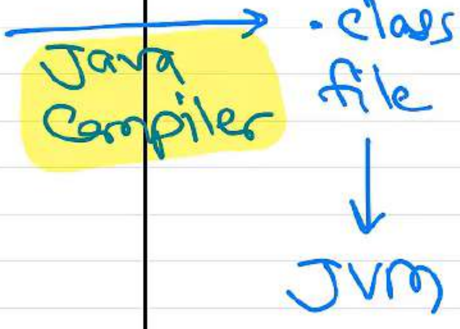
Since Java 5.0

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



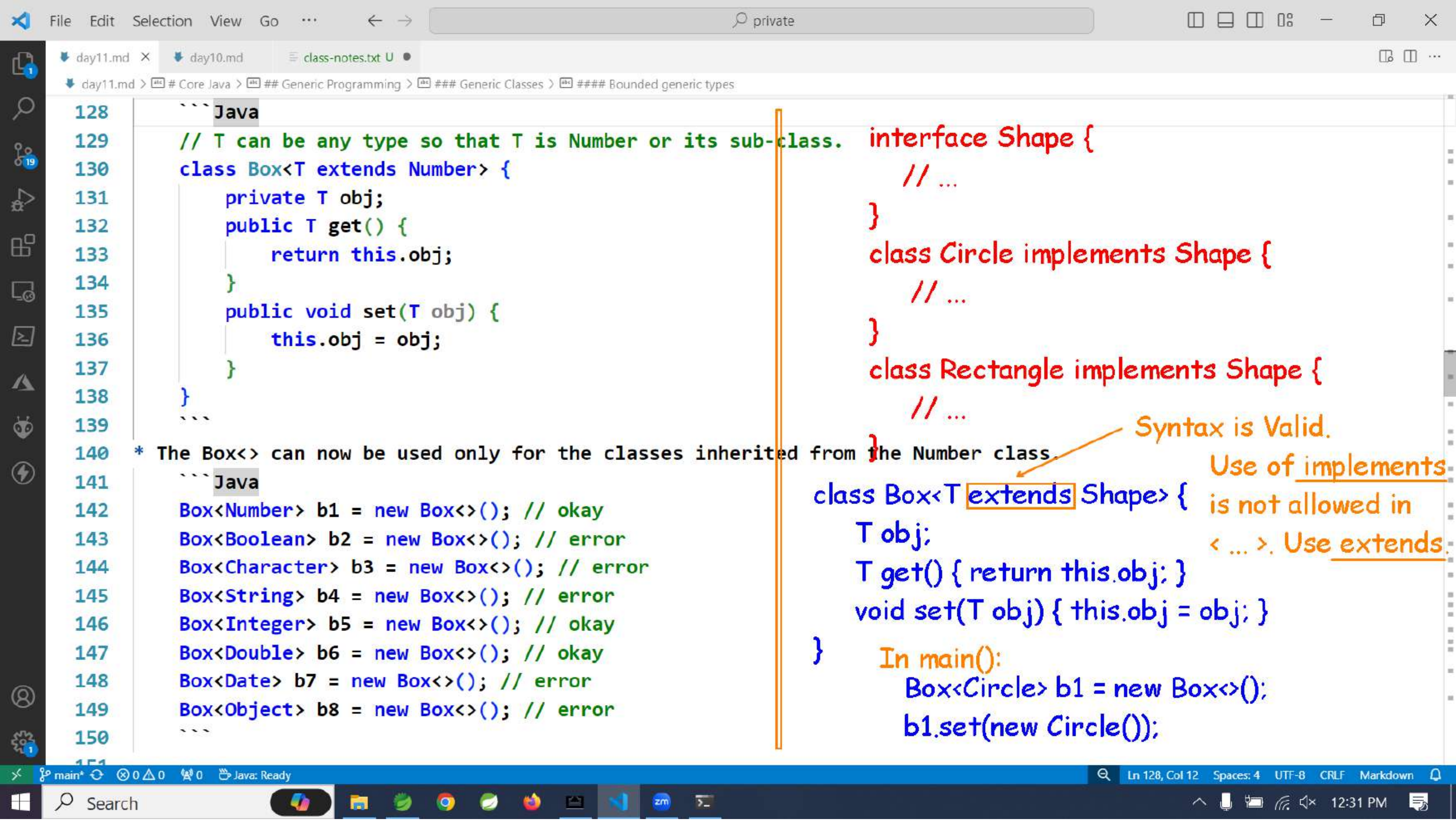
```
class Box {
    private Object obj;
    public void set(Object obj) {
        this.obj = obj;
    }
    public Object get() {
        return this.obj;
    }
}
```

```
main():
    Box<String> b1 = new Box<String>();
    b1.set("Hello");
    String r1 = b1.get();
    Box<Double> b2 = new Box<Double>();
    b2.set(3.14);
    Double r2 = b2.get();
```

The type-safety of Java generics is ensured by the compiler. JVM doesn't do any type-checking at runtime.

For JVM all references are like Object references.

There is no type info present in .class file. This called as Type Erasure.



File Edit Selection View Go ···
private
day11.md
day10.md
class-notes.txt U
day11.md > # Core Java > ## Generic Programming > ### Generic Classes > #### Bounded generic types
main* ⊗ 0 ⚠ 0 ₍ᵢ₎ 0 Java: Ready
Ln 128, Col 12 Spaces: 4 UTF-8 CRLF Markdown
Search
12:31 PM

~~~
    ```Java
    // T can be any type so that T is Number or its sub-class.
    class Box<T extends Number> {
        private T obj;
        public T get() {
            return this.obj;
        }
        public void set(T obj) {
            this.obj = obj;
        }
    }
    ```
* The Box<> can now be used only for the classes inherited from the Number class.
    ```Java
    Box<Number> b1 = new Box<>(); // okay
    Box<Boolean> b2 = new Box<>(); // error
    Box<Character> b3 = new Box<>(); // error
    Box<String> b4 = new Box<>(); // error
    Box<Integer> b5 = new Box<>(); // okay
    Box<Double> b6 = new Box<>(); // okay
    Box<Date> b7 = new Box<>(); // error
    Box<Object> b8 = new Box<>(); // error
    ```
~~~

```
interface Shape {
   // ...
}
class Circle implements Shape {
    // ...
}
class Rectangle implements Shape {
    // ...
}
```

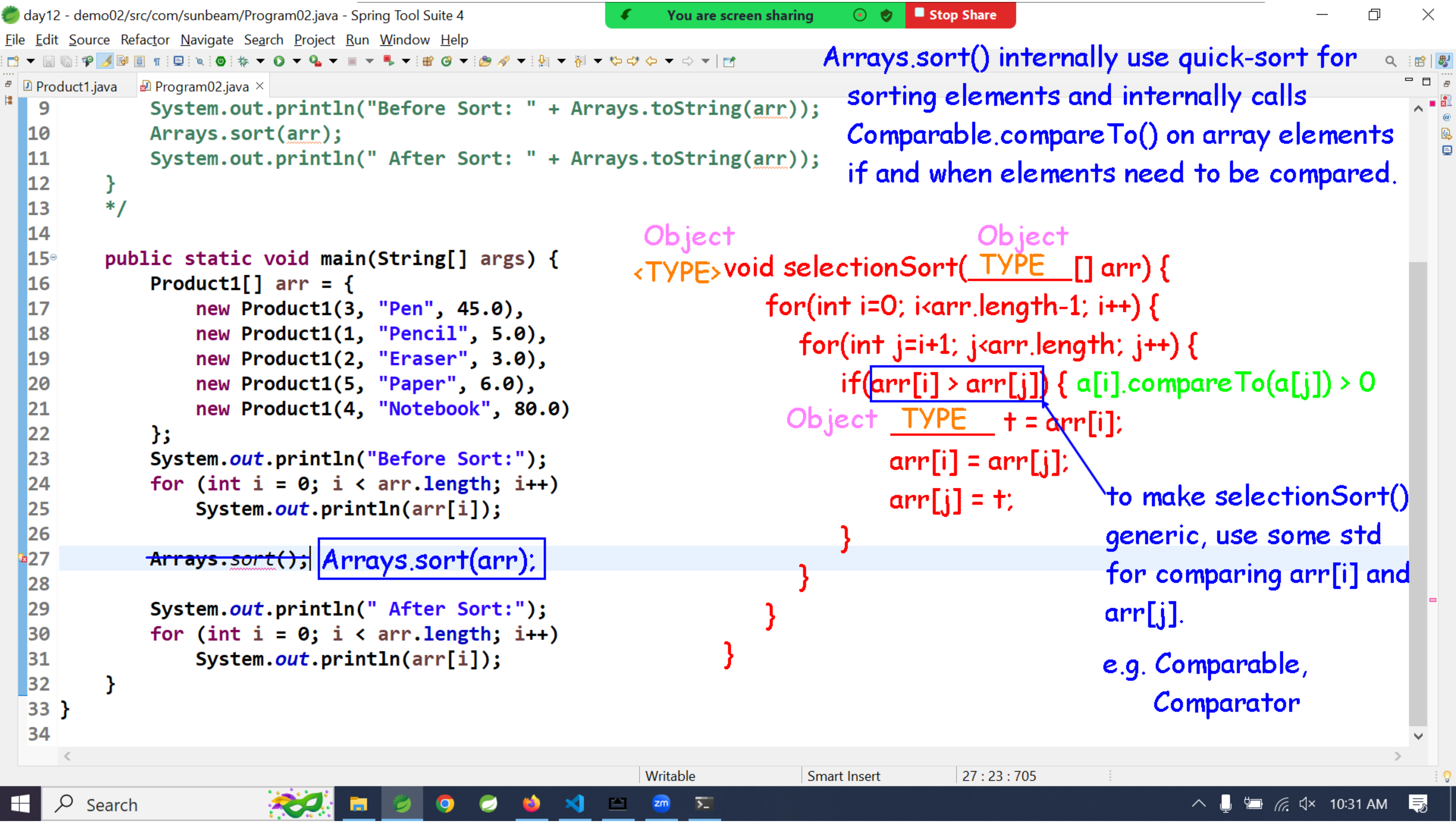
Syntax is Valid.

```
class Box<T extends Shape> {
    T obj;
    T get() { return this.obj; }
    void set(T obj) { this.obj = obj; }
}
```

Use of implements is not allowed in < ... >. Use extends.

In main():

```
Box<Circle> b1 = new Box<>();
b1.set(new Circle());
```


day12 - demo02/src/com/sunbeam/Program02.java - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Product1.java
Program02.java
Writable
Smart Insert
27 : 23 : 705
Search
10:31 AM

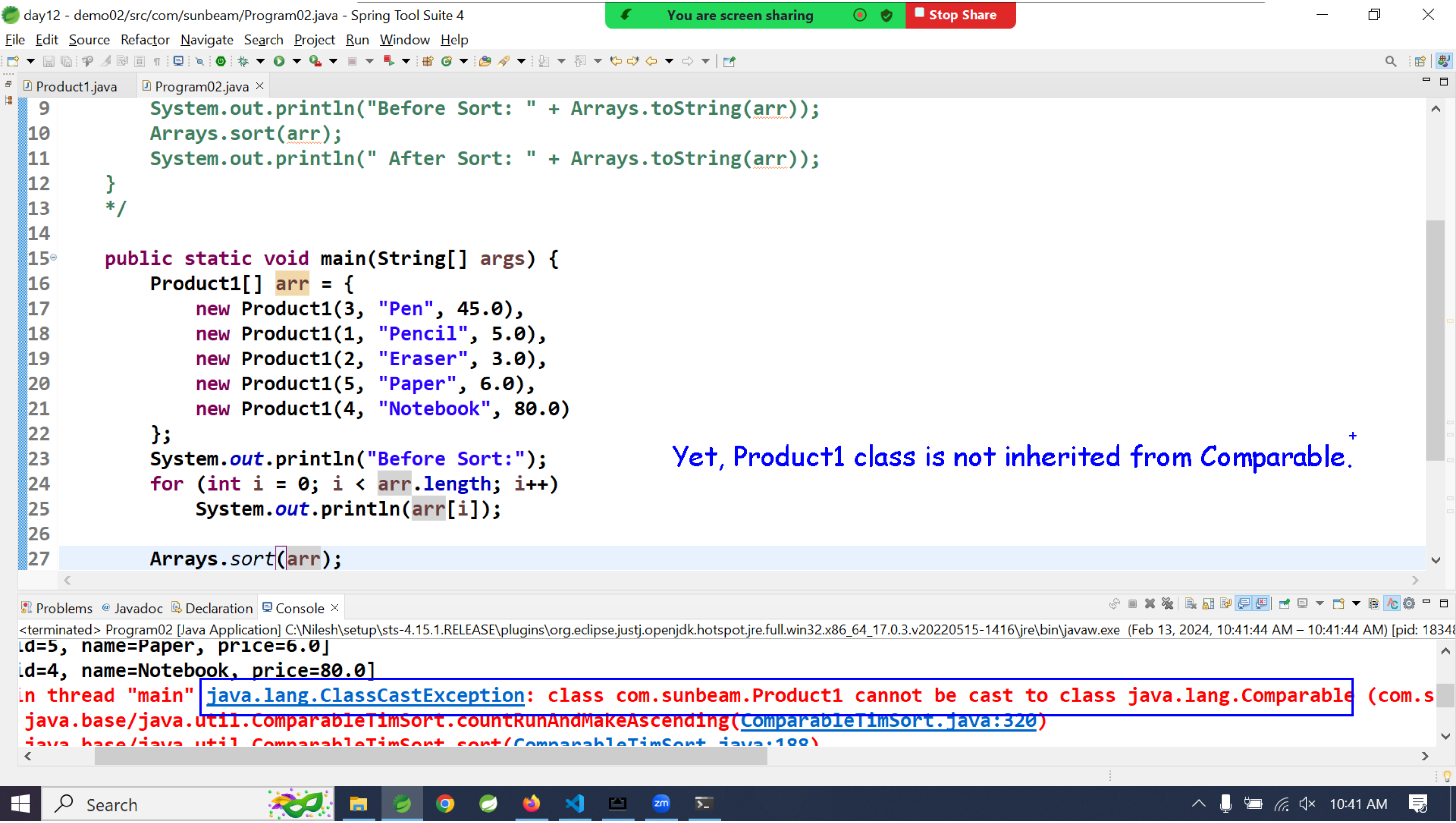
```java
        System.out.println("Before Sort: " + Arrays.toString(arr));
        Arrays.sort(arr);
        System.out.println(" After Sort: " + Arrays.toString(arr));
    }
    */

    public static void main(String[] args) {
        Product1[] arr = {
            new Product1(3, "Pen", 45.0),
            new Product1(1, "Pencil", 5.0),
            new Product1(2, "Eraser", 3.0),
            new Product1(5, "Paper", 6.0),
            new Product1(4, "Notebook", 80.0)
        };
        System.out.println("Before Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

        ~~Arrays.sort();~~ Arrays.sort(arr);

        System.out.println(" After Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);
    }
}
```

Arrays.sort() internally use quick-sort for sorting elements and internally calls Comparable.compareTo() on array elements if and when elements need to be compared.

```
Object                          Object
<TYPE>void selectionSort(TYPE[] arr) {
    for(int i=0; i<arr.length-1; i++) {
        for(int j=i+1; j<arr.length; j++) {
            if(arr[i] > arr[j]) { a[i].compareTo(a[j]) > 0
   Object       TYPE t = arr[i];
                arr[i] = arr[j];
                arr[j] = t;
            }
        }
    }
}
```

to make selectionSort() generic, use some std for comparing arr[i] and arr[j].

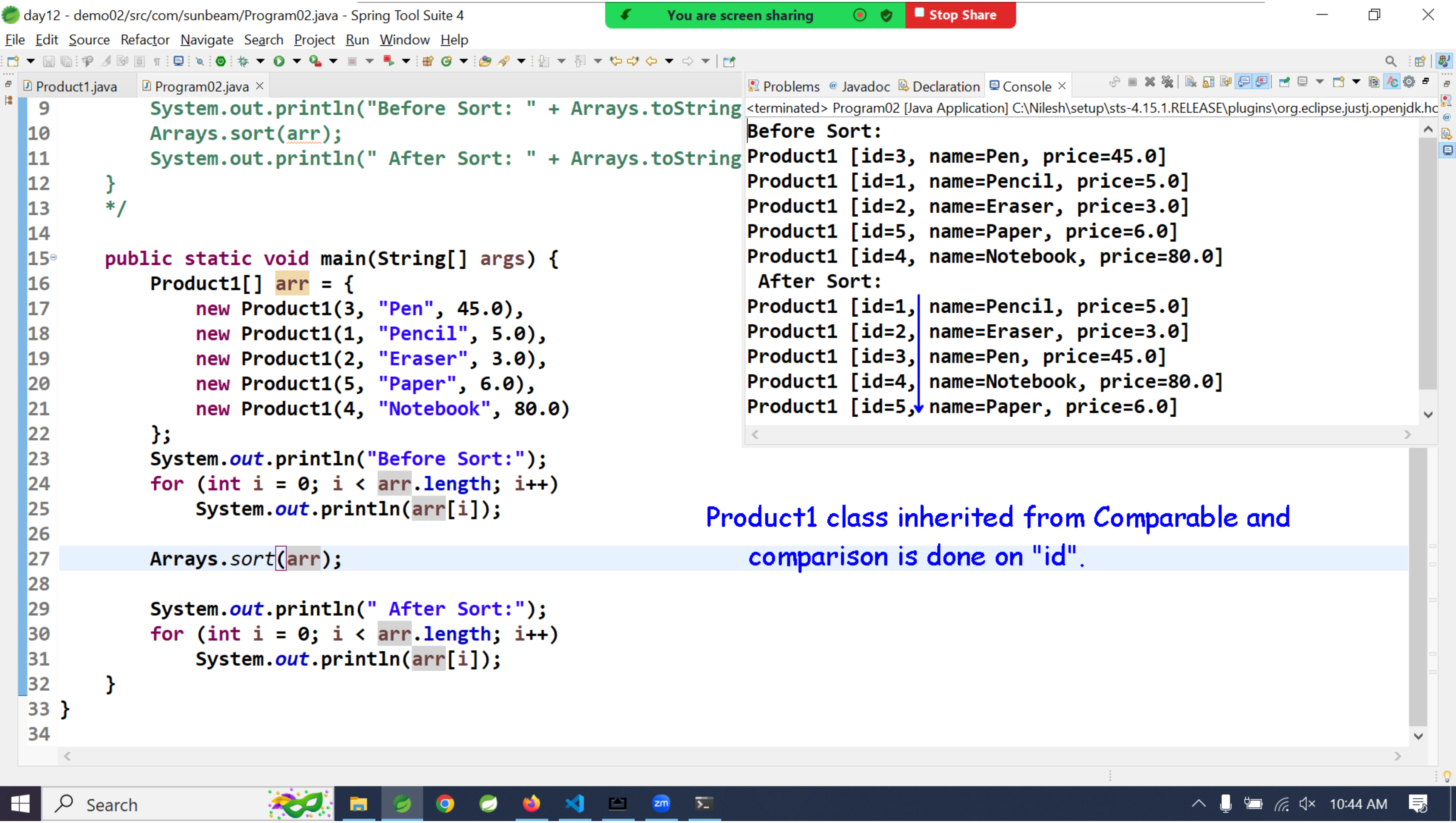
e.g. Comparable,
Comparator


day12 - demo02/src/com/sunbeam/Program02.java - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Product1.java
Program02.java
Problems Javadoc Declaration Console
Search
10:41 AM

```java
        System.out.println("Before Sort: " + Arrays.toString(arr));
        Arrays.sort(arr);
        System.out.println(" After Sort: " + Arrays.toString(arr));
    }
    */

    public static void main(String[] args) {
        Product1[] arr = {
            new Product1(3, "Pen", 45.0),
            new Product1(1, "Pencil", 5.0),
            new Product1(2, "Eraser", 3.0),
            new Product1(5, "Paper", 6.0),
            new Product1(4, "Notebook", 80.0)
        };
        System.out.println("Before Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

        Arrays.sort(arr);
```

Yet, Product1 class is not inherited from Comparable.

```
<terminated> Program02 [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 13, 2024, 10:41:44 AM – 10:41:44 AM) [pid: 1834
id=5, name=Paper, price=6.0]
id=4, name=Notebook, price=80.0]
in thread "main" java.lang.ClassCastException: class com.sunbeam.Product1 cannot be cast to class java.lang.Comparable (com.s
 java.base/java.util.ComparableTimSort.countRunAndMakeAscending(ComparableTimSort.java:320)
 java.base/java.util.ComparableTimSort.sort(ComparableTimSort.java:188)
```

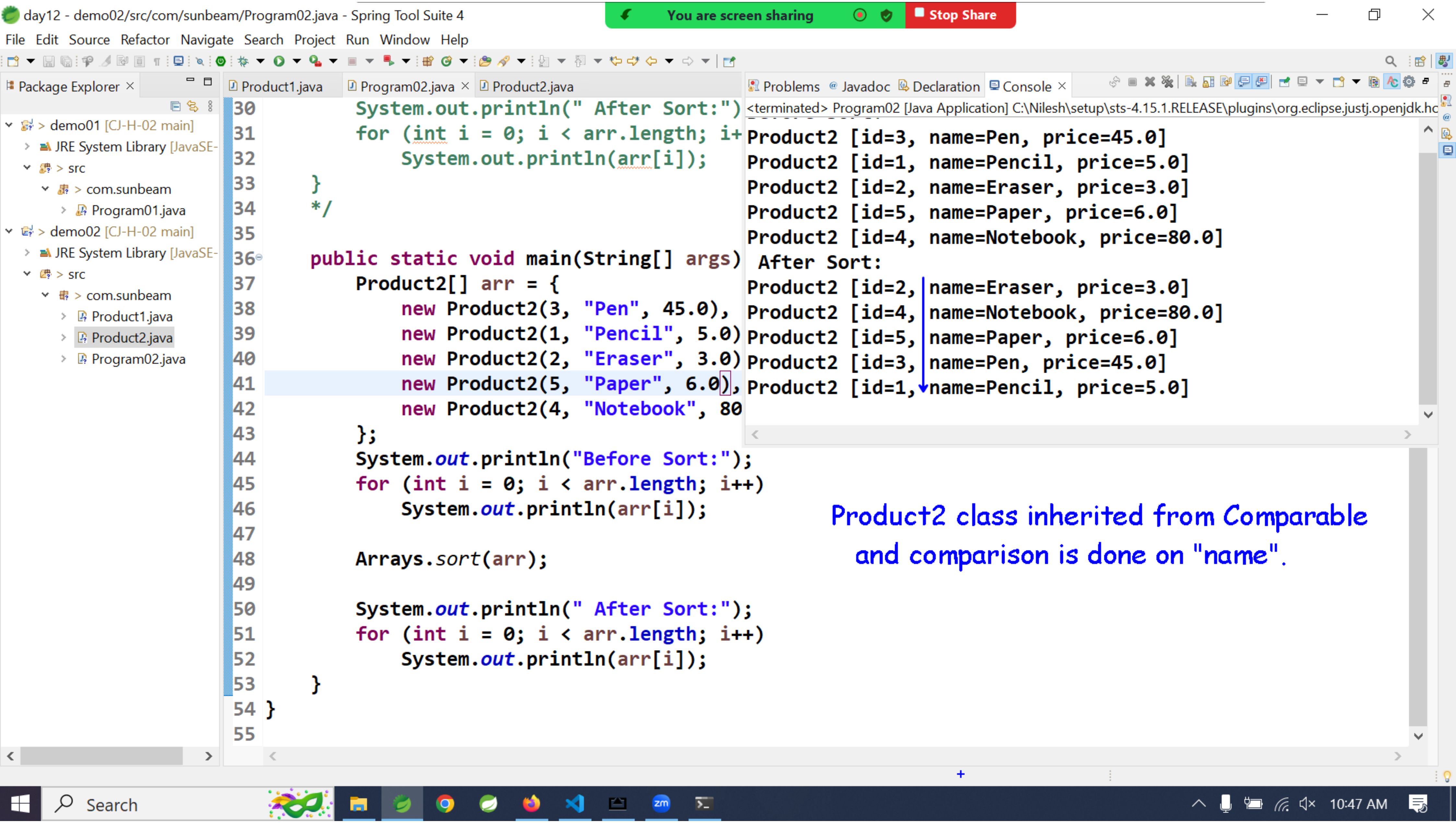


day12 - demo02/src/com/sunbeam/Program02.java - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Product1.java
Program02.java
Problems
Javadoc
Declaration
Console
Search
10:44 AM

```java
        System.out.println("Before Sort: " + Arrays.toString
        Arrays.sort(arr);
        System.out.println(" After Sort: " + Arrays.toString
    }
    */

    public static void main(String[] args) {
        Product1[] arr = {
            new Product1(3, "Pen", 45.0),
            new Product1(1, "Pencil", 5.0),
            new Product1(2, "Eraser", 3.0),
            new Product1(5, "Paper", 6.0),
            new Product1(4, "Notebook", 80.0)
        };
        System.out.println("Before Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

        Arrays.sort(arr);

        System.out.println(" After Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);
    }
}
```

```
<terminated> Program02 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hc
Before Sort:
Product1 [id=3, name=Pen, price=45.0]
Product1 [id=1, name=Pencil, price=5.0]
Product1 [id=2, name=Eraser, price=3.0]
Product1 [id=5, name=Paper, price=6.0]
Product1 [id=4, name=Notebook, price=80.0]
 After Sort:
Product1 [id=1, name=Pencil, price=5.0]
Product1 [id=2, name=Eraser, price=3.0]
Product1 [id=3, name=Pen, price=45.0]
Product1 [id=4, name=Notebook, price=80.0]
Product1 [id=5, name=Paper, price=6.0]
```

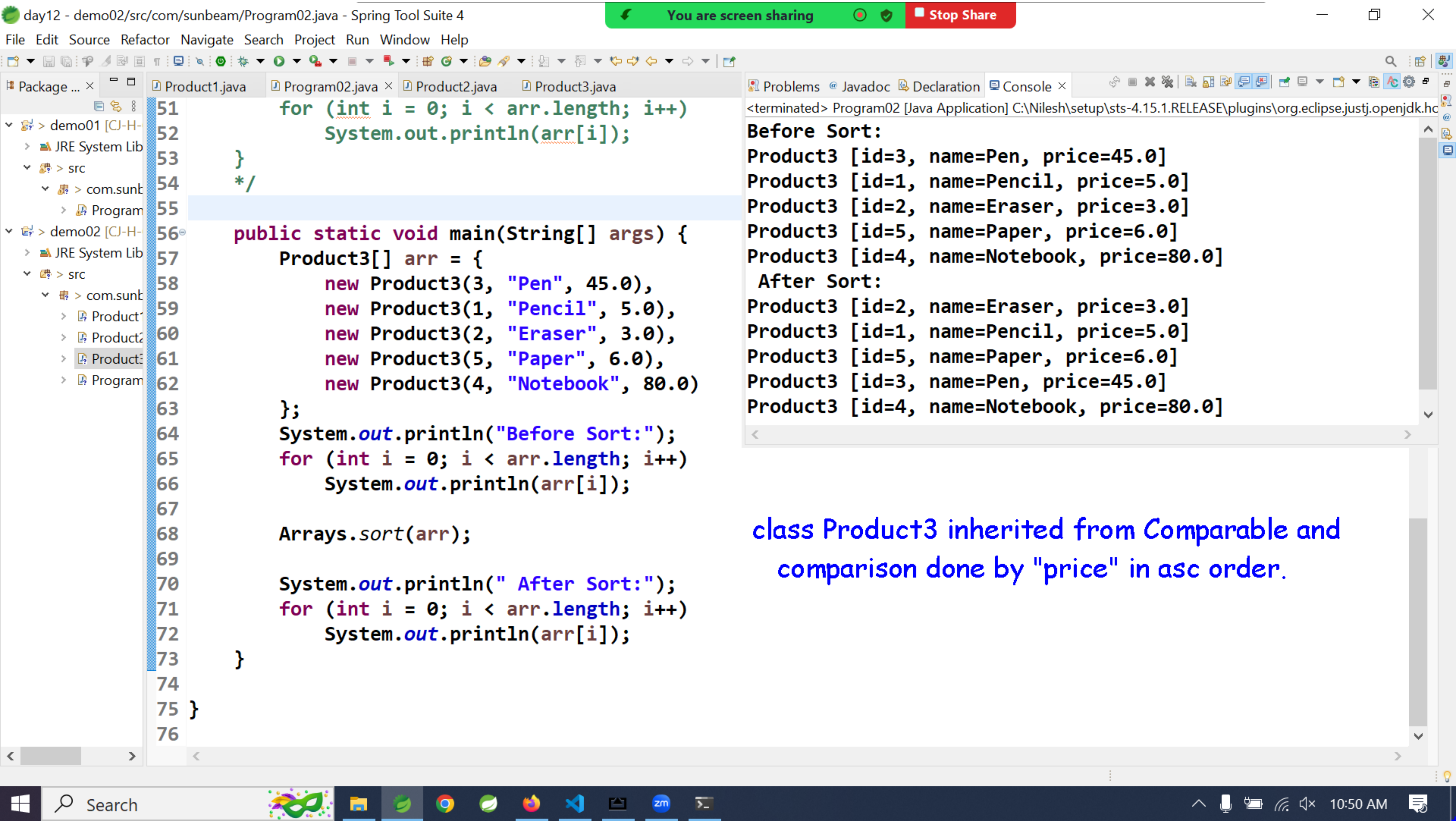
Product1 class inherited from Comparable and comparison is done on "id".



```
        System.out.println(" After Sort:")
        for (int i = 0; i < arr.length; i+
            System.out.println(arr[i]);
    }
    */

    public static void main(String[] args)
        Product2[] arr = {
            new Product2(3, "Pen", 45.0),
            new Product2(1, "Pencil", 5.0)
            new Product2(2, "Eraser", 3.0)
            new Product2(5, "Paper", 6.0),
            new Product2(4, "Notebook", 80
        };
        System.out.println("Before Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

        Arrays.sort(arr);

        System.out.println(" After Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);
    }
}
```

```
<terminated> Program02 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hc
Product2 [id=3, name=Pen, price=45.0]
Product2 [id=1, name=Pencil, price=5.0]
Product2 [id=2, name=Eraser, price=3.0]
Product2 [id=5, name=Paper, price=6.0]
Product2 [id=4, name=Notebook, price=80.0]
 After Sort:
Product2 [id=2, name=Eraser, price=3.0]
Product2 [id=4, name=Notebook, price=80.0]
Product2 [id=5, name=Paper, price=6.0]
Product2 [id=3, name=Pen, price=45.0]
Product2 [id=1, name=Pencil, price=5.0]
```

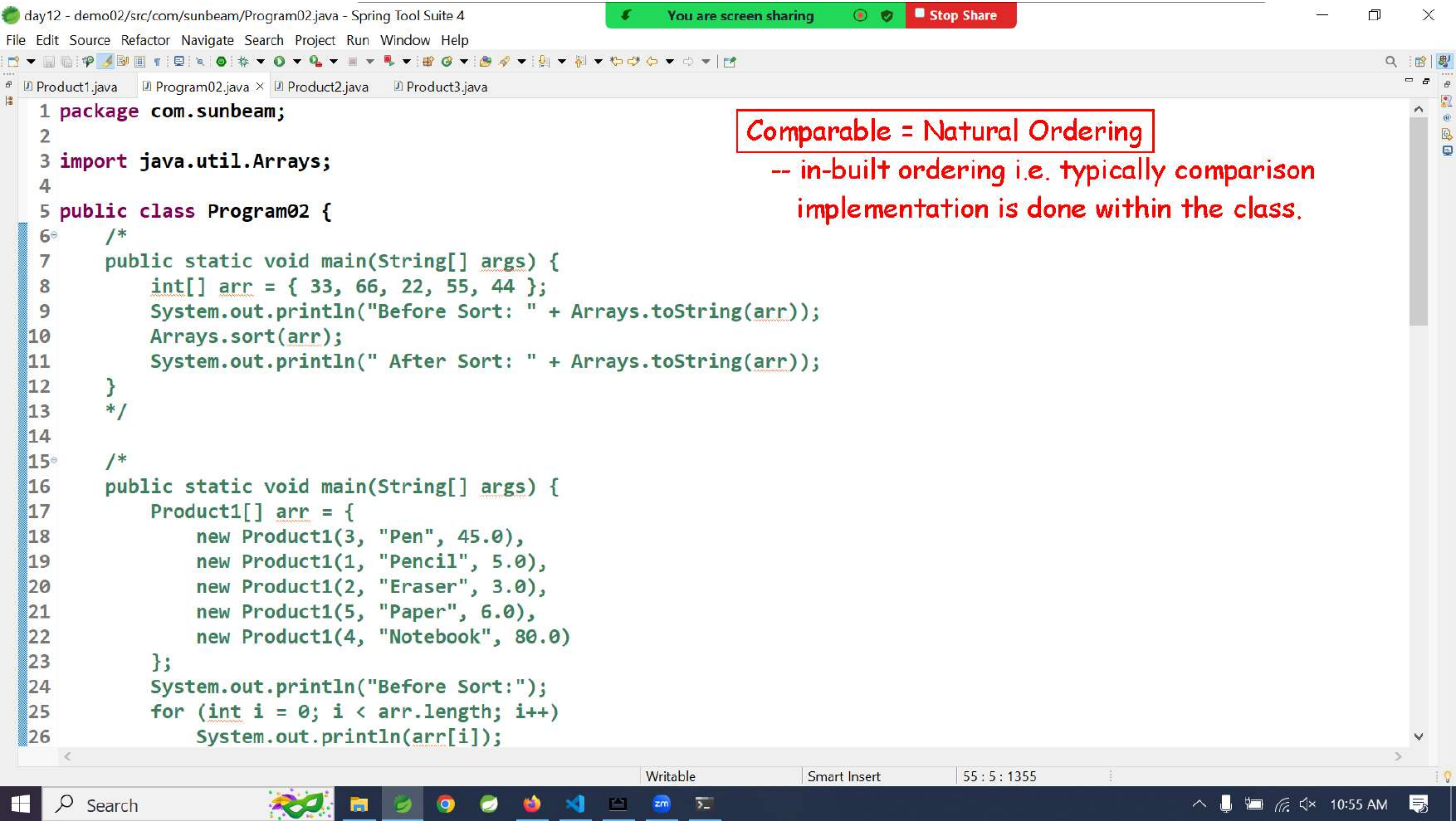
Product2 class inherited from Comparable and comparison is done on "name".



```java
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);
    }
    */

    public static void main(String[] args) {
        Product3[] arr = {
            new Product3(3, "Pen", 45.0),
            new Product3(1, "Pencil", 5.0),
            new Product3(2, "Eraser", 3.0),
            new Product3(5, "Paper", 6.0),
            new Product3(4, "Notebook", 80.0)
        };
        System.out.println("Before Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

        Arrays.sort(arr);

        System.out.println(" After Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);
    }

}
```

```
<terminated> Program02 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hc
Before Sort:
Product3 [id=3, name=Pen, price=45.0]
Product3 [id=1, name=Pencil, price=5.0]
Product3 [id=2, name=Eraser, price=3.0]
Product3 [id=5, name=Paper, price=6.0]
Product3 [id=4, name=Notebook, price=80.0]
 After Sort:
Product3 [id=2, name=Eraser, price=3.0]
Product3 [id=1, name=Pencil, price=5.0]
Product3 [id=5, name=Paper, price=6.0]
Product3 [id=3, name=Pen, price=45.0]
Product3 [id=4, name=Notebook, price=80.0]
```

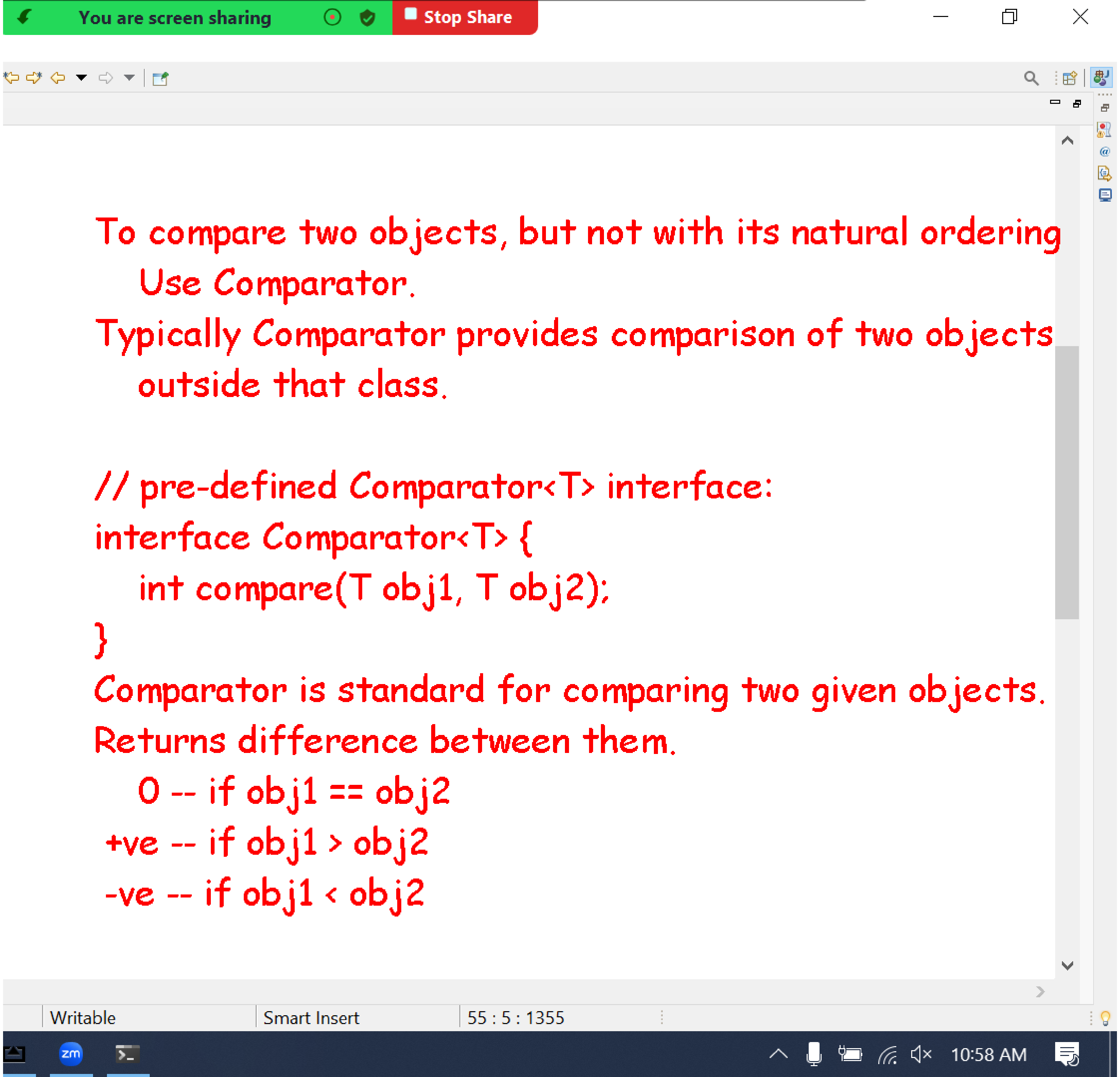
class Product3 inherited from Comparable and comparison done by "price" in asc order.


day12 - demo02/src/com/sunbeam/Program02.java - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Product1.java
Program02.java
Product2.java
Product3.java
Writable
Smart Insert
55 : 5 : 1355
Search
10:55 AM

Comparable = Natural Ordering

-- in-built ordering i.e. typically comparison implementation is done within the class.

```java
package com.sunbeam;

import java.util.Arrays;

public class Program02 {
    /*
    public static void main(String[] args) {
        int[] arr = { 33, 66, 22, 55, 44 };
        System.out.println("Before Sort: " + Arrays.toString(arr));
        Arrays.sort(arr);
        System.out.println(" After Sort: " + Arrays.toString(arr));
    }
    */

    /*
    public static void main(String[] args) {
        Product1[] arr = {
            new Product1(3, "Pen", 45.0),
            new Product1(1, "Pencil", 5.0),
            new Product1(2, "Eraser", 3.0),
            new Product1(5, "Paper", 6.0),
            new Product1(4, "Notebook", 80.0)
        };
        System.out.println("Before Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);
```

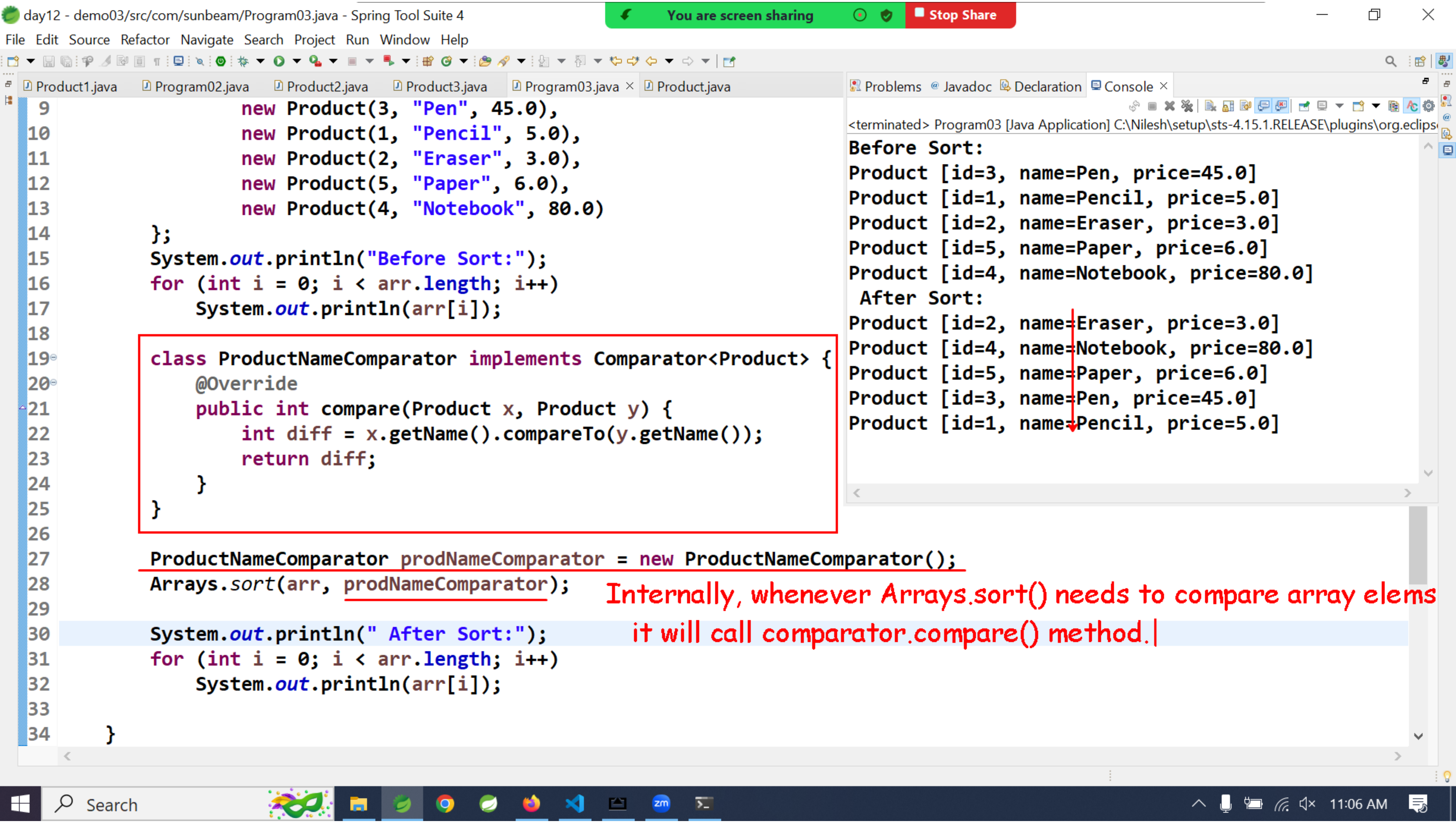


To compare two objects, but not with its natural ordering
    Use Comparator.
Typically Comparator provides comparison of two objects
    outside that class.

```
// pre-defined Comparator<T> interface:
interface Comparator<T> {
    int compare(T obj1, T obj2);
}
```

Comparator is standard for comparing two given objects.
Returns difference between them.

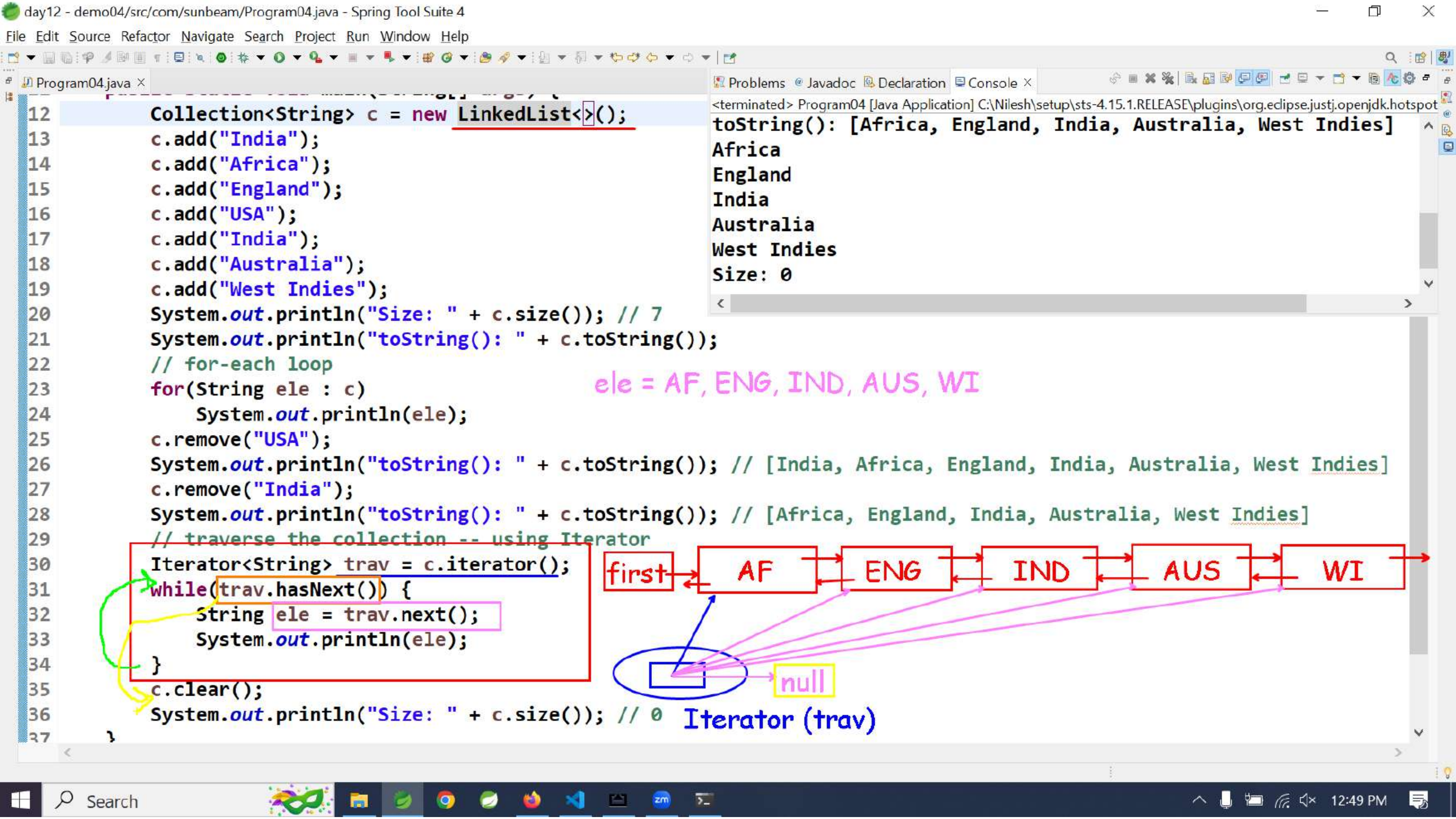
```
    0 -- if obj1 == obj2
 +ve -- if obj1 > obj2
 -ve -- if obj1 < obj2
```



```java
                new Product(3, "Pen", 45.0),
                new Product(1, "Pencil", 5.0),
                new Product(2, "Eraser", 3.0),
                new Product(5, "Paper", 6.0),
                new Product(4, "Notebook", 80.0)
        };
        System.out.println("Before Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

        class ProductNameComparator implements Comparator<Product> {
            @Override
            public int compare(Product x, Product y) {
                int diff = x.getName().compareTo(y.getName());
                return diff;
            }
        }

        ProductNameComparator prodNameComparator = new ProductNameComparator();
        Arrays.sort(arr, prodNameComparator);

        System.out.println(" After Sort:");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

    }
```

```
<terminated> Program03 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclips
Before Sort:
Product [id=3, name=Pen, price=45.0]
Product [id=1, name=Pencil, price=5.0]
Product [id=2, name=Eraser, price=3.0]
Product [id=5, name=Paper, price=6.0]
Product [id=4, name=Notebook, price=80.0]
 After Sort:
Product [id=2, name=Eraser, price=3.0]
Product [id=4, name=Notebook, price=80.0]
Product [id=5, name=Paper, price=6.0]
Product [id=3, name=Pen, price=45.0]
Product [id=1, name=Pencil, price=5.0]
```

Internally, whenever Arrays.sort() needs to compare array elems
it will call comparator.compare() method.

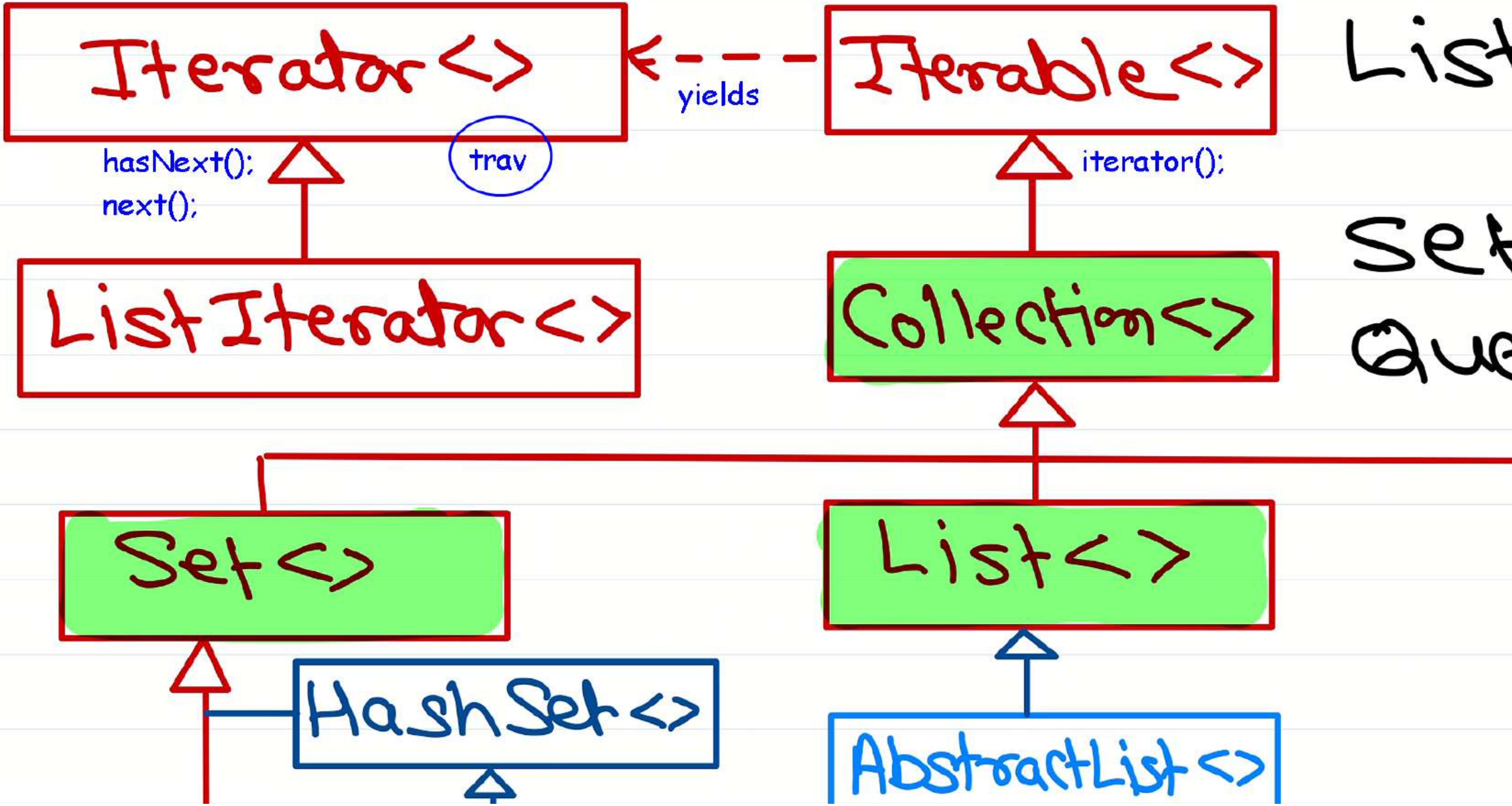


day12 - demo04/src/com/sunbeam/Program04.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program04.java
Problems
Javadoc
Declaration
Console
ele = AF, ENG, IND, AUS, WI
first
AF
ENG
IND
AUS
WI
null
Iterator (trav)
Search
12:49 PM

```
        Collection<String> c = new LinkedList<>();
        c.add("India");
        c.add("Africa");
        c.add("England");
        c.add("USA");
        c.add("India");
        c.add("Australia");
        c.add("West Indies");
        System.out.println("Size: " + c.size()); // 7
        System.out.println("toString(): " + c.toString());
        // for-each loop
        for(String ele : c)
            System.out.println(ele);
        c.remove("USA");
        System.out.println("toString(): " + c.toString()); // [India, Africa, England, India, Australia, West Indies]
        c.remove("India");
        System.out.println("toString(): " + c.toString()); // [Africa, England, India, Australia, West Indies]
        // traverse the collection -- using Iterator
        Iterator<String> trav = c.iterator();
        while(trav.hasNext()) {
            String ele = trav.next();
            System.out.println(ele);
        }
        c.clear();
        System.out.println("Size: " + c.size()); // 0
    }
```

```
<terminated> Program04 [Java Application] C:\Nilesh\setup\sts-4.15.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot
toString(): [Africa, England, India, Australia, West Indies]
Africa
England
India
Australia
West Indies
Size: 0
```

for-each loop works for any class inherited from Iterable.


Iterator<>
yields
Iterable<>
hasNext();
next();
trav
iterator();
ListIterator<>
Collection<>
Set<>
List<>
HashSet<>
AbstractList<>
List
Set
Que