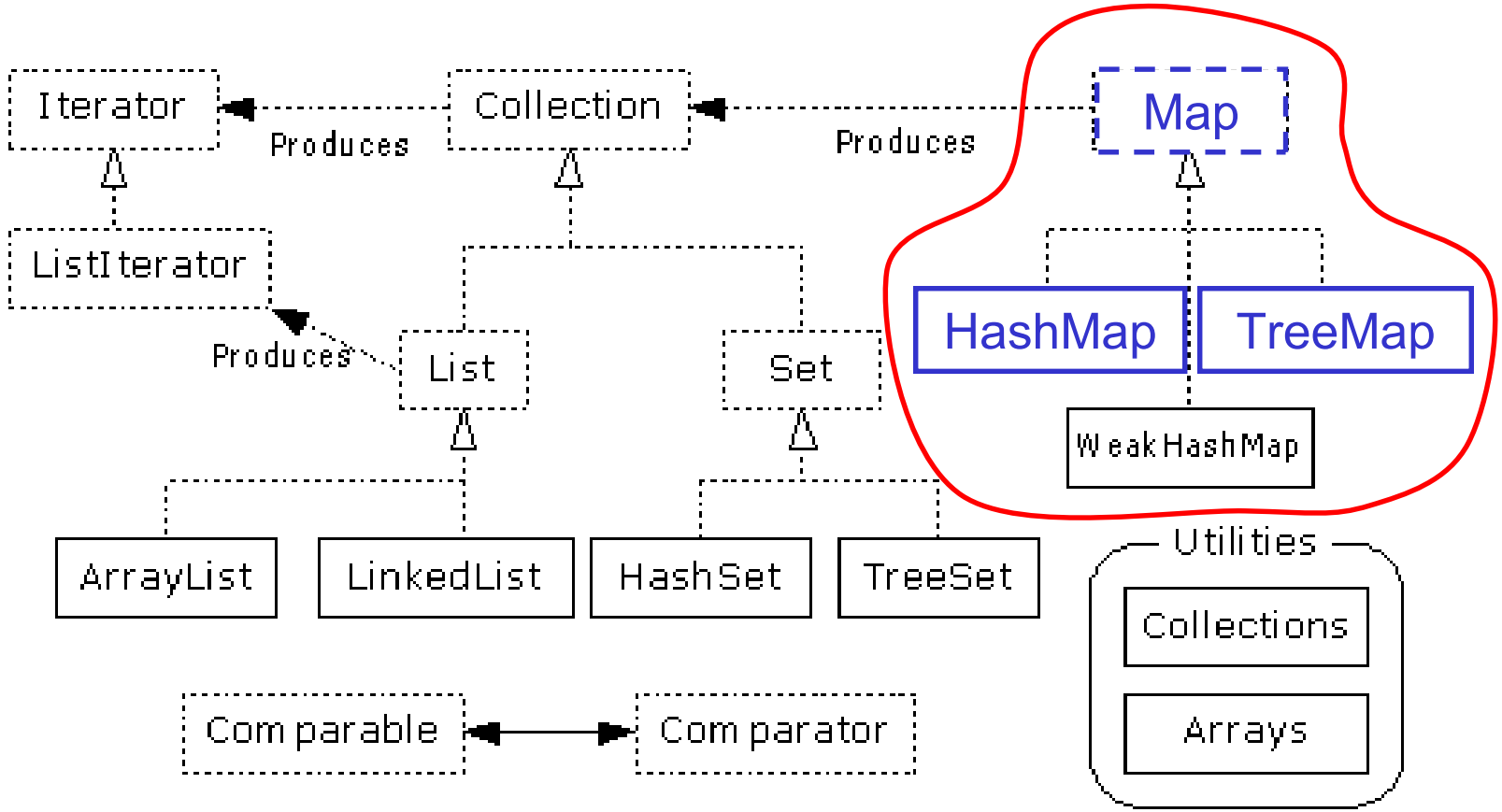


Map interface

- **`public interface Map`**
- A class that implements the Map interface provides an object that maps keys to values
  - » a map cannot contain duplicate keys
  - » each key can map to at most one value
- This is how the property files for the sliding ovals were implemented

```
arg = (String)pMap.get("width");
if (arg != null) {
    width = Integer.parseInt(arg);
}
```

Concrete classes that implement Map

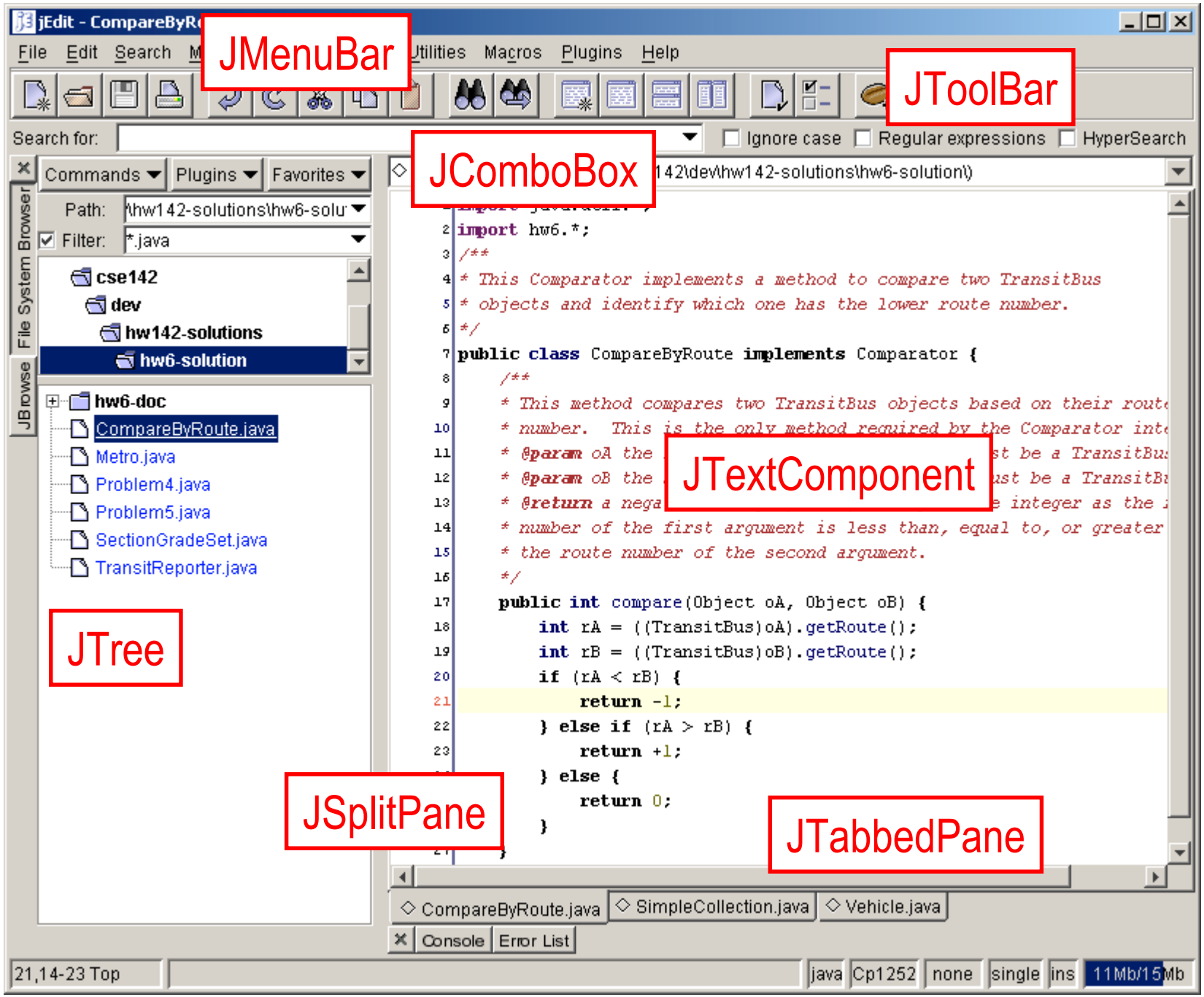


HashMap and TreeMap

- HashMap
  - » provides constant-time performance for the basic operations (get and put)
- TreeMap
  - » guarantees that the map will be in ascending key order
  - » provides guaranteed log(n) time cost for the containsKey, get, put and remove operations

Swing package

- Provides all the components needed for building a graphical user interface
  - » package javax.swing and related packages
- Can build nice looking applications to run on a variety of underlying systems
  - » for example jEdit, BusView


JMenuBar
JToolBar
JComboBox
JTextComponent
JTree
JSplitPane
JTabbedPane
File Edit Search Utilities Macros Plugins Help
Search for:
Ignore case
Regular expressions
HyperSearch
Commands
Plugins
Favorites
Path:
Filter:
*.java
cse142
dev
hw142-solutions
hw6-solution
hw6-doc
CompareByRoute.java
Metro.java
Problem4.java
Problem5.java
SectionGradeSet.java
TransitReporter.java
import hw6.*;
/**
* This Comparator implements a method to compare two TransitBus
* objects and identify which one has the lower route number.
*/
public class CompareByRoute implements Comparator {
/**
* This method compares two TransitBus objects based on their route
* number. This is the only method required by the Comparator inte
* @param oA the
* @param oB the
* @return a nega
* number of the first argument is less than, equal to, or greater
* the route number of the second argument.
*/
public int compare(Object oA, Object oB) {
int rA = ((TransitBus)oA).getRoute();
int rB = ((TransitBus)oB).getRoute();
if (rA < rB) {
return -1;
} else if (rA > rB) {
return +1;
} else {
return 0;
}
CompareByRoute.java
SimpleCollection.java
Vehicle.java
Console
Error List
21,14-23 Top
java
Cp1252
none
single
ins
11Mb/15Mb

useful packages

- java.net
  - » Classes for implementing networking applications
- java.io
  - » System input and output through data streams, serialization and the file system
- java.awt
  - » Basic classes for creating user interfaces and for painting graphics and images
  - » User interface event classes (mouse, keyboard,…)

useful packages

- javax.accessibility
  - » Defines a contract between user-interface components and an assistive technology that provides access to those components
- internationalization (java.util, java.text, …)
  - » Messages, labels on GUI components, online help, sounds, colors, graphics, icons, dates, times, numbers, currencies, measurements, phone numbers, honorifics and personal titles, postal addresses, page layouts

useful packages

- java.sql
  - » Classes for accessing and processing data in databases
- java.util.zip
  - » classes for reading and writing the standard ZIP and GZIP file formats
- java.util.jar
  - » classes for reading and writing the JAR (Java ARchive) file format, which is based on the standard ZIP file format with an optional manifest file