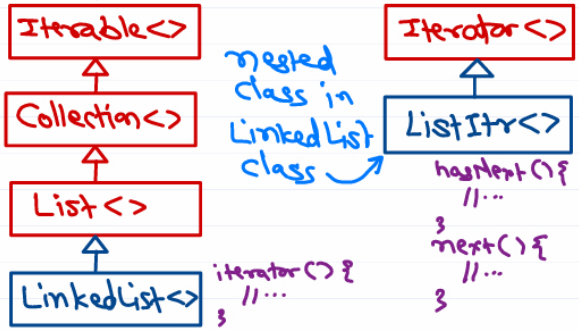
To access elems one by one from any collection we use Iterator.

```
interface Iterator<T>{
    boolean hasNext();
    T next();
}
```

And to get Iterator obj of a collection we use Iterable.

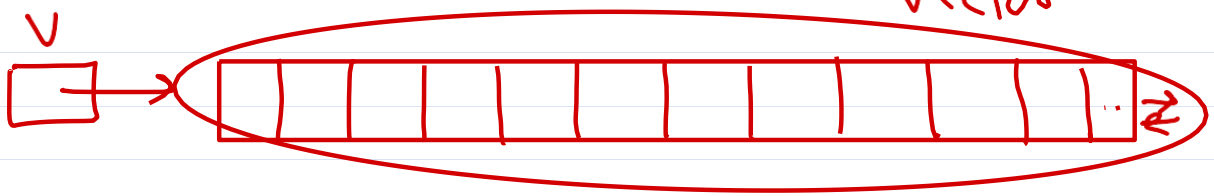
```
interface Iterable<T>{
    Iterator<T> iterator();
}
```



Vector vs ArrayList vs LinkedList

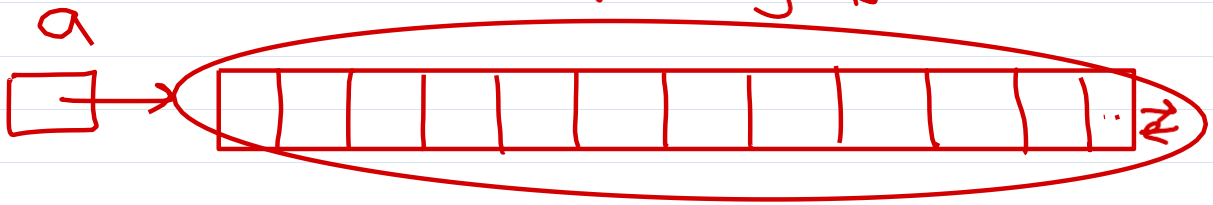
Vector v = new Vector();

* Dynamically growable/shrinkable array.
* Synchronized class / slower
* Legacy (1.0) * growth = 2 * capacity

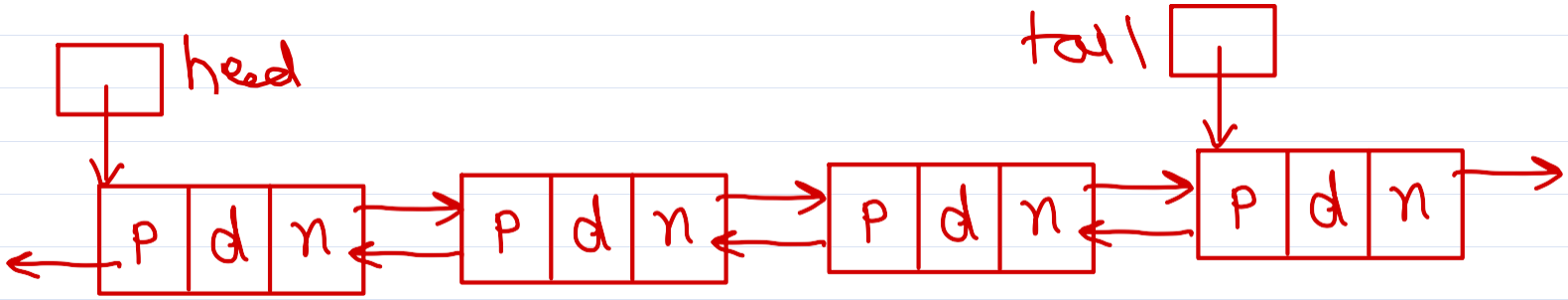


ArrayList a = new ArrayList();

* Dynamically growable/shrinkable array.
* Non-synchronized / Faster
* Collection framework (1.2)
* growth = 1.5 * capacity



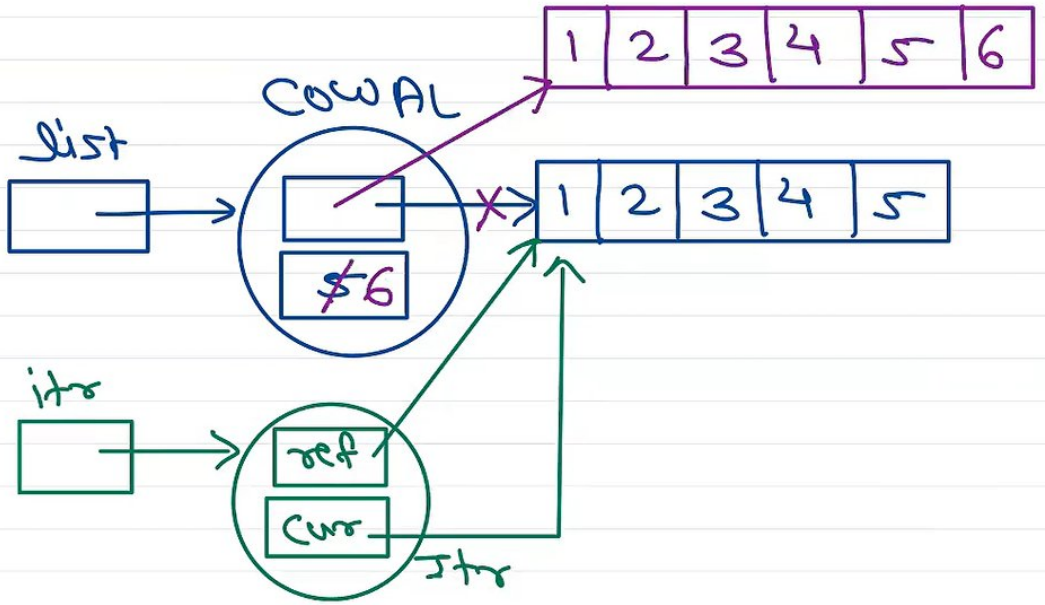
LinkedList l = new LinkedList();

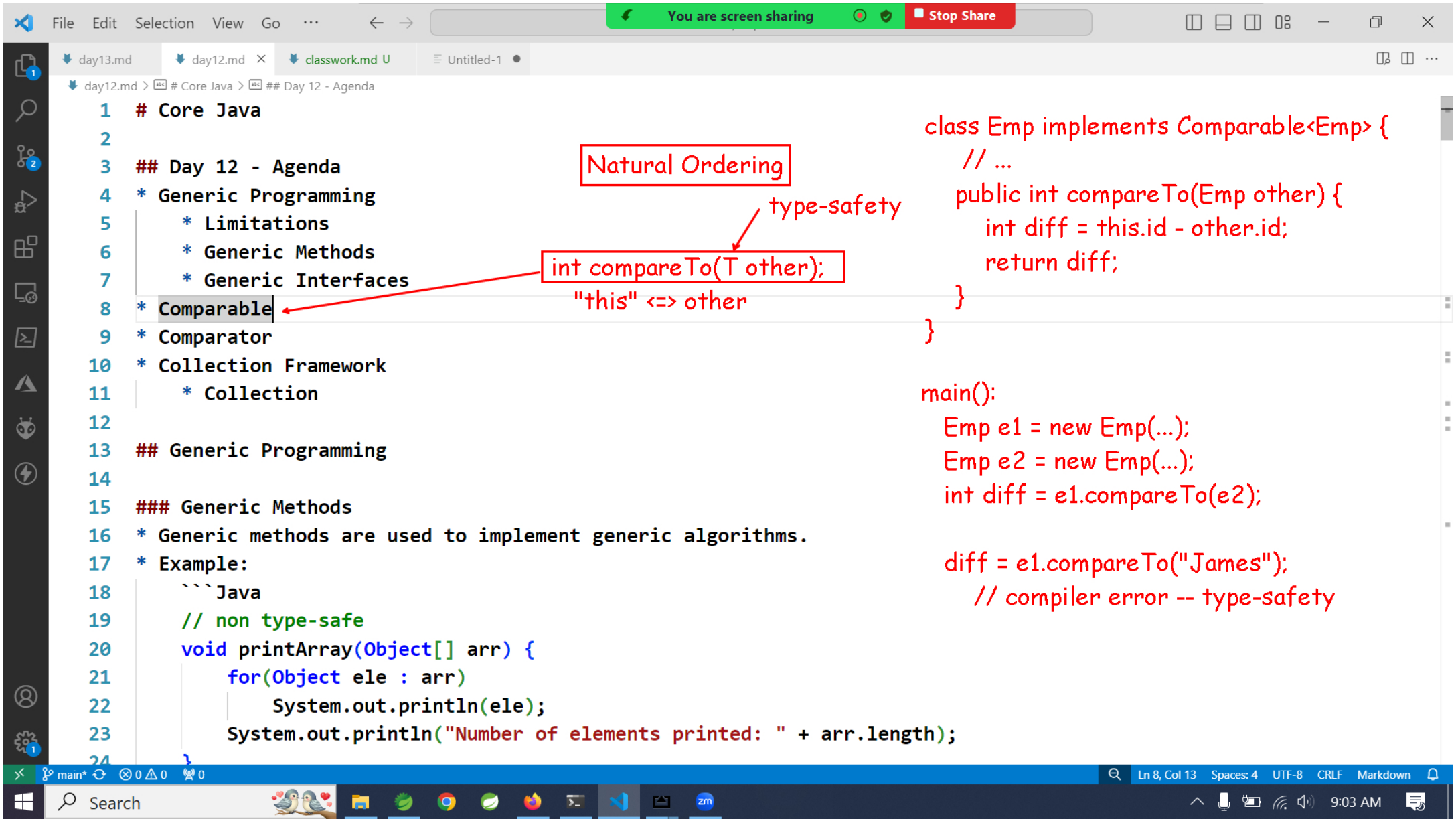


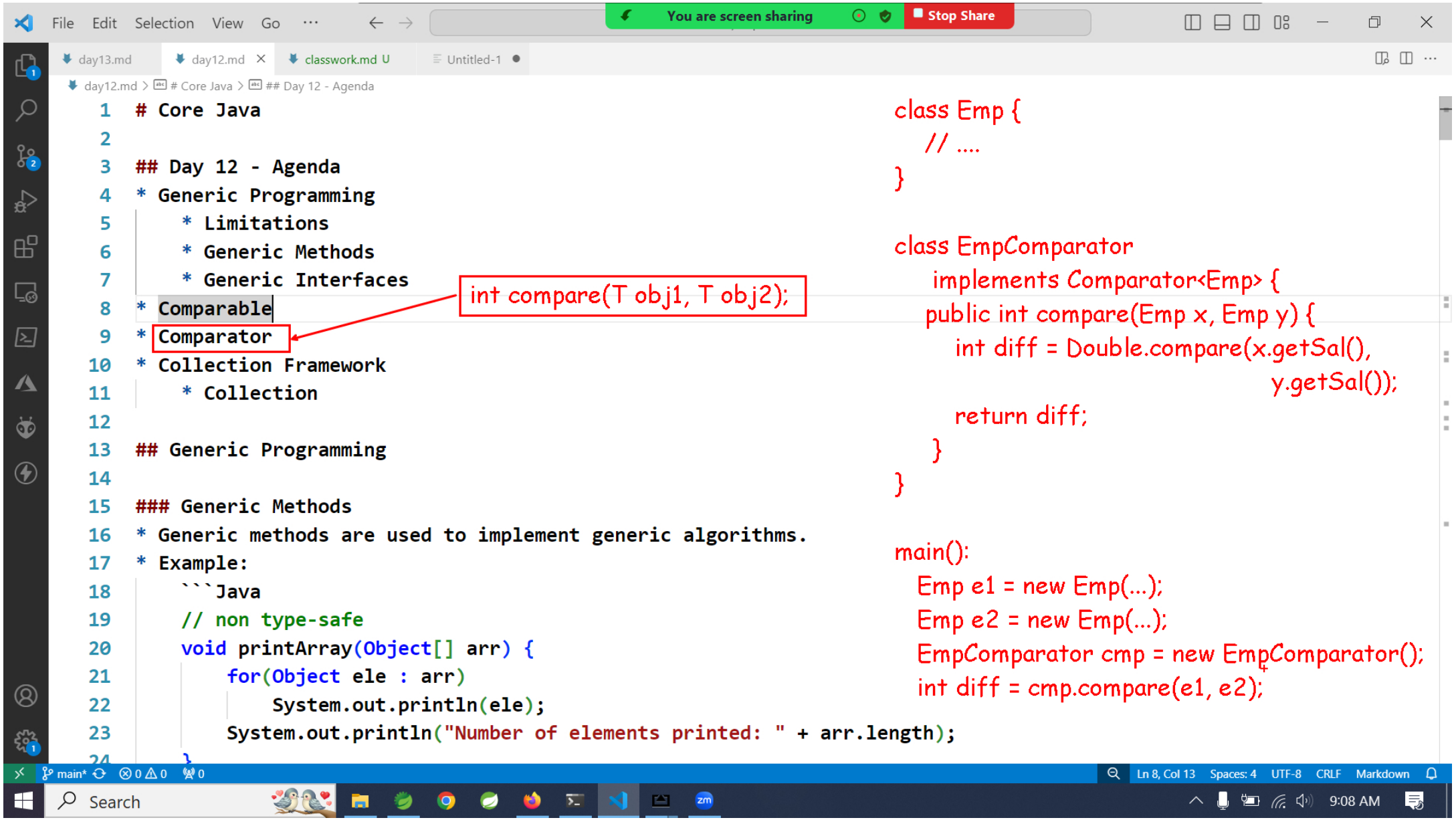
* Doubly Linked List
* frequent add/delete op.
* slower random access.
* Inherited from List, Queue

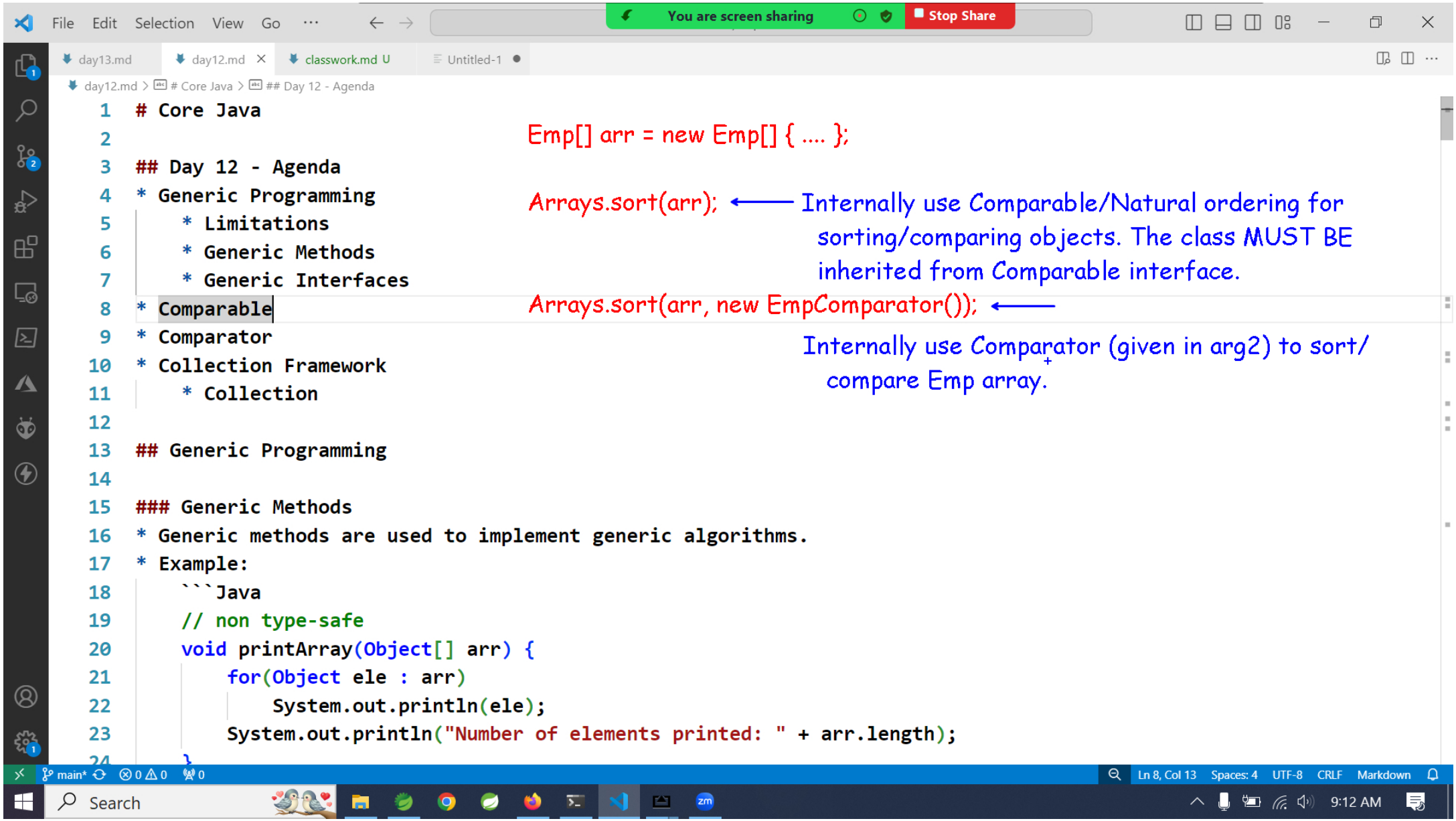
CopyOnWriteArrayList

```
list = new CopyOnWriteArrayList();
Collections.addAll(list, 1,2,3,4,5);

itr1 = list.iterator();
while (itr1.hasNext()) {
    ele = itr1.next();
    sop(ele);
    if (ele == 3)
        itr1.add(6);
}
```




File Edit Selection View Go ···
You are screen sharing
Stop Share
day13.md
day12.md
classwork.md U
Untitled-1
day12.md > # Core Java > ## Day 12 - Agenda
1 # Core Java
2
3 ## Day 12 - Agenda
4 * Generic Programming
5 * Limitations
6 * Generic Methods
7 * Generic Interfaces
8 * Comparable
9 * Comparator
10 * Collection Framework
11 * Collection
12
13 ## Generic Programming
14
15 ### Generic Methods
16 * Generic methods are used to implement generic algorithms.
17 * Example:
18 ```Java
19 // non type-safe
20 void printArray(Object[] arr) {
21 for(Object ele : arr)
22 System.out.println(ele);
23 System.out.println("Number of elements printed: " + arr.length);
Natural Ordering
type-safety
int compareTo(T other);
"this" <=> other
class Emp implements Comparable<Emp> {
// ...
public int compareTo(Emp other) {
int diff = this.id - other.id;
return diff;
}
}
main():
Emp e1 = new Emp(...);
Emp e2 = new Emp(...);
int diff = e1.compareTo(e2);
diff = e1.compareTo("James");
// compiler error -- type-safety
main* 0 0 0
Ln 8, Col 13 Spaces: 4 UTF-8 CRLF Markdown
Search
9:03 AM


File Edit Selection View Go ···
You are screen sharing
Stop Share
day13.md
day12.md
classwork.md U
Untitled-1
day12.md > # Core Java > ## Day 12 - Agenda
1 # Core Java
2
3 ## Day 12 - Agenda
4 * Generic Programming
5 * Limitations
6 * Generic Methods
7 * Generic Interfaces
8 * Comparable
9 * Comparator
10 * Collection Framework
11 * Collection
12
13 ## Generic Programming
14
15 ### Generic Methods
16 * Generic methods are used to implement generic algorithms.
17 * Example:
18 ```Java
19 // non type-safe
20 void printArray(Object[] arr) {
21 for(Object ele : arr)
22 System.out.println(ele);
23 System.out.println("Number of elements printed: " + arr.length);
int compare(T obj1, T obj2);
class Emp {
// ....
}
class EmpComparator
implements Comparator<Emp> {
public int compare(Emp x, Emp y) {
int diff = Double.compare(x.getSal(),
y.getSal());
return diff;
}
}
main():
Emp e1 = new Emp(...);
Emp e2 = new Emp(...);
EmpComparator cmp = new EmpComparator();
int diff = cmp.compare(e1, e2);
main* 0 0 0
Ln 8, Col 13 Spaces: 4 UTF-8 CRLF Markdown
Search
9:08 AM


File Edit Selection View Go ···
You are screen sharing
Stop Share
day13.md
day12.md
classwork.md U
Untitled-1
day12.md > # Core Java > ## Day 12 - Agenda
1 # Core Java
2
3 ## Day 12 - Agenda
4 * Generic Programming
5 * Limitations
6 * Generic Methods
7 * Generic Interfaces
8 * Comparable
9 * Comparator
10 * Collection Framework
11 * Collection
12
13 ## Generic Programming
14
15 ### Generic Methods
16 * Generic methods are used to implement generic algorithms.
17 * Example:
18 ```Java
19 // non type-safe
20 void printArray(Object[] arr) {
21 for(Object ele : arr)
22 System.out.println(ele);
23 System.out.println("Number of elements printed: " + arr.length);
Emp[] arr = new Emp[] { .... };
Arrays.sort(arr);
Internally use Comparable/Natural ordering for sorting/comparing objects. The class MUST BE inherited from Comparable interface.
Arrays.sort(arr, new EmpComparator());
Internally use Comparator (given in arg2) to sort/ compare Emp array.
main*
Ln 8, Col 13
Spaces: 4
UTF-8
CRLF
Markdown
Search
9:12 AM

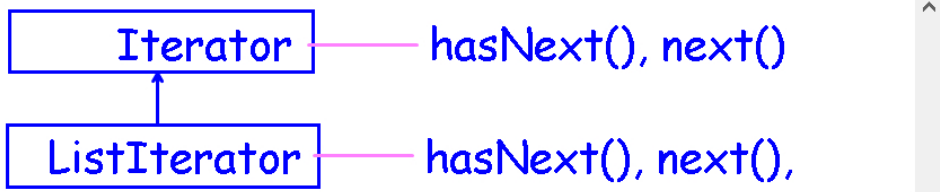
day13 - demo01/src/com/sunbeam/Program01.java - Spring Tool Suite 4

You are screen sharing | Stop Share

File Edit Source Refactor Navigate Search Project Run Window Help

Program01.java

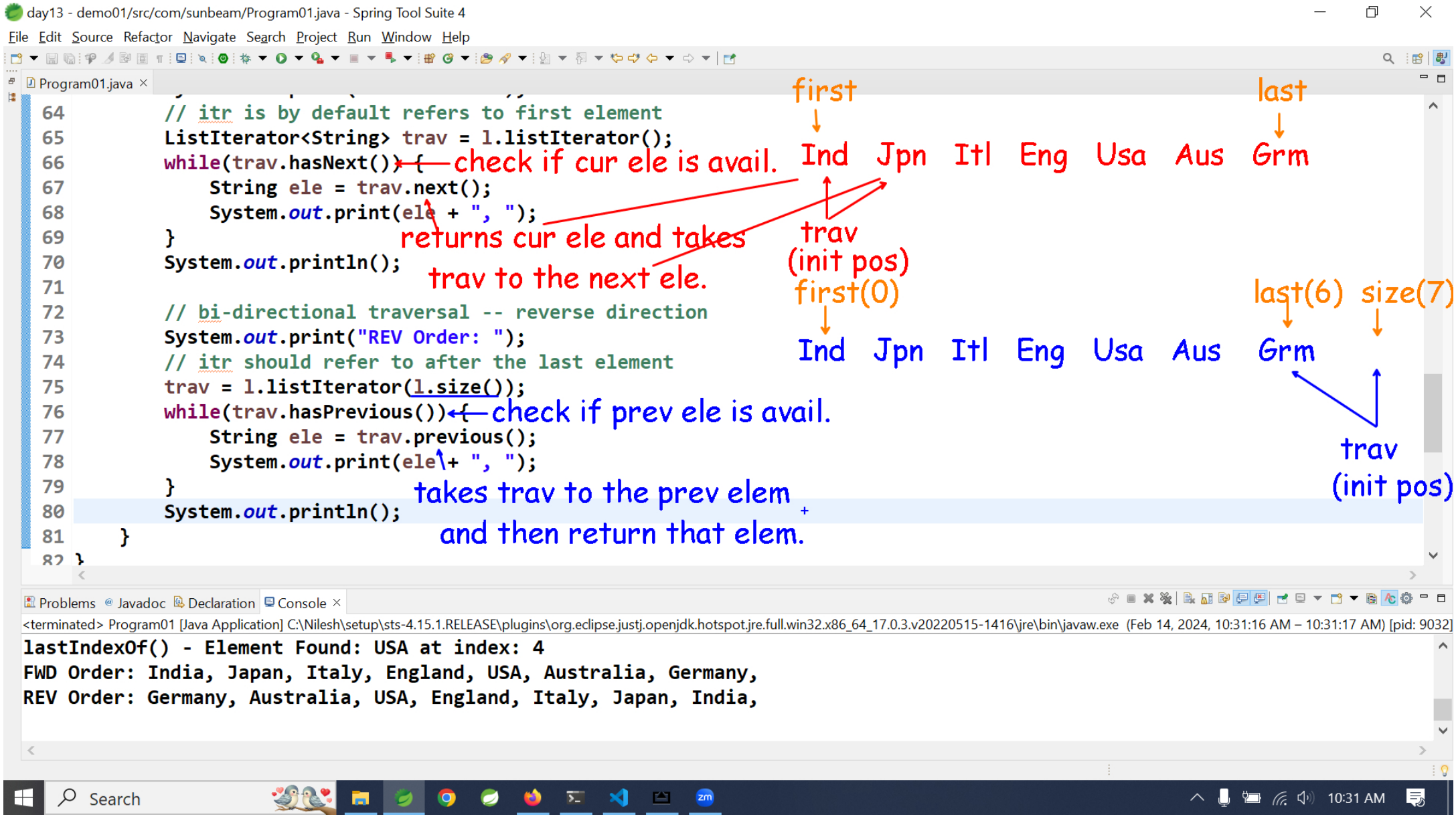
```
60
61          // bi-directional traversal
62      }
63  }
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
```

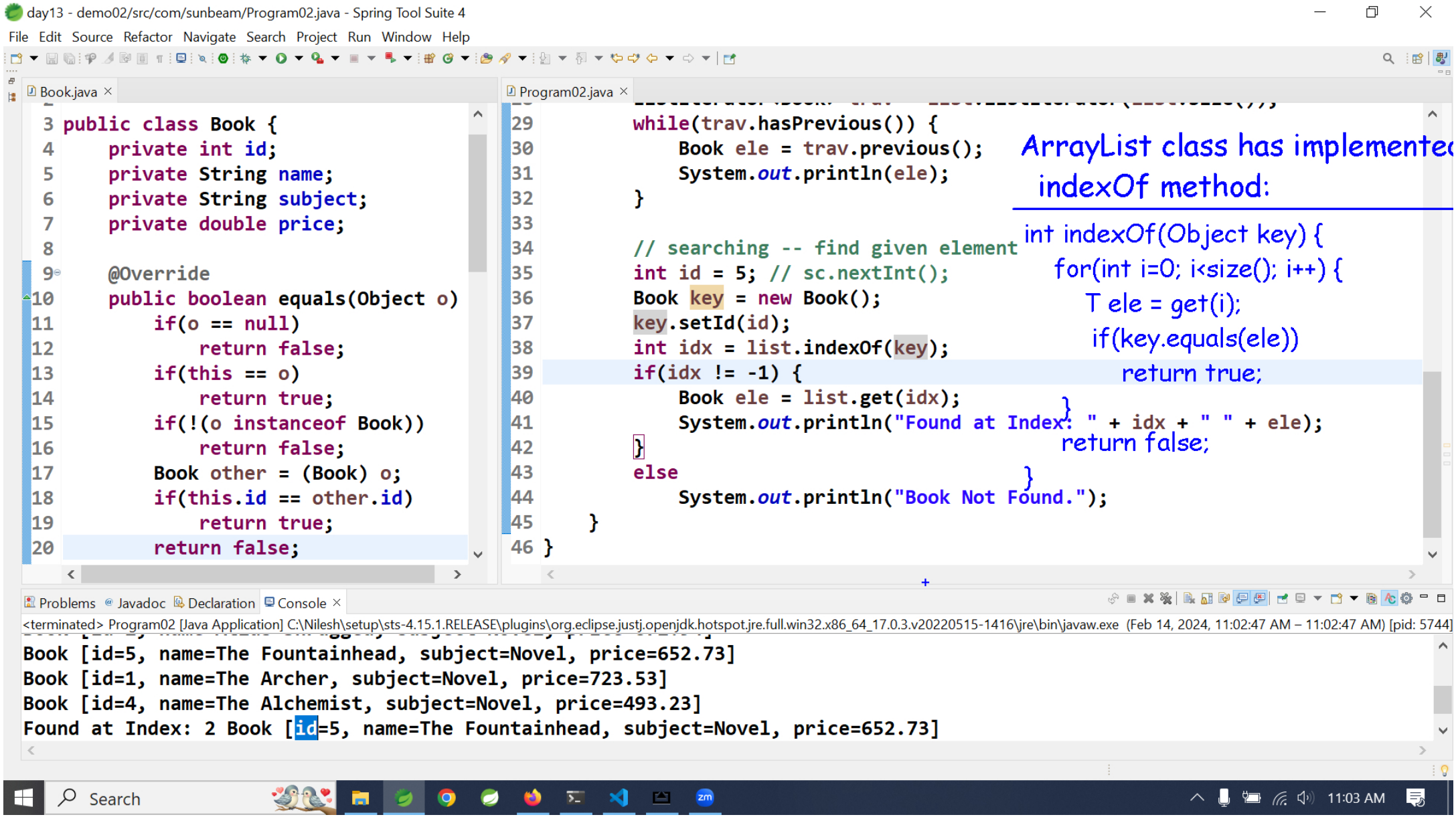


Writable | Smart Insert | 85 : 1 : 2042

Search

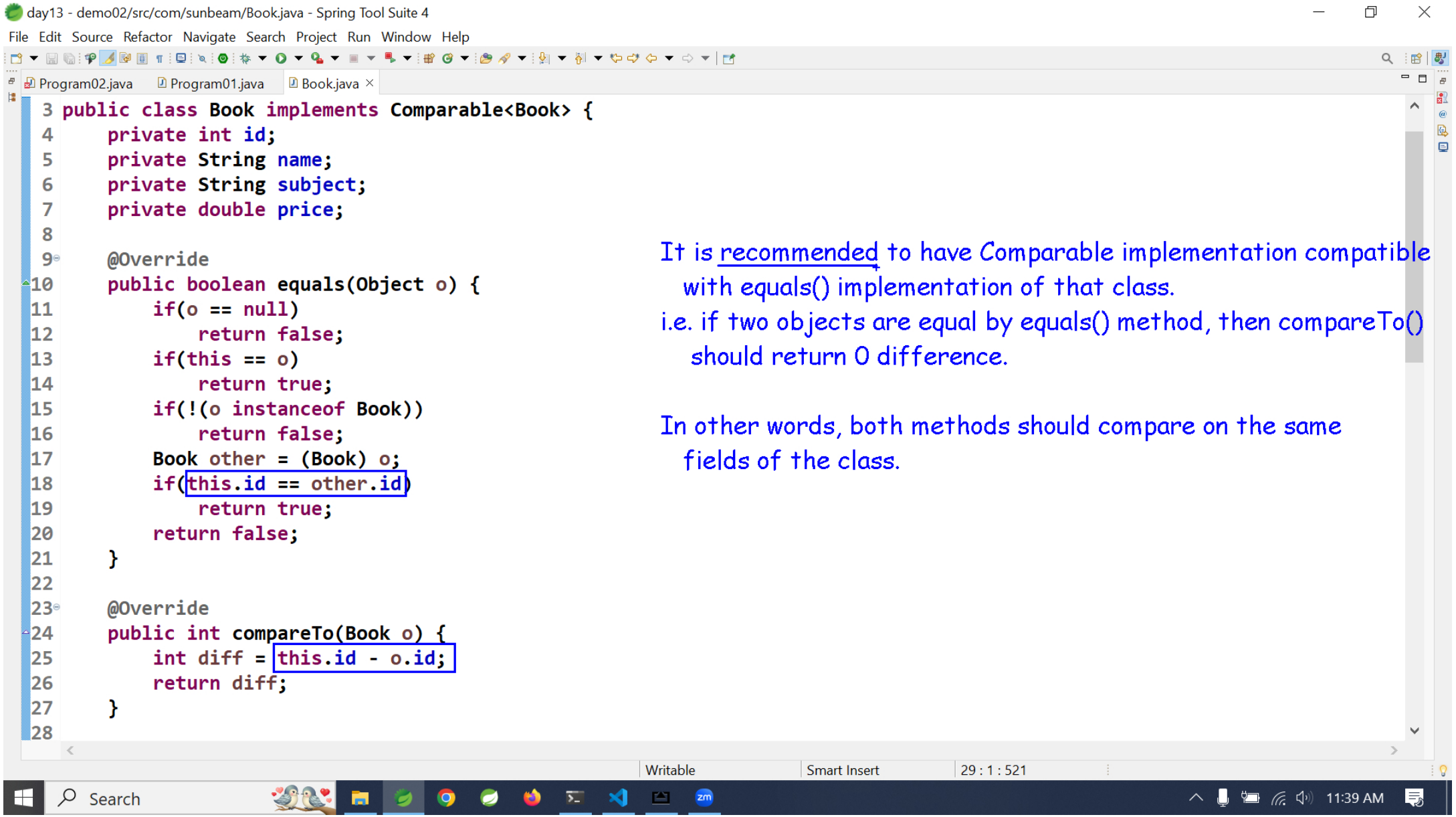
10:23 AM


day13 - demo01/src/com/sunbeam/Program01.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program01.java
// itr is by default refers to first element
ListIterator<String> trav = l.listIterator();
while(trav.hasNext()) {
String ele = trav.next();
System.out.print(ele + ", ");
}
System.out.println();
// bi-directional traversal -- reverse direction
System.out.print("REV Order: ");
// itr should refer to after the last element
trav = l.listIterator(l.size());
while(trav.hasPrevious()) {
String ele = trav.previous();
System.out.print(ele + ", ");
}
System.out.println();
}
}
check if cur ele is avail.
returns cur ele and takes trav to the next ele.
first
last
Ind Jpn Itl Eng Usa Aus Grm
trav (init pos)
first(0)
last(6) size(7)
Ind Jpn Itl Eng Usa Aus Grm
trav (init pos)
check if prev ele is avail.
takes trav to the prev elem and then return that elem.
Problems Javadoc Declaration Console
<terminated> Program01 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\bin\javaw.exe (Feb 14, 2024, 10:31:16 AM – 10:31:17 AM) [pid: 9032]
lastIndexOf() - Element Found: USA at index: 4
FWD Order: India, Japan, Italy, England, USA, Australia, Germany,
REV Order: Germany, Australia, USA, England, Italy, Japan, India,
Search
10:31 AM




```java
public class Book {
    private int id;
    private String name;
    private String subject;
    private double price;

    @Override
    public boolean equals(Object o)
        if(o == null)
            return false;
        if(this == o)
            return true;
        if(!(o instanceof Book))
            return false;
        Book other = (Book) o;
        if(this.id == other.id)
            return true;
        return false;
```

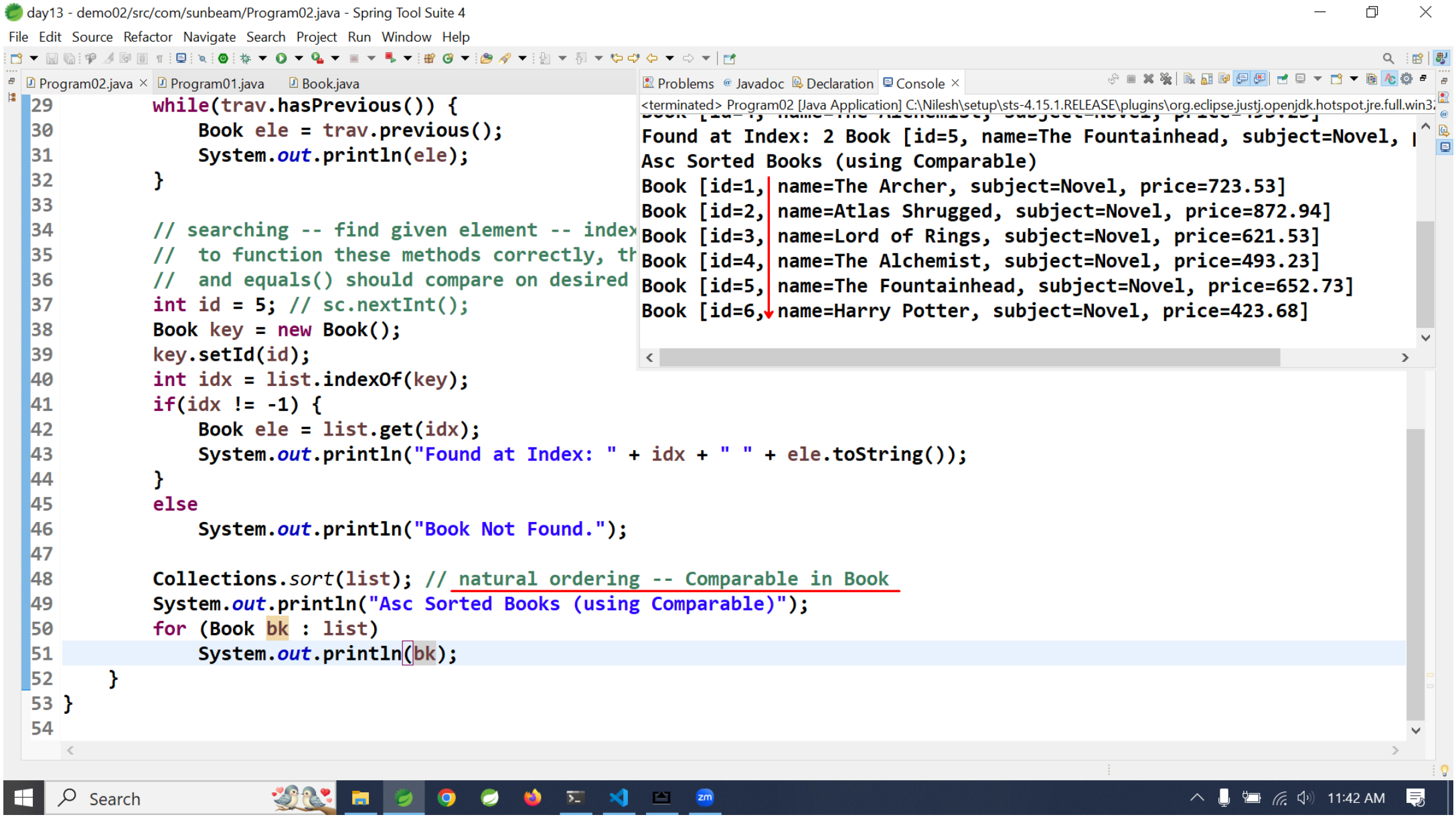
```java
        while(trav.hasPrevious()) {
            Book ele = trav.previous();
            System.out.println(ele);
        }

        // searching -- find given element
        int id = 5; // sc.nextInt();
        Book key = new Book();
        key.setId(id);
        int idx = list.indexOf(key);
        if(idx != -1) {
            Book ele = list.get(idx);
            System.out.println("Found at Index: " + idx + " " + ele);
        }
        else
            System.out.println("Book Not Found.");
    }
}
```

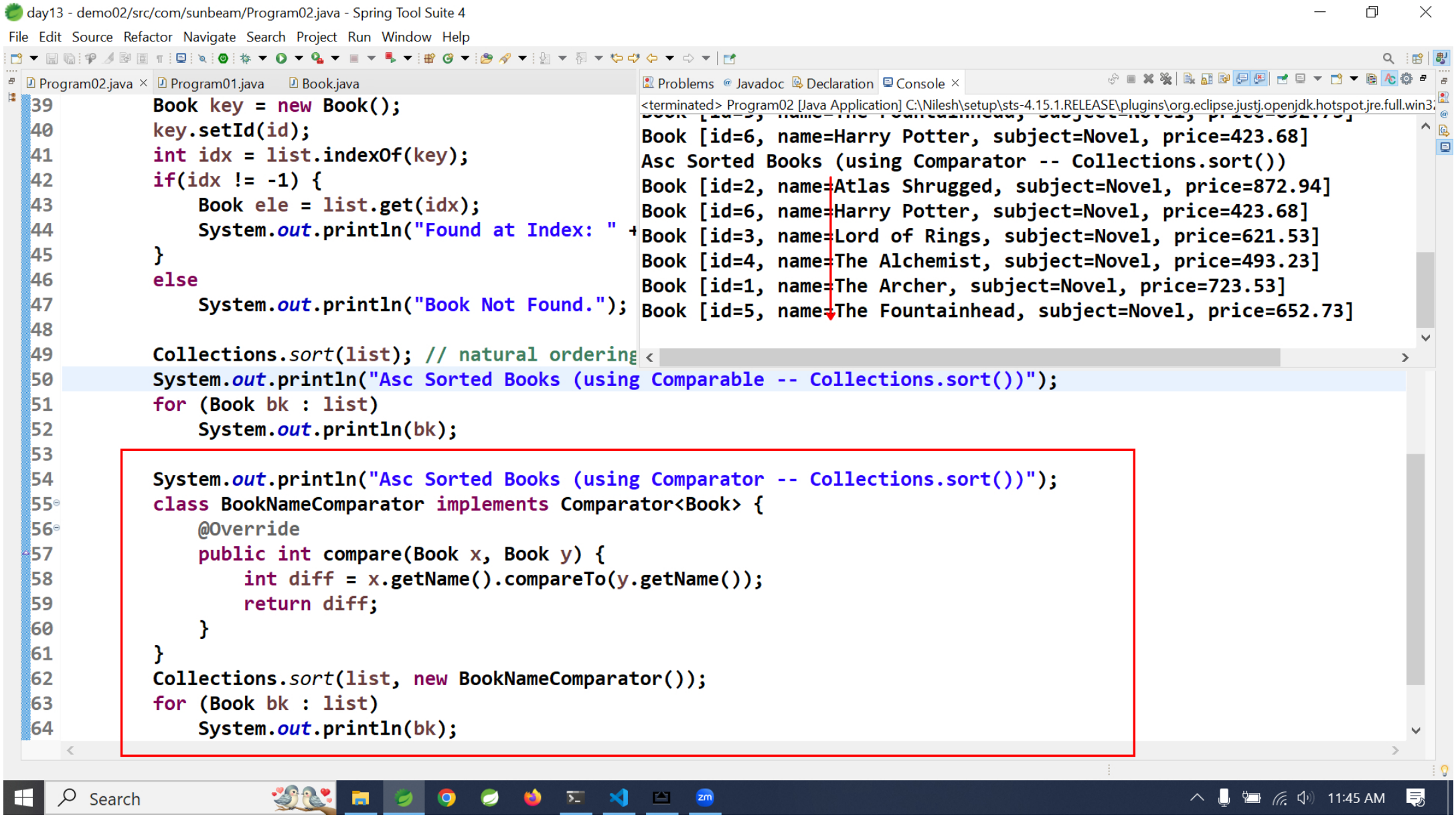


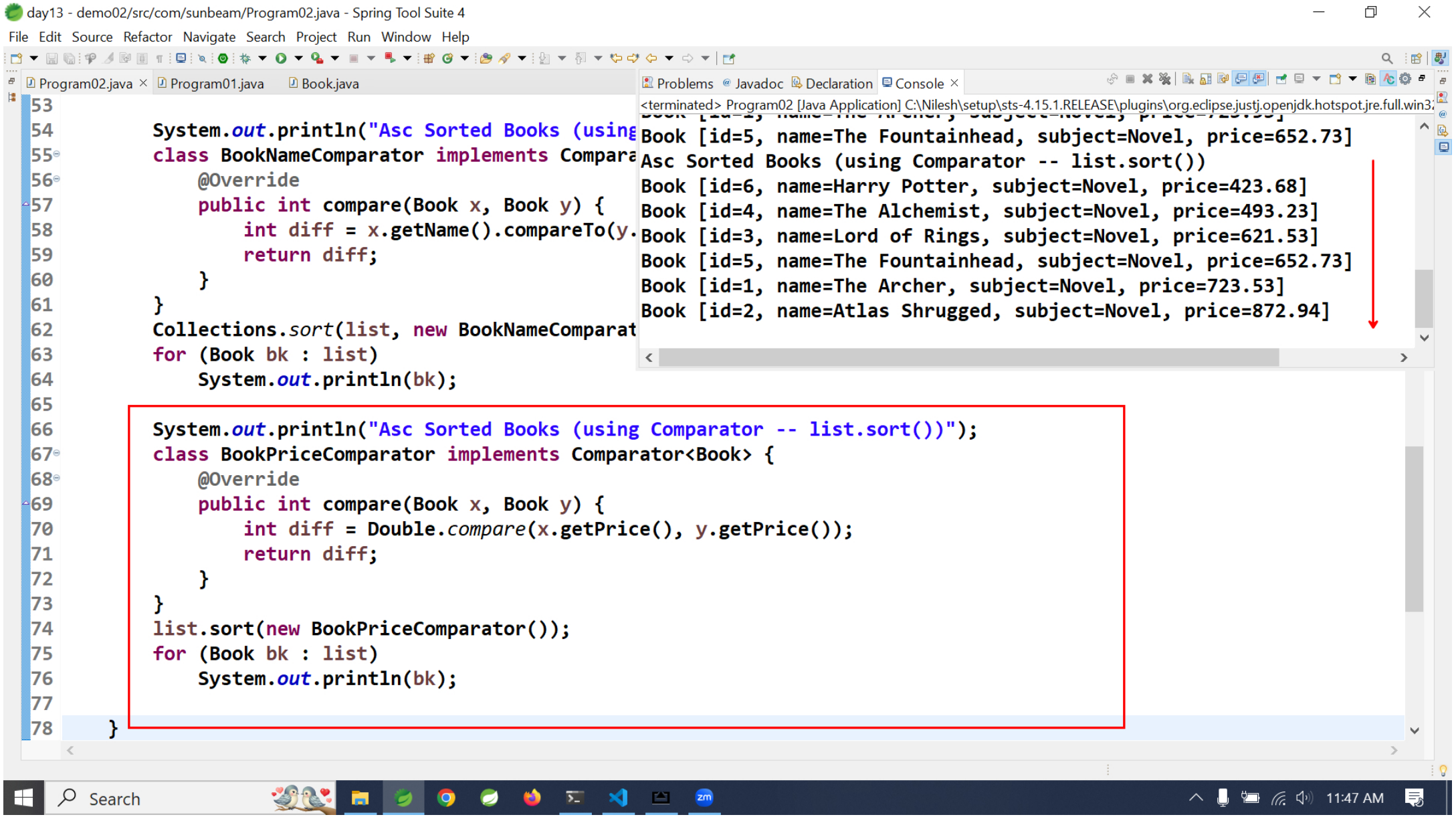
It is recommended to have Comparable implementation compatible with equals() implementation of that class.

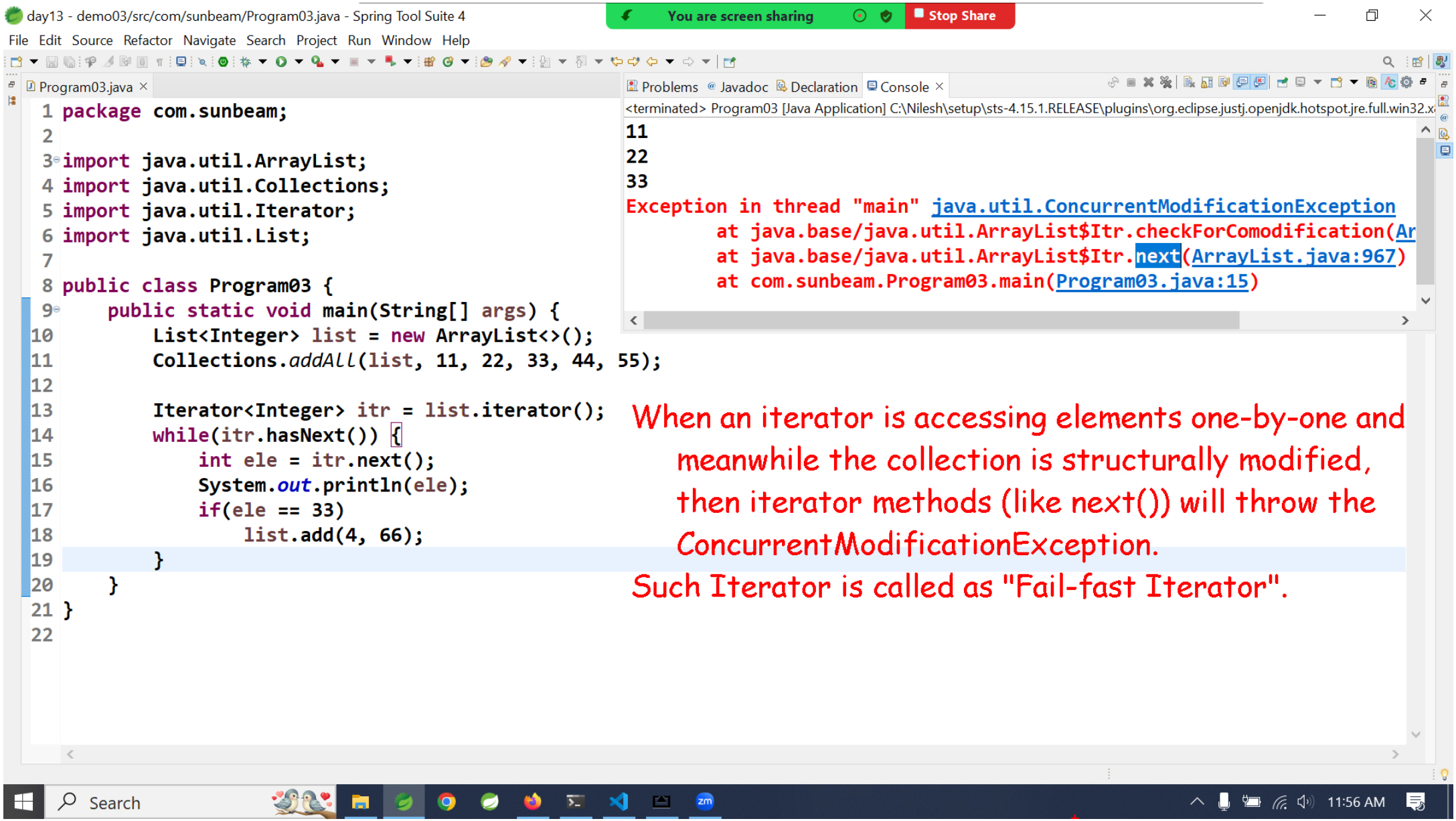
i.e. if two objects are equal by equals() method, then compareTo() should return 0 difference.

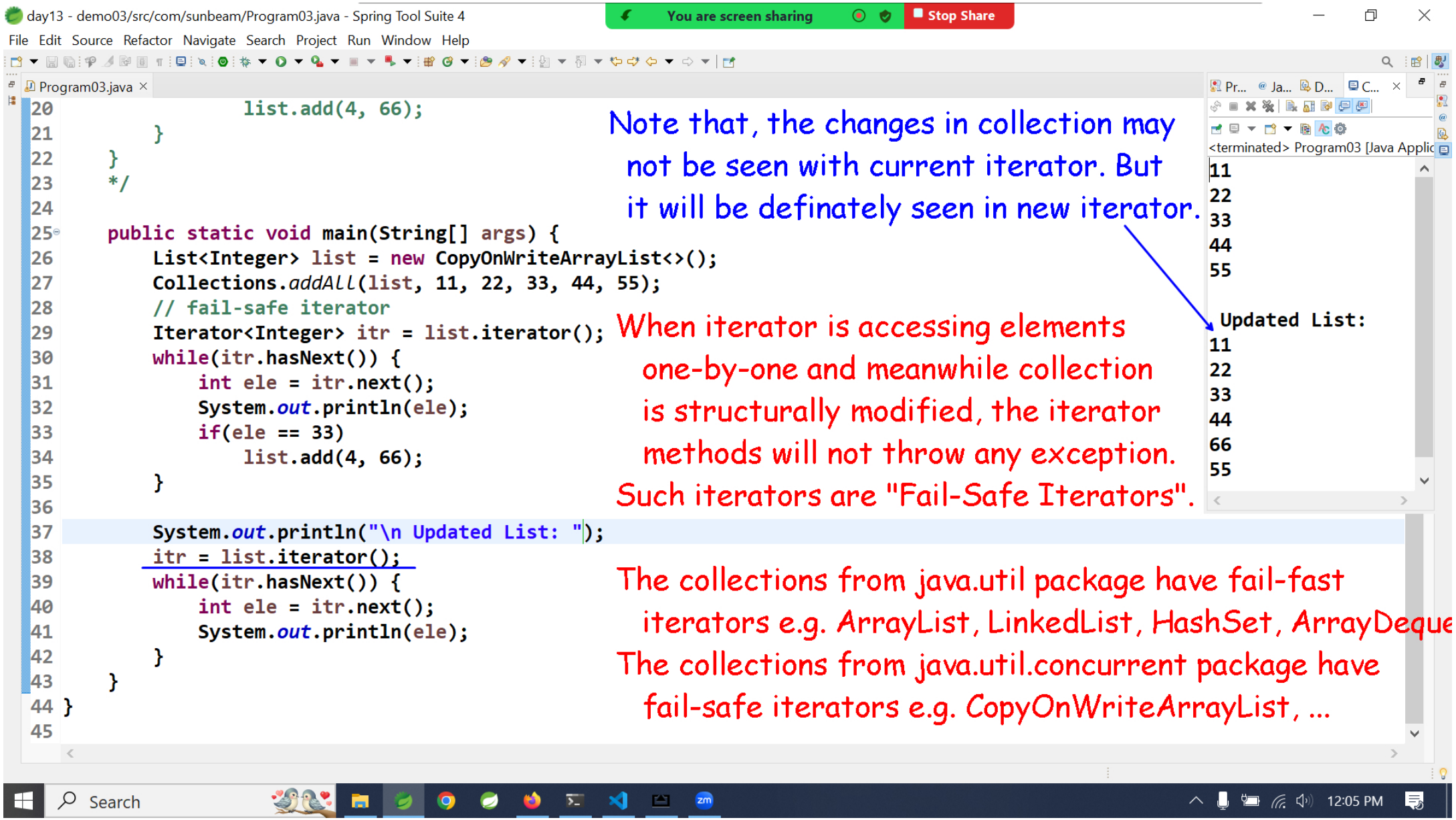
In other words, both methods should compare on the same fields of the class.

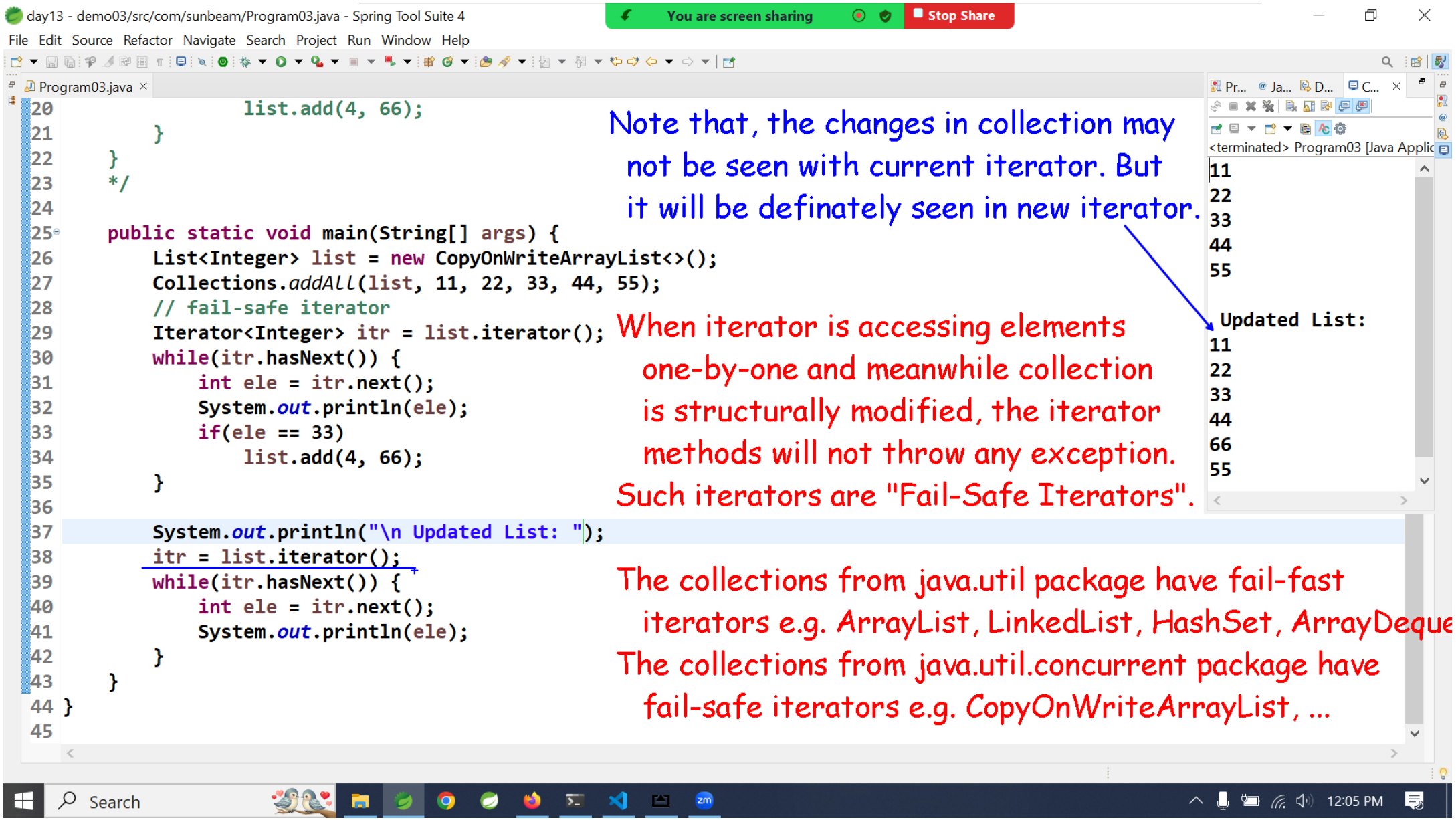

day13 - demo02/src/com/sunbeam/Program02.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program02.java
Program01.java
Book.java
Problems
Javadoc
Declaration
Console
<terminated> Program02 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
Found at Index: 2 Book [id=5, name=The Fountainhead, subject=Novel,
Asc Sorted Books (using Comparable)
Book [id=1, name=The Archer, subject=Novel, price=723.53]
Book [id=2, name=Atlas Shrugged, subject=Novel, price=872.94]
Book [id=3, name=Lord of Rings, subject=Novel, price=621.53]
Book [id=4, name=The Alchemist, subject=Novel, price=493.23]
Book [id=5, name=The Fountainhead, subject=Novel, price=652.73]
Book [id=6, name=Harry Potter, subject=Novel, price=423.68]
29 while(trav.hasPrevious()) {
30 Book ele = trav.previous();
31 System.out.println(ele);
32 }
33
34 // searching -- find given element -- index
35 // to function these methods correctly, th
36 // and equals() should compare on desired
37 int id = 5; // sc.nextInt();
38 Book key = new Book();
39 key.setId(id);
40 int idx = list.indexOf(key);
41 if(idx != -1) {
42 Book ele = list.get(idx);
43 System.out.println("Found at Index: " + idx + " " + ele.toString());
44 }
45 else
46 System.out.println("Book Not Found.");
47
48 Collections.sort(list); // natural ordering -- Comparable in Book
49 System.out.println("Asc Sorted Books (using Comparable)");
50 for (Book bk : list)
51 System.out.println(bk);
52 }
53 }
54
Search
11:42 AM

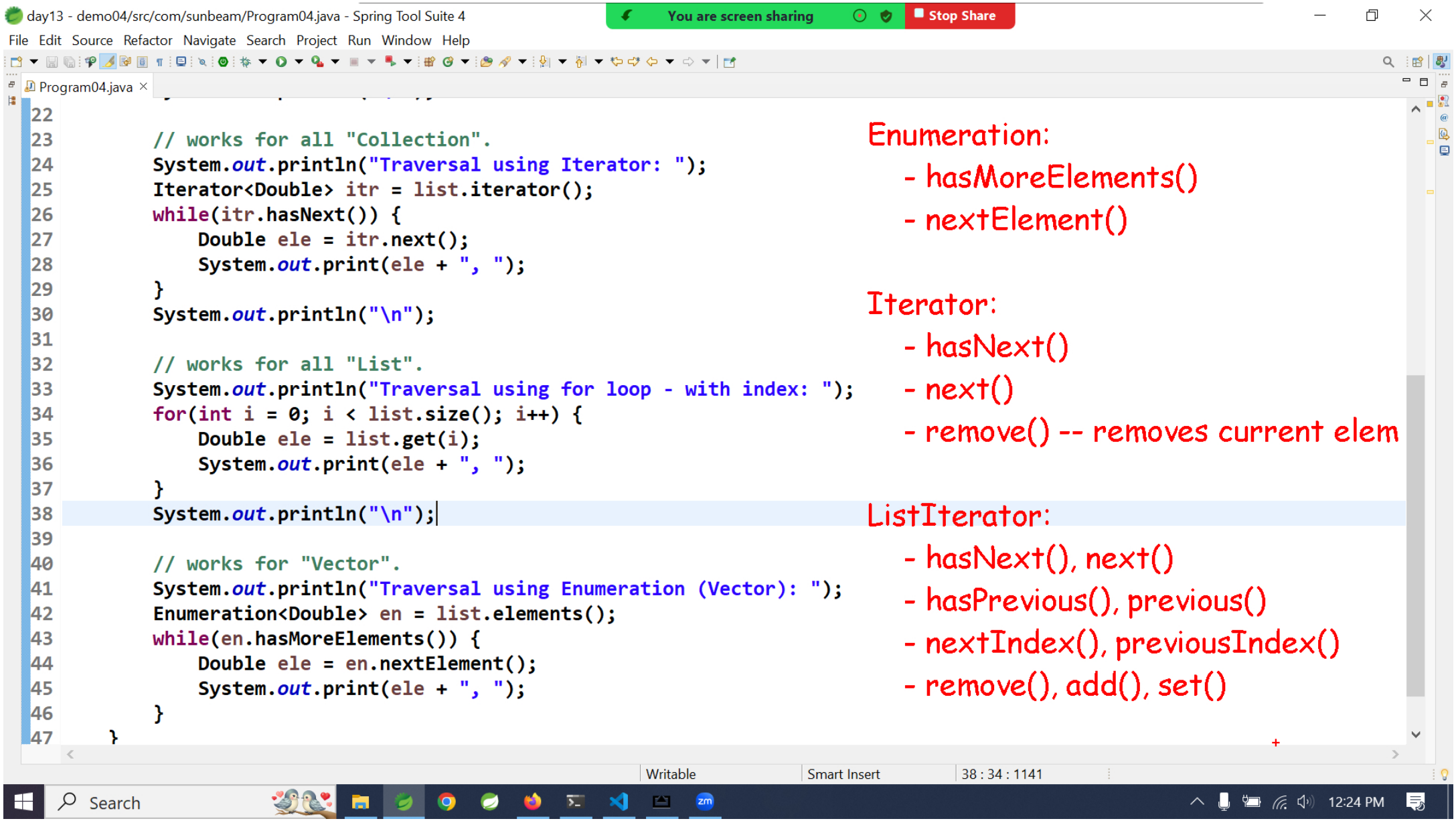

day13 - demo02/src/com/sunbeam/Program02.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program02.java
Program01.java
Book.java
Problems
Javadoc
Declaration
Console
<terminated> Program02 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
Book [id=6, name=Harry Potter, subject=Novel, price=423.68]
Asc Sorted Books (using Comparator -- Collections.sort())
Book [id=2, name=Atlas Shrugged, subject=Novel, price=872.94]
Book [id=6, name=Harry Potter, subject=Novel, price=423.68]
Book [id=3, name=Lord of Rings, subject=Novel, price=621.53]
Book [id=4, name=The Alchemist, subject=Novel, price=493.23]
Book [id=1, name=The Archer, subject=Novel, price=723.53]
Book [id=5, name=The Fountainhead, subject=Novel, price=652.73]
Book key = new Book();
key.setId(id);
int idx = list.indexOf(key);
if(idx != -1) {
Book ele = list.get(idx);
System.out.println("Found at Index: " +
}
else
System.out.println("Book Not Found.");
Collections.sort(list); // natural ordering
System.out.println("Asc Sorted Books (using Comparable -- Collections.sort())");
for (Book bk : list)
System.out.println(bk);
System.out.println("Asc Sorted Books (using Comparator -- Collections.sort())");
class BookNameComparator implements Comparator<Book> {
@Override
public int compare(Book x, Book y) {
int diff = x.getName().compareTo(y.getName());
return diff;
}
}
Collections.sort(list, new BookNameComparator());
for (Book bk : list)
System.out.println(bk);
Search
11:45 AM


day13 - demo02/src/com/sunbeam/Program02.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program02.java
Program01.java
Book.java
Problems
Javadoc
Declaration
Console
<terminated> Program02 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
Book [id=5, name=The Fountainhead, subject=Novel, price=652.73]
Asc Sorted Books (using Comparator -- list.sort())
Book [id=6, name=Harry Potter, subject=Novel, price=423.68]
Book [id=4, name=The Alchemist, subject=Novel, price=493.23]
Book [id=3, name=Lord of Rings, subject=Novel, price=621.53]
Book [id=5, name=The Fountainhead, subject=Novel, price=652.73]
Book [id=1, name=The Archer, subject=Novel, price=723.53]
Book [id=2, name=Atlas Shrugged, subject=Novel, price=872.94]
53
54 System.out.println("Asc Sorted Books (using
55 class BookNameComparator implements Compara
56 @Override
57 public int compare(Book x, Book y) {
58 int diff = x.getName().compareTo(y.
59 return diff;
60 }
61 }
62 Collections.sort(list, new BookNameComparat
63 for (Book bk : list)
64 System.out.println(bk);
65
66 System.out.println("Asc Sorted Books (using Comparator -- list.sort())");
67 class BookPriceComparator implements Comparator<Book> {
68 @Override
69 public int compare(Book x, Book y) {
70 int diff = Double.compare(x.getPrice(), y.getPrice());
71 return diff;
72 }
73 }
74 list.sort(new BookPriceComparator());
75 for (Book bk : list)
76 System.out.println(bk);
77
78 }
Search
11:47 AM


day13 - demo03/src/com/sunbeam/Program03.java - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Program03.java
1 package com.sunbeam;
2
3 import java.util.ArrayList;
4 import java.util.Collections;
5 import java.util.Iterator;
6 import java.util.List;
7
8 public class Program03 {
9 public static void main(String[] args) {
10 List<Integer> list = new ArrayList<>();
11 Collections.addAll(list, 11, 22, 33, 44, 55);
12
13 Iterator<Integer> itr = list.iterator();
14 while(itr.hasNext()) {
15 int ele = itr.next();
16 System.out.println(ele);
17 if(ele == 33)
18 list.add(4, 66);
19 }
20 }
21 }
22
Problems Javadoc Declaration Console
<terminated> Program03 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x
11
22
33
Exception in thread "main" java.util.ConcurrentModificationException
at java.base/java.util.ArrayList$Itr.checkForComodification(Ar
at java.base/java.util.ArrayList$Itr.next(ArrayList.java:967)
at com.sunbeam.Program03.main(Program03.java:15)
When an iterator is accessing elements one-by-one and meanwhile the collection is structurally modified, then iterator methods (like next()) will throw the ConcurrentModificationException.
Such Iterator is called as "Fail-fast Iterator".
Search
11:56 AM

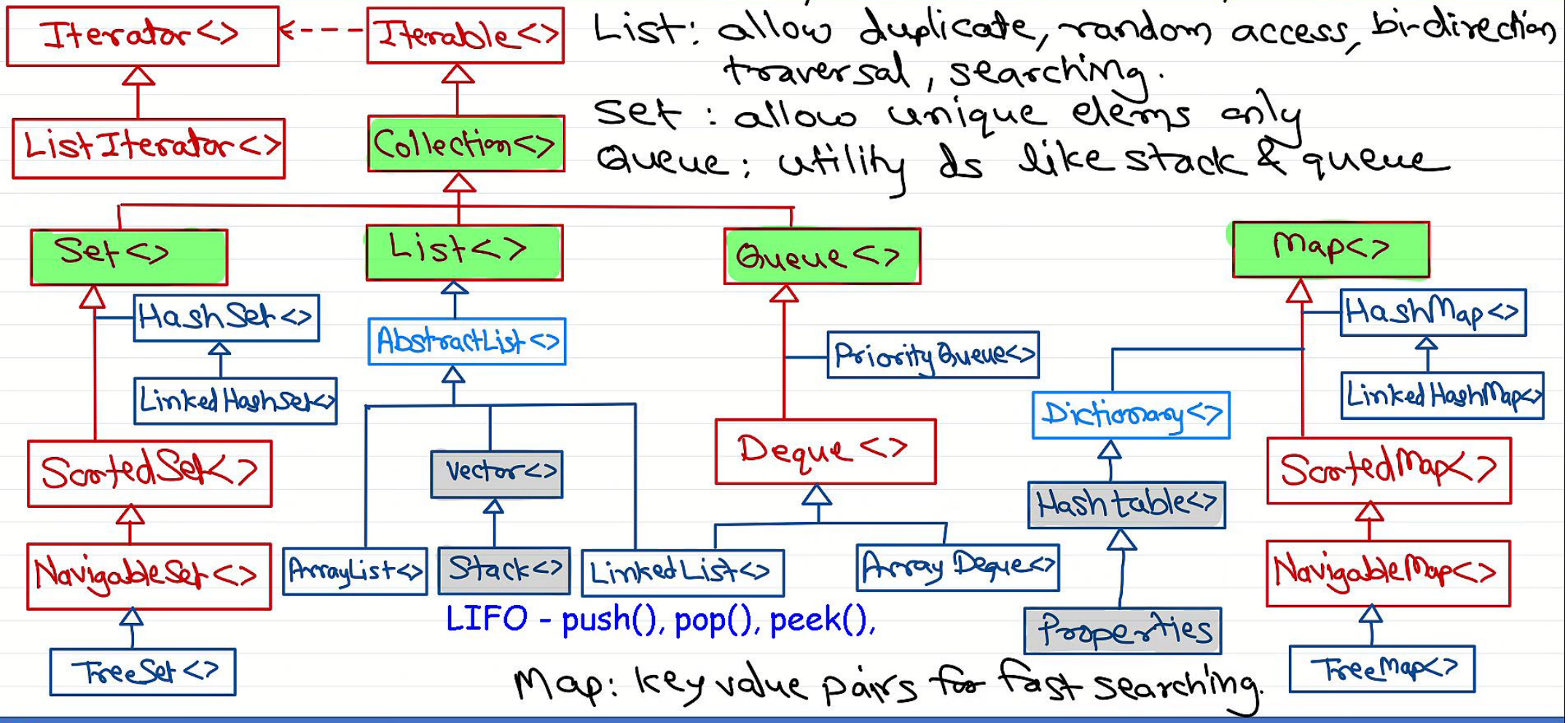

day13 - demo03/src/com/sunbeam/Program03.java - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Program03.java
list.add(4, 66);
}
}
*/
public static void main(String[] args) {
List<Integer> list = new CopyOnWriteArrayList<>();
Collections.addAll(list, 11, 22, 33, 44, 55);
// fail-safe iterator
Iterator<Integer> itr = list.iterator();
while(itr.hasNext()) {
int ele = itr.next();
System.out.println(ele);
if(ele == 33)
list.add(4, 66);
}
System.out.println("\n Updated List: ");
itr = list.iterator();
while(itr.hasNext()) {
int ele = itr.next();
System.out.println(ele);
}
}
}
Note that, the changes in collection may not be seen with current iterator. But it will be definately seen in new iterator.
When iterator is accessing elements one-by-one and meanwhile collection is structurally modified, the iterator methods will not throw any exception. Such iterators are "Fail-Safe Iterators".
<terminated> Program03 [Java Applic
11
22
33
44
55
Updated List:
11
22
33
44
66
55
The collections from java.util package have fail-fast iterators e.g. ArrayList, LinkedList, HashSet, ArrayDeque
The collections from java.util.concurrent package have fail-safe iterators e.g. CopyOnWriteArrayList, ...
Search
12:05 PM

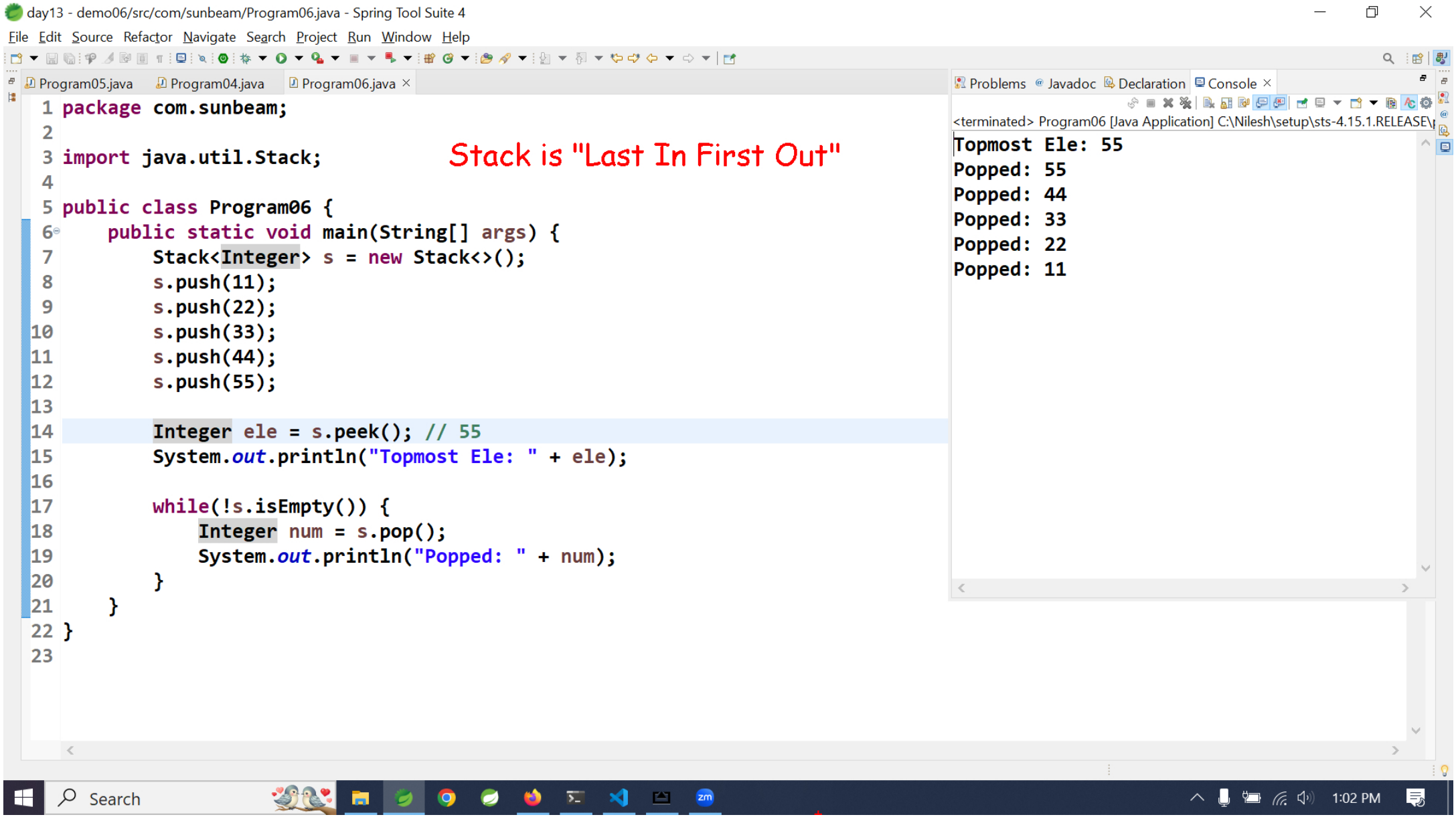

day13 - demo03/src/com/sunbeam/Program03.java - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Program03.java
list.add(4, 66);
}
}
*/
public static void main(String[] args) {
List<Integer> list = new CopyOnWriteArrayList<>();
Collections.addAll(list, 11, 22, 33, 44, 55);
// fail-safe iterator
Iterator<Integer> itr = list.iterator();
while(itr.hasNext()) {
int ele = itr.next();
System.out.println(ele);
if(ele == 33)
list.add(4, 66);
}
System.out.println("\n Updated List: ");
itr = list.iterator();
while(itr.hasNext()) {
int ele = itr.next();
System.out.println(ele);
}
}
}
Note that, the changes in collection may not be seen with current iterator. But it will be definately seen in new iterator.
When iterator is accessing elements one-by-one and meanwhile collection is structurally modified, the iterator methods will not throw any exception. Such iterators are "Fail-Safe Iterators".
The collections from java.util package have fail-fast iterators e.g. ArrayList, LinkedList, HashSet, ArrayDeque
The collections from java.util.concurrent package have fail-safe iterators e.g. CopyOnWriteArrayList, ...
<terminated> Program03 [Java Applic
11
22
33
44
55
Updated List:
11
22
33
44
66
55
Search
12:05 PM


day13 - demo04/src/com/sunbeam/Program04.java - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Program04.java
// works for all "Collection".
System.out.println("Traversal using Iterator: ");
Iterator<Double> itr = list.iterator();
while(itr.hasNext()) {
Double ele = itr.next();
System.out.print(ele + ", ");
}
System.out.println("\n");
// works for all "List".
System.out.println("Traversal using for loop - with index: ");
for(int i = 0; i < list.size(); i++) {
Double ele = list.get(i);
System.out.print(ele + ", ");
}
System.out.println("\n");
// works for "Vector".
System.out.println("Traversal using Enumeration (Vector): ");
Enumeration<Double> en = list.elements();
while(en.hasMoreElements()) {
Double ele = en.nextElement();
System.out.print(ele + ", ");
}
}
Enumeration:
- hasMoreElements()
- nextElement()
Iterator:
- hasNext()
- next()
- remove() -- removes current elem
ListIterator:
- hasNext(), next()
- hasPrevious(), previous()
- nextIndex(), previousIndex()
- remove(), add(), set()
Writable
Smart Insert
38 : 34 : 1141
Search
12:24 PM

Java collection framework

Collection: basic collection i.e. add ele, remove ele, traverse collection, etc.



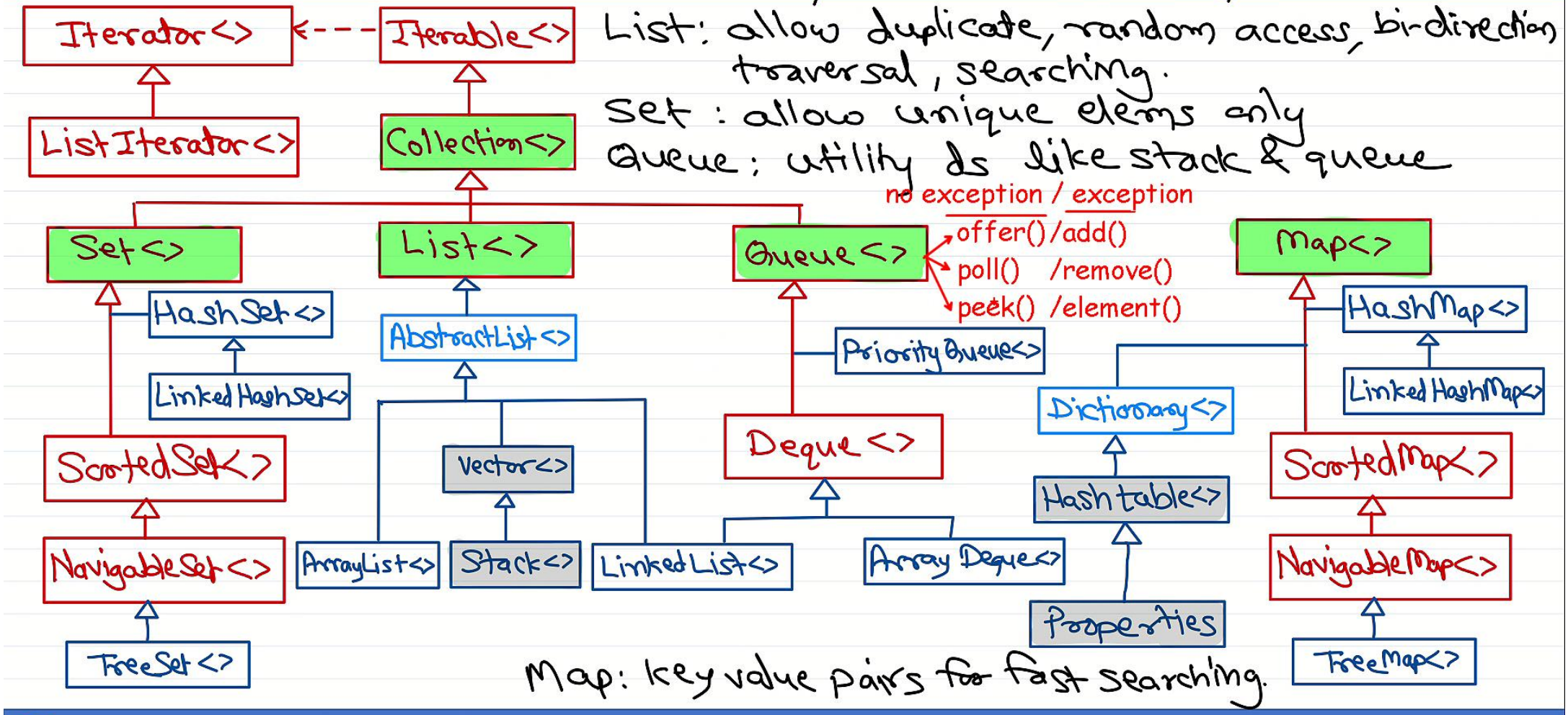


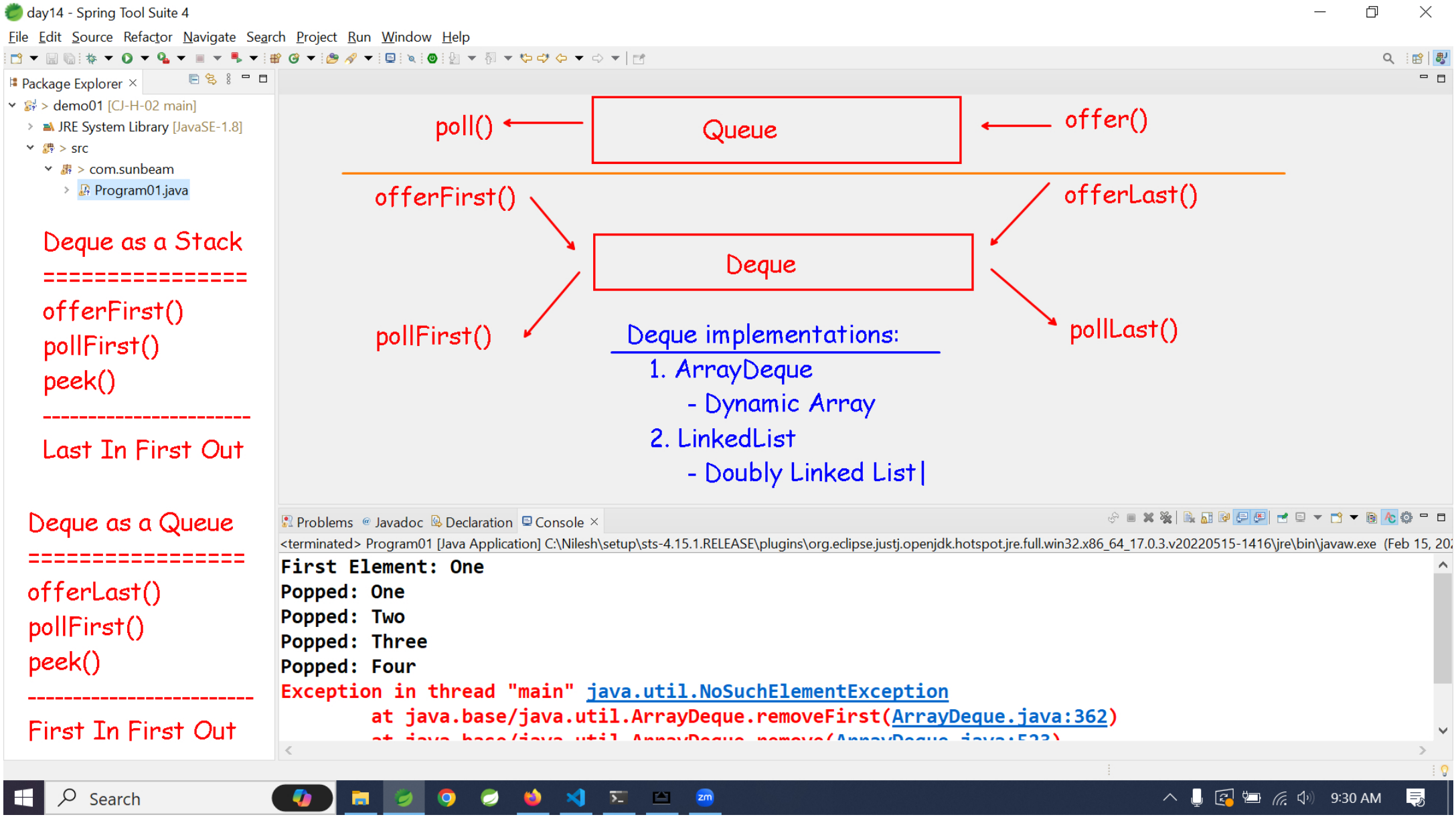
Stack is "Last In First Out"

```java
package com.sunbeam;

import java.util.Stack;

public class Program06 {
    public static void main(String[] args) {
        Stack<Integer> s = new Stack<>();
        s.push(11);
        s.push(22);
        s.push(33);
        s.push(44);
        s.push(55);

        Integer ele = s.peek(); // 55
        System.out.println("Topmost Ele: " + ele);

        while(!s.isEmpty()) {
            Integer num = s.pop();
            System.out.println("Popped: " + num);
        }
    }
}
```

Java collection framework

Collection: basic collection i.e. add ele, remove ele, traverse collection, etc.




day14 - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
> demo01 [CJ-H-02 main]
JRE System Library [JavaSE-1.8]
> src
> com.sunbeam
Program01.java
Deque as a Stack
==========
offerFirst()
pollFirst()
peek()

Last In First Out
Deque as a Queue
==========
offerLast()
pollFirst()
peek()

First In First Out
poll()
Queue
offer()
offerFirst()
offerLast()
Deque
pollFirst()
pollLast()
Deque implementations:
1. ArrayDeque
- Dynamic Array
2. LinkedList
- Doubly Linked List
Problems Javadoc Declaration Console
<terminated> Program01 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\bin\javaw.exe
First Element: One
Popped: One
Popped: Two
Popped: Three
Popped: Four
Exception in thread "main" java.util.NoSuchElementException
at java.base/java.util.ArrayDeque.removeFirst(ArrayDeque.java:362)
Search
9:30 AM

Revision

- Cloneable Interface -- marker interface - Empty Interface
- Object class - protected Object clone(){ }
- Shallow copy of reference -- Outer - Object copy
- Deep copy -- Outer Object -- Inner Object

- Garbage Collection
  - nullify the reference
  - reassign the reference
  - Object created locally inside the method
- Finalize method
  - Before destroying the Object GC calls finalize method(resource release)
  - But gc gets delayed finalize method will not get called (resource leakage)
- String
  - String literal / constant

```
String str = "Sunbeam"; // String pool
```

  - using new Keyword

```
String st = new String("Infotech"); //Heap
```

- StringBuffer
  - Legacy -- java 1.0 / Mutable
  - Synchronized - Slower - Thread Safe
  -StringBuffer sb1 = new StringBuffer(..);
   append()
- StringBuilder
  - Non-Synchronized - faster
  - Thread unsafe
   -StringBuilder sb1 = new StringBuilder("Sunbeam");
  - Mutable

- Resource management
  - close() of Closeable interface -- should be called manually -- finally block
  - try-resource -- AutoCloseable interface - close() will be called automatically
- JVM arch |

java generics till 1.4 using Object class

```
class Box{
  private Object obj;
  public void setObj(Object obj){
     this.obj = obj
  }
  public Object getObj(){
    return this.obj;
  }
}
```

```
Box b1 = new Box();
b1.setObj(new Integer(100));
Integer r1 = (Integer)b1.getObj();
sysout(r1); // 100
```

```
Box b2 = new Box();
b2.setObj(new Double(11.33));
Double r2 = (Double)b2.getObj();
sysout(r2); //11.33
```

```
Box b3 = new Box();
b3.setObj(new Long(1));
Long r3 = (Long)b3.getObj();
sysout(r3); // 1
```

```
Box b4 = new Box();
b4.setObj(new Date());
Date r4 = (Date) b4.getObj();
sysout(r4);
```

```
Box b5 = new Box();
b5.setObj("Hello");
String r5 = (String)b5.getObj();
sysout(r5);
```

```
Box b6 = new Box();
b6.setObj("123");
Integer r6 = (Integer) b6.getObj();//ClassCastExc
sysout(r6);
```

Type checking is done at runtime (no type safe)

```
List list = new ArrayList
list.add(10);
list.add(20);
list.add(30);
list.add("40");
for(Object O : list)
 Integer ele = (Integer)
                    O;
sysout(ele);
}
```

Type-unsafe - Type checking is done at runtime

//java generic 1.5 and above

```
class Box<TYPE> {
  private TYPE obj;
 public void setObj(TYPE obj){
      this.obj = obj;
 }
public TYPE getObj(){
    return this.obj
}
```

```
Box<Integer> b1 = new Box<Integer>();
b1.setObj(new Integer(100));
Integer r1 = b1.getObj();
sysout(r1); // 100
```

```
Box<Double> b2 = new Box<Double>();
b2.setObj(new Double(11.33));
Double r2 = b2.getObj();
sysout(r2); // 11.33
```

```
Box<Long> b3 = new Box<Long>();
b3.setObj(new Long(11));
Long r3 = b3.getObj();
sysout(r3); // 11

Box<Date> b4 = new Box<Date>();
b4.setObj(new Date());
Date r4 = n4.getObj();
sysout(r4);

Box<Integer> b5 = new Box<Integer>();
b5.setObj("123"); // Compiler-Error
      //compiletime type checking

List<Integer>list = new ArrayList<Integer>();
list.add(10);
list.add(20);
list.add("30");
for(Integer ele : list)
    sysout(ele);
```

Compiler Error (type-checking at compiletime) |

Revision

- Problems - Runtime problems
  - Errors
    - Not recoverable
  - Exception
    - Recoverable
- Runtime Problems -- Throwable
- try , catch, throw , throws ,finally
- try - method whose execution may fail must be tried , "try" to execute
- catch - Handling logic
- Checked exception
  - Checked by java compiler
  - Programmer must be aware to handle it
    - throws or catch imp
  - Most of the problems arise out of JVM
  - Subclass of Exception (but not Runtime Exception) |

- unchecked Exception
  - Not checked by java compiler
  - Most of problems arises
    - Programmer's Mistake
    - User's Mistake
  - If we dont handle exception JVM will handle it and terminate the appln
  - Runtime Exception -- subclasses - Unchecked exception

- Throwable -- Generic catch - All the exception can be handled
  - printStackTrace();
  - getMessage();
- Custom-Exception class |

- Genrics
  - Generic class
  - Generic method
  - Generic Interface

- Java 1.4 -- Generics -- Object class
  - Not type-safety , type - unsafe
  - Type-checking is done at runtime
- Java 5.0 - Above - Generics
  - Type-checking is done at Compile-time
  - Type-safe code
- class Box<T> // generic class
  class Box<T extends Number> // Bounded Generic type
- class Box<T>
  - Box<?> // ? - wildcard -- UnboundedGeneric type reference
  - Box<? extends Number> - Upper-Bounded Generic type reference
  - Box<? super Number> - Lower -bounded Generic type reference |

- Generic methods
  - Generic method using Object class (1.4)
  - Generic method

```
public static <T> void printTypedArray y(T[] arr){
}
```

  - Bounded

```
public static <T extends Number< void printNumberArray(T[] arr){
..
} |
```

Generic Interfaces

- Comparable
  - Natural Ordering
  - Typical Comparison implementation is written inside the class
  - BuiltIn Ordering
  - int compareTo(T obj);
  - Arrays.sort(arr);
    - It will call compareTo method of Comparable interface if and when elements needs to be compared
  - java.lang package

- Comparator
  - java.util package
  - Typical Comparison implementation is written outside the class
  - Arrays.sort(arr, ...);
    - It will call compare method of Comparator interface if and when elements needs to be compared
  - int compare(T obj1 , T obj2); |