Lesson 05 Demo 03

Differentiating Set and List

Objective: To demonstrate the difference between Set and List in Java, highlighting their unique features and usage scenarios for managing email addresses, focusing on handling duplicates and maintaining order

Tools Required: Eclipse IDE

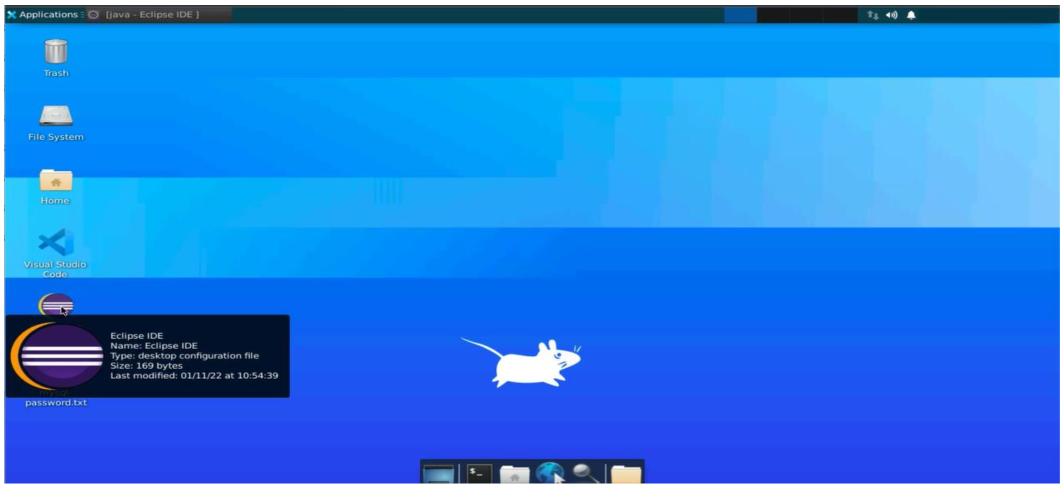
Prerequisites: None

Steps to be followed:

1. Create a new project
2. Create an ArrayList of type String to store email addresses
3. Handle duplicate records using the ArrayList
4. Execute the code with example data
5. Implement data storage using HashSet
6. Clear the emails using the clear method
7. Iterate over the email addresses using an iterator
8. Implement the LinkedHashSet and execute the code with example data

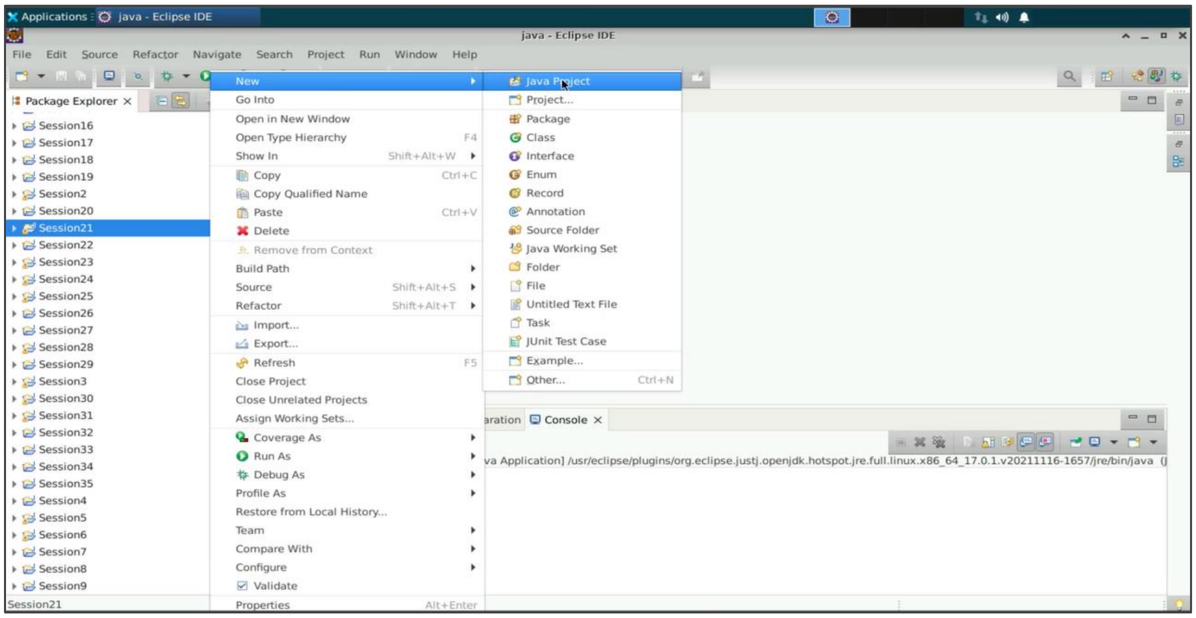
Step 1: Create a new project

1.1 Open the **Eclipse IDE**

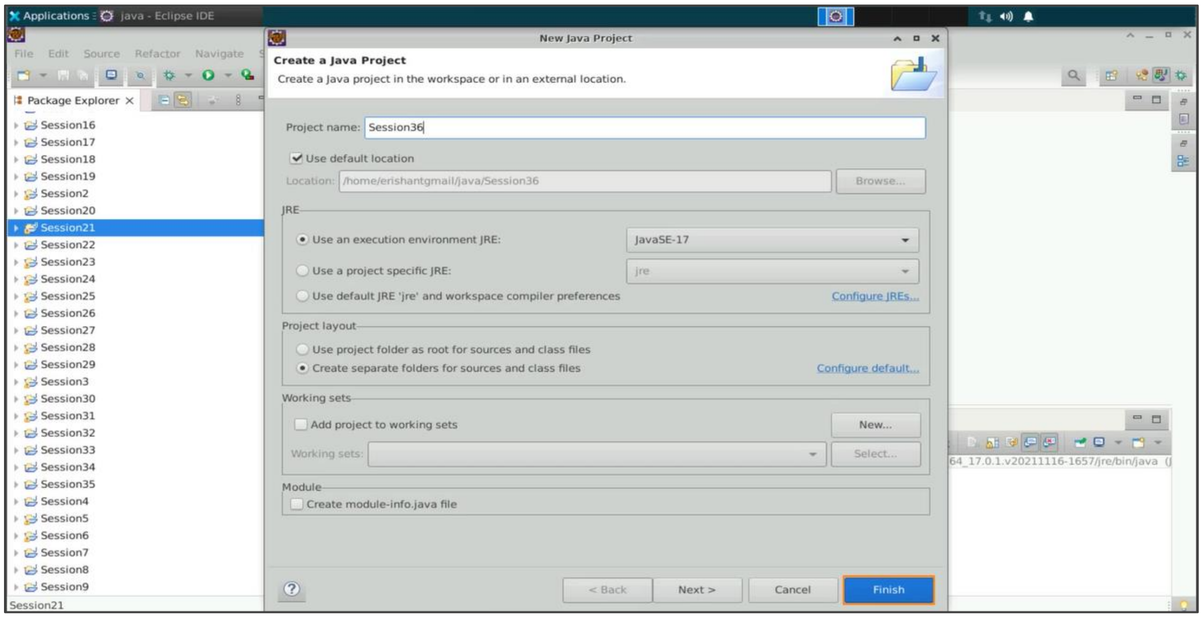


Eclipse IDE
Name: Eclipse IDE
Type: desktop configuration file
Size: 169 bytes
Last modified: 01/11/22 at 10:54:39

1.2 Create a new Java project named **Session37**



```
va Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java
```



Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: Session36

[✓] Use default location

Location: /home/erishantgmail/java/Session36

JRE
- (●) Use an execution environment JRE: JavaSE-17
- ( ) Use a project specific JRE: jre
- ( ) Use default JRE 'jre' and workspace compiler preferences

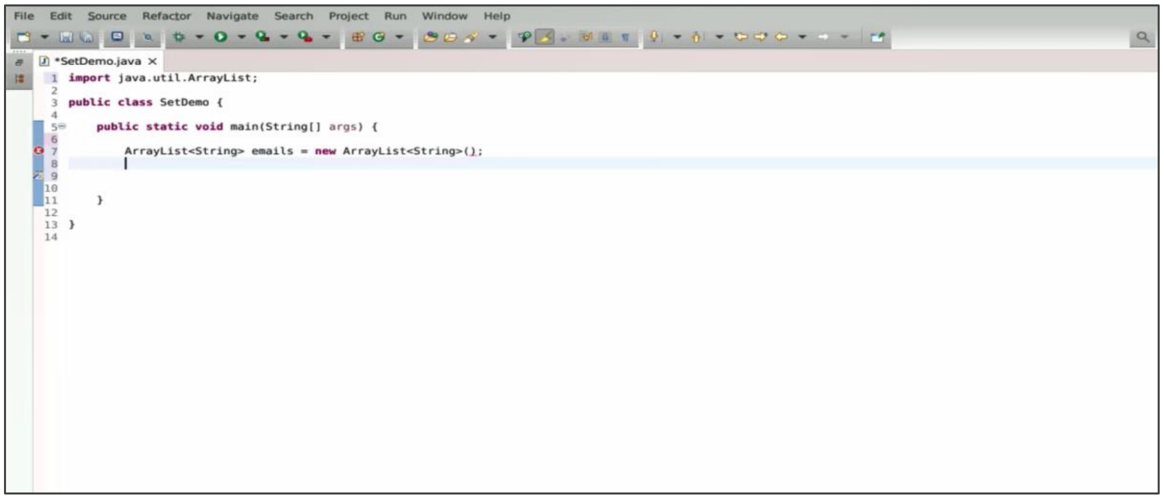
Project layout
- ( ) Use project folder as root for sources and class files
- (●) Create separate folders for sources and class files

Working sets
- [ ] Add project to working sets
- Working sets:

Module
- [ ] Create module-info.java file

Step 2: Create an ArrayList of type String and add email addresses

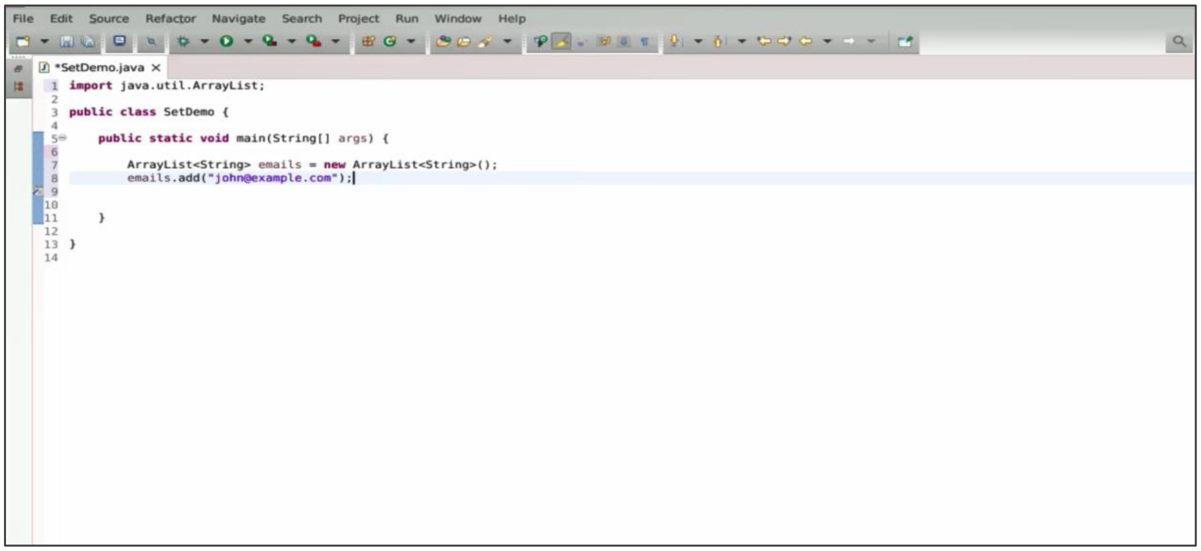
2.1 Create an ArrayList of type String to store email addresses



File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

```java
import java.util.ArrayList;

public class SetDemo {

    public static void main(String[] args) {

        ArrayList<String> emails = new ArrayList<String>();


    }

}
```

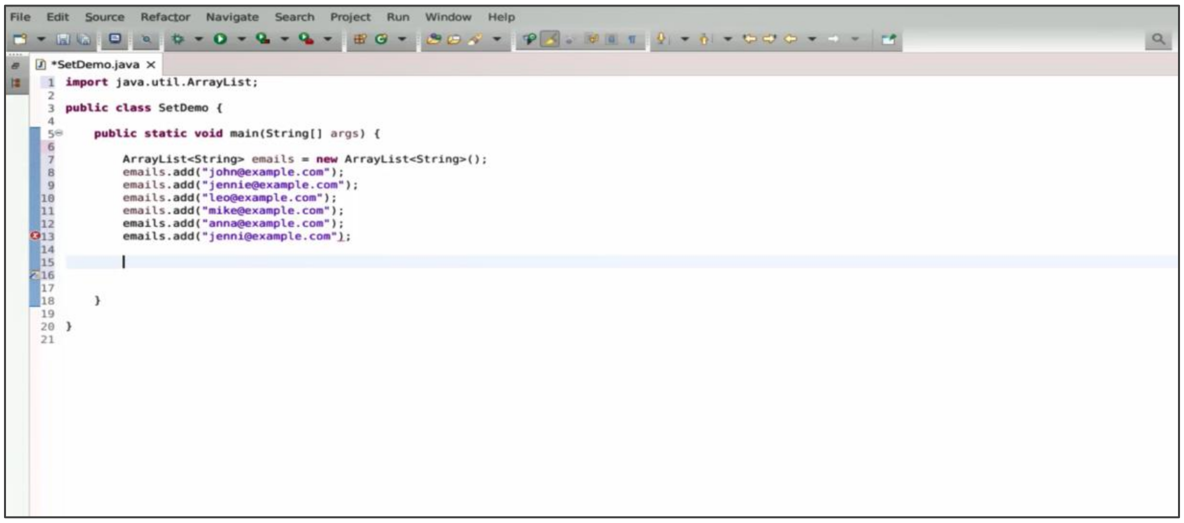
2.2 Add the email **john@example.com** to the list



File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

```java
import java.util.ArrayList;

public class SetDemo {

    public static void main(String[] args) {

        ArrayList<String> emails = new ArrayList<String>();
        emails.add("john@example.com");


    }
}
```

2.3 Similarly, add **jennie@example.com**, **leo@example.com**, and **mike@example.com** to the list

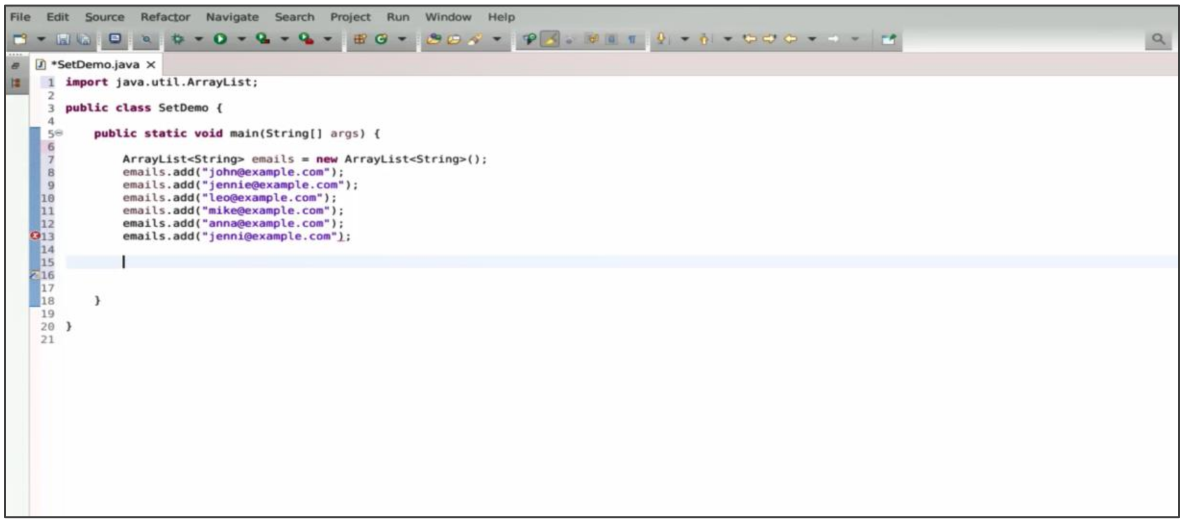

File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

```java
import java.util.ArrayList;

public class SetDemo {

    public static void main(String[] args) {

        ArrayList<String> emails = new ArrayList<String>();
        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jenni@example.com");



    }

}
```

Note: ArrayList allows duplicate records to be stored.

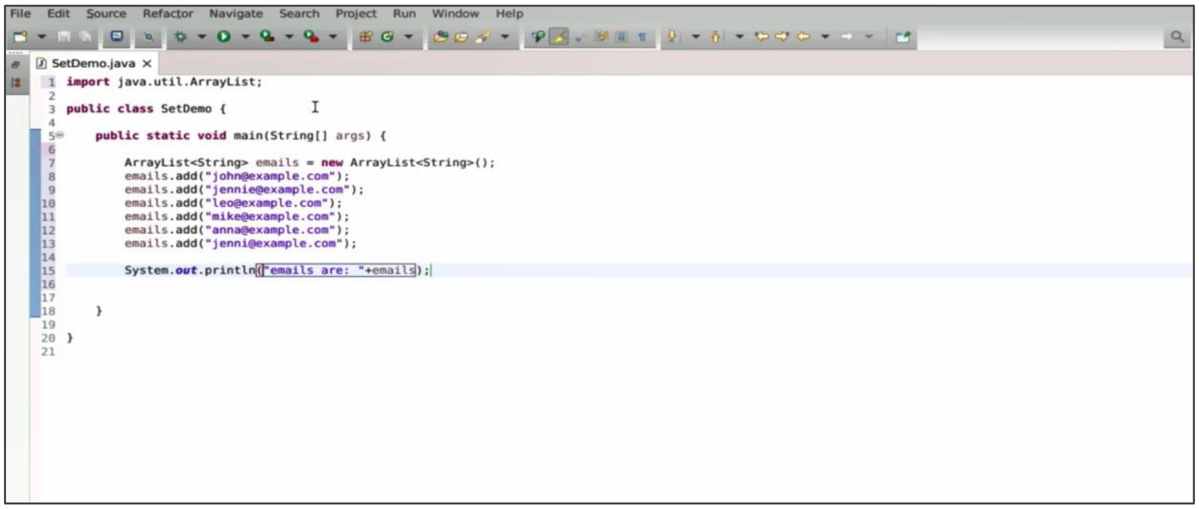
Step 3: Handle duplicate records with ArrayList

3.1 Notice that **jennie@example.com** is added twice, indicating that ArrayList allows duplicate records


File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

```java
import java.util.ArrayList;

public class SetDemo {

    public static void main(String[] args) {

        ArrayList<String> emails = new ArrayList<String>();
        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jenni@example.com");



    }

}
```

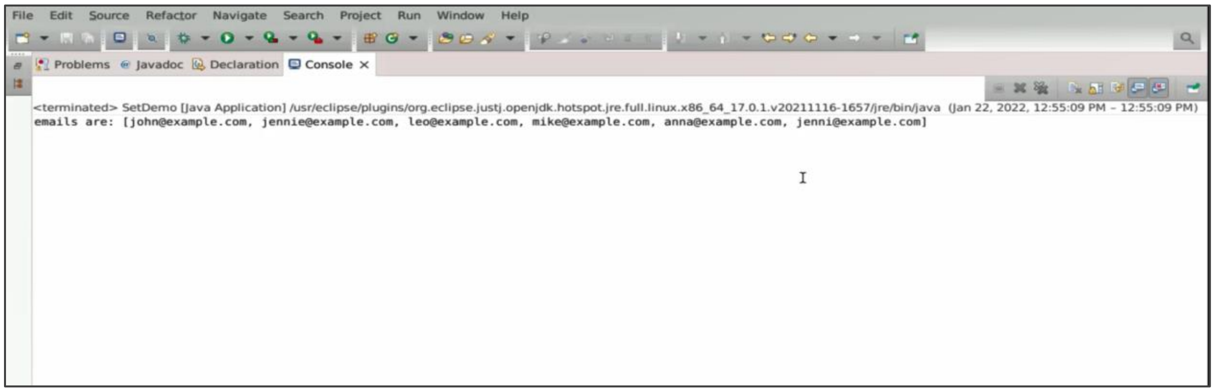
3.2 Print the emails using the **println** function: “**emails are: “+ emails**



```java
import java.util.ArrayList;

public class SetDemo {

    public static void main(String[] args) {

        ArrayList<String> emails = new ArrayList<String>();
        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jenni@example.com");

        System.out.println("emails are: "+emails);


    }

}
```

Step 4: Execute the code with example data

4.1 Run the code and observe the output



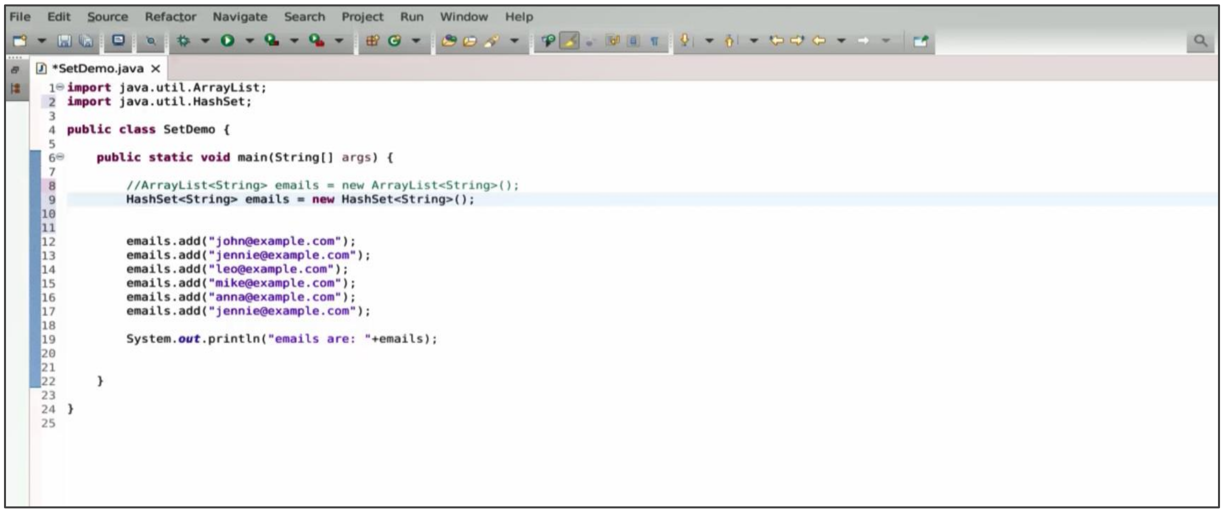
```
<terminated> SetDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Jan 22, 2022, 12:55:09 PM – 12:55:09 PM)
emails are: [john@example.com, jennie@example.com, leo@example.com, mike@example.com, anna@example.com, jenni@example.com]
```

You will see **jennie@example.com** listed twice.

Note: ArrayList maintains the order of insertion.

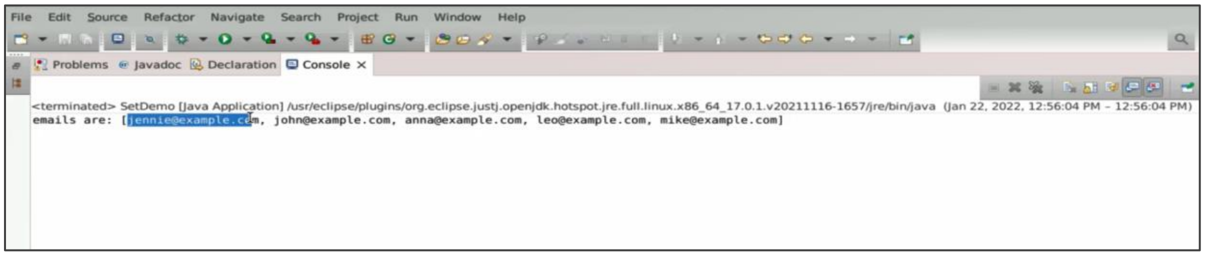
Step 5: Implement HashSet for unique data storage using hashing

5.1 Create a HashSet of type String named **emails** to store email addresses using hashing


File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

```java
import java.util.ArrayList;
import java.util.HashSet;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        HashSet<String> emails = new HashSet<String>();


        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);


    }

}
```

5.2 Run the program and observe that there are no duplicate email addresses present in the HashSet

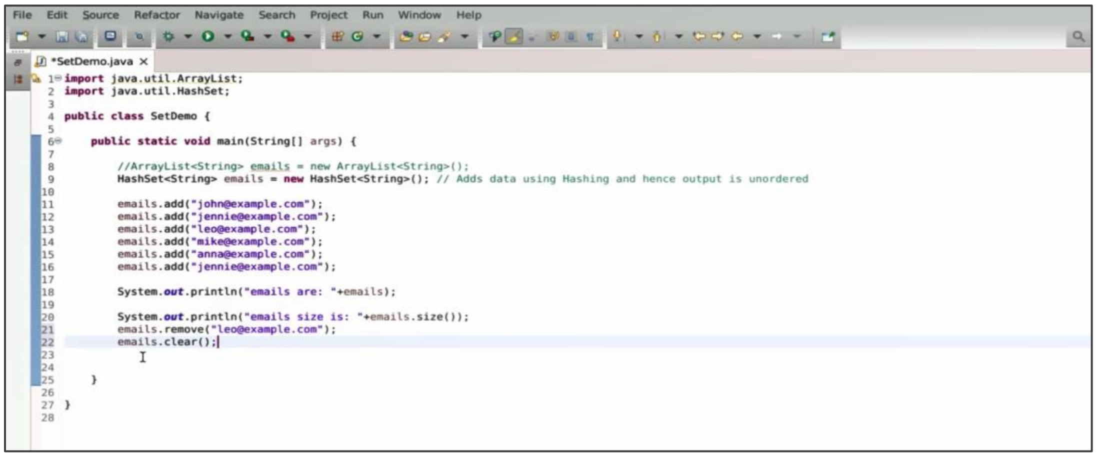

File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console

```
<terminated> SetDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Jan 22, 2022, 12:56:04 PM – 12:56:04 PM)
emails are: [jennie@example.com, john@example.com, anna@example.com, leo@example.com, mike@example.com]
```

Note: HashSet ensures uniqueness and does not maintain insertion order.

Step 6: Clear the emails using the clear method

6.1 Use the **clear** method to remove all email addresses from the HashSet

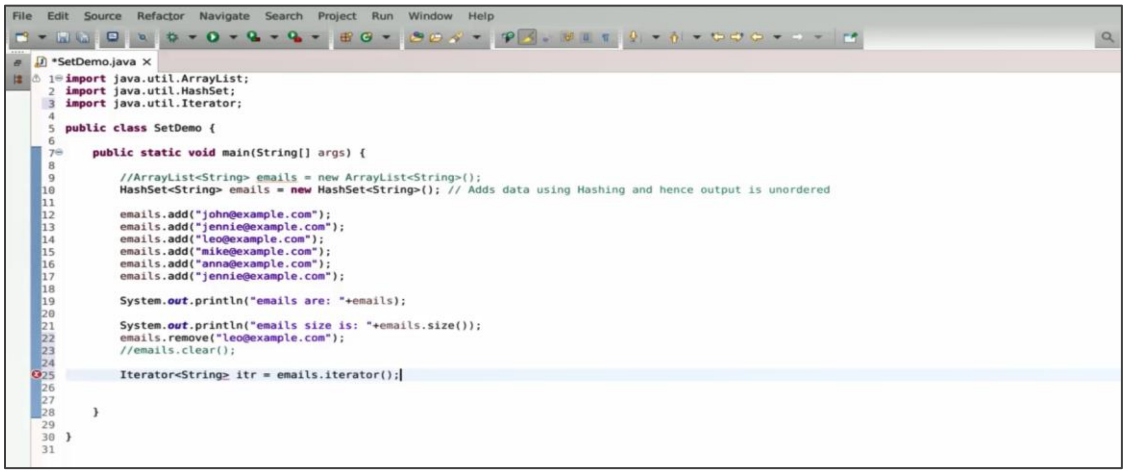

File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

```java
import java.util.ArrayList;
import java.util.HashSet;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hence output is unordered

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        emails.remove("leo@example.com");
        emails.clear();


    }

}
```

Note: HashSet does not support indexing, so individual records cannot be accessed.

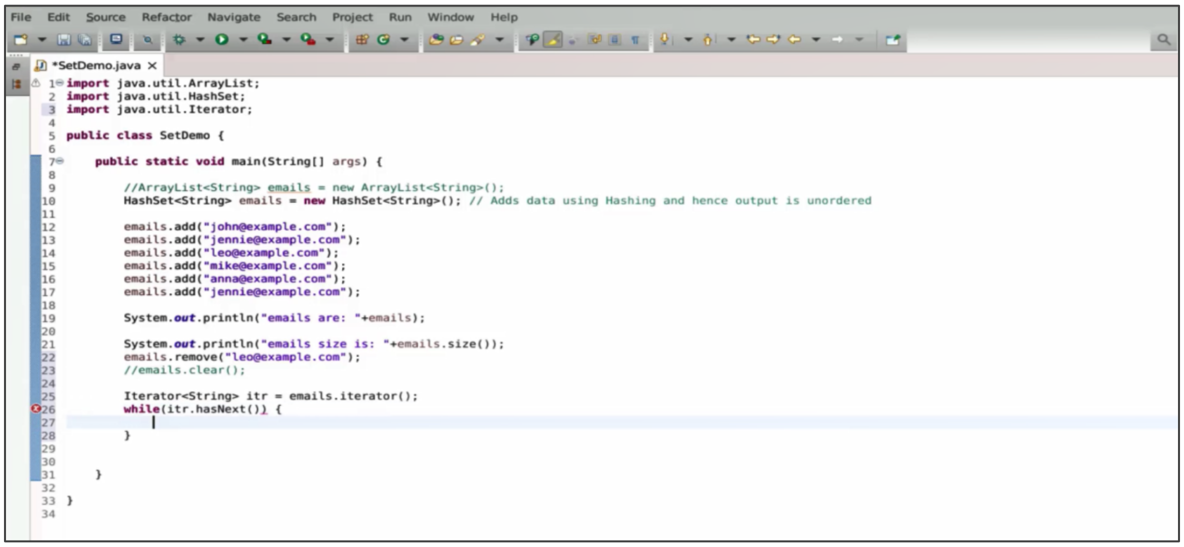
Step 7: Iterate over the email addresses using the iterator API

7.1 To iterate over the HashSet, use the iterator API. Declare an iterator of type String named **itr** and initialize it with **emails.iterator()**.


File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

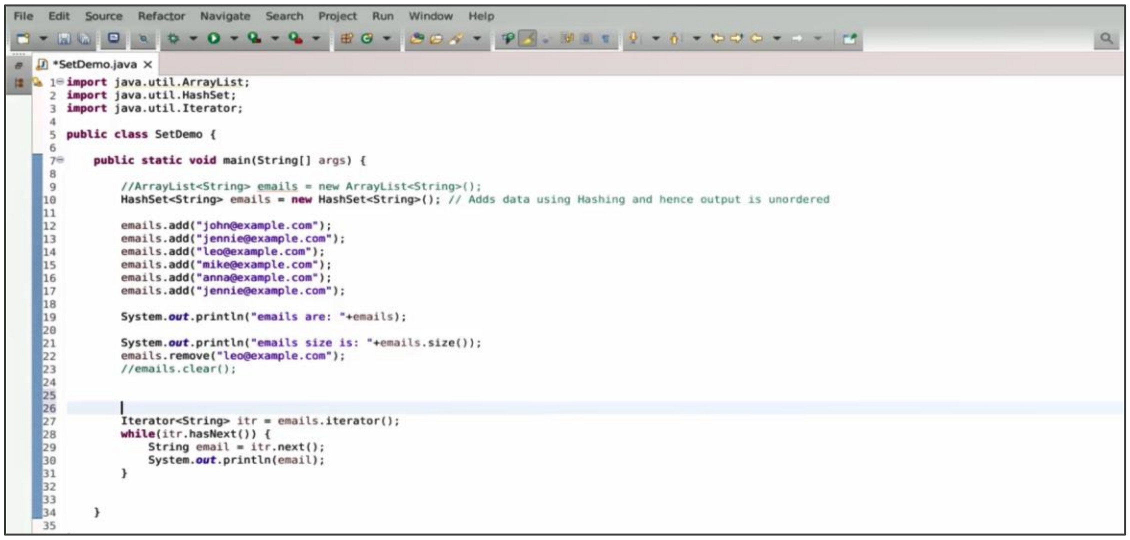
```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hence output is unordered

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        emails.remove("leo@example.com");
        //emails.clear();

        Iterator<String> itr = emails.iterator();


    }

}
```

7.2 Use a while loop with the condition **itr.hasNext()**


File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

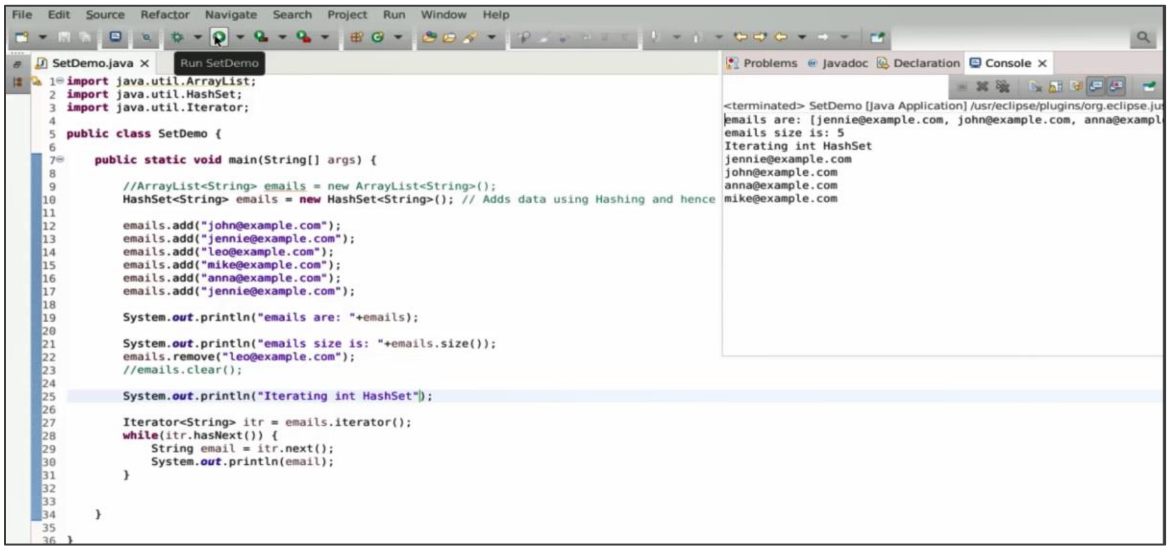
```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hence output is unordered

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        emails.remove("leo@example.com");
        //emails.clear();

        Iterator<String> itr = emails.iterator();
        while(itr.hasNext()) {
            
        }

    }

}
```

7.3 Within the loop, retrieve the next email using **String email = itr.next()** and print it


File Edit Source Refactor Navigate Search Project Run Window Help
*SetDemo.java

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hence output is unordered

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        emails.remove("leo@example.com");
        //emails.clear();


        Iterator<String> itr = emails.iterator();
        while(itr.hasNext()) {
            String email = itr.next();
            System.out.println(email);
        }


    }
```

7.4 Run the code and observe that you can iterate over the HashSet

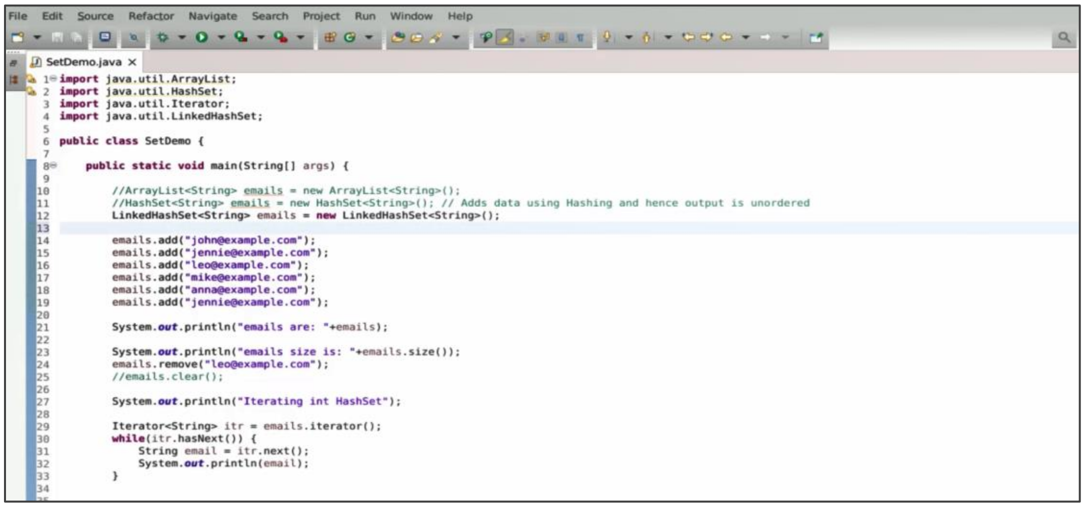

File Edit Source Refactor Navigate Search Project Run Window Help
SetDemo.java
Run SetDemo
Problems
Javadoc
Declaration
Console

```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hence

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        emails.remove("leo@example.com");
        //emails.clear();

        System.out.println("Iterating int HashSet");

        Iterator<String> itr = emails.iterator();
        while(itr.hasNext()) {
            String email = itr.next();
            System.out.println(email);
        }


    }

}
```

```
<terminated> SetDemo [Java Application] /usr/eclipse/plugins/org.eclipse.ju
emails are: [jennie@example.com, john@example.com, anna@exampl
emails size is: 5
Iterating int HashSet
jennie@example.com
john@example.com
anna@example.com
mike@example.com
```

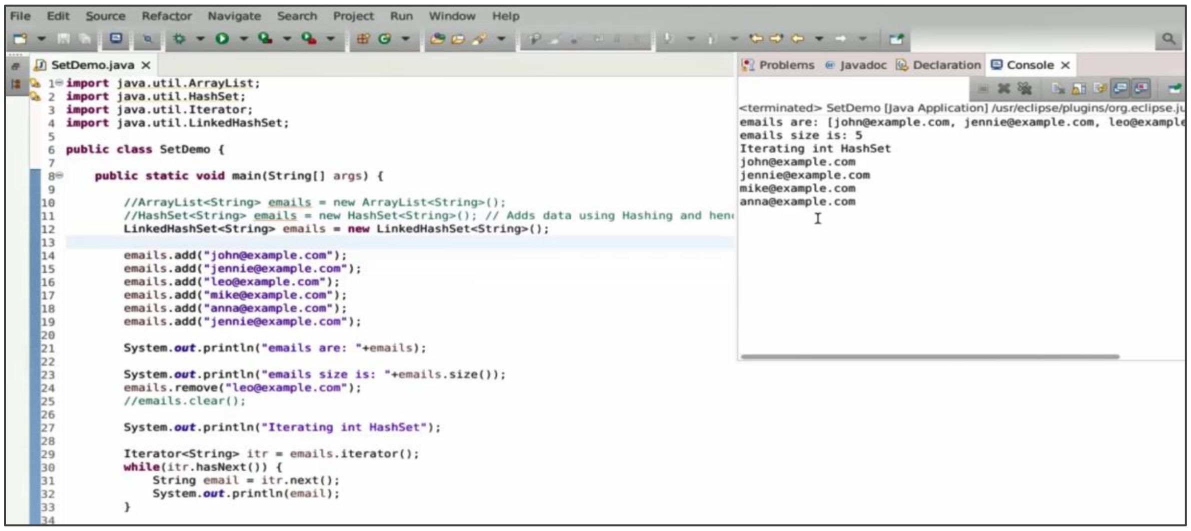
Step 8: Implement LinkedHashSet to maintain insertion order

8.1 If you want to maintain the order of insertion, use **LinkedHashSet** instead of HashSet


File Edit Source Refactor Navigate Search Project Run Window Help
SetDemo.java

```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashSet;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        //HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hence output is unordered
        LinkedHashSet<String> emails = new LinkedHashSet<String>();

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        emails.remove("leo@example.com");
        //emails.clear();

        System.out.println("Iterating int HashSet");

        Iterator<String> itr = emails.iterator();
        while(itr.hasNext()) {
            String email = itr.next();
            System.out.println(email);
        }
```

8.2 Run the code and observe that the email addresses are displayed in the same order as they were added

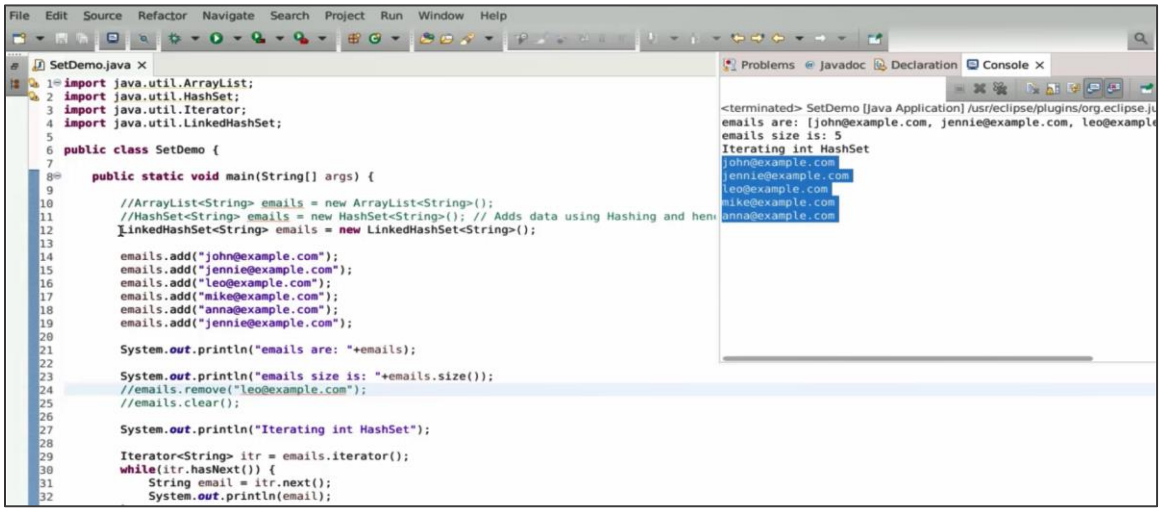


```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashSet;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        //HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hen
        LinkedHashSet<String> emails = new LinkedHashSet<String>();

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        emails.remove("leo@example.com");
        //emails.clear();

        System.out.println("Iterating int HashSet");

        Iterator<String> itr = emails.iterator();
        while(itr.hasNext()) {
            String email = itr.next();
            System.out.println(email);
        }
```

```
<terminated> SetDemo [Java Application] /usr/eclipse/plugins/org.eclipse.ju
emails are: [john@example.com, jennie@example.com, leo@example
emails size is: 5
Iterating int HashSet
john@example.com
jennie@example.com
mike@example.com
anna@example.com
```

Note: LinkedHashSet maintains both uniqueness and insertion order.

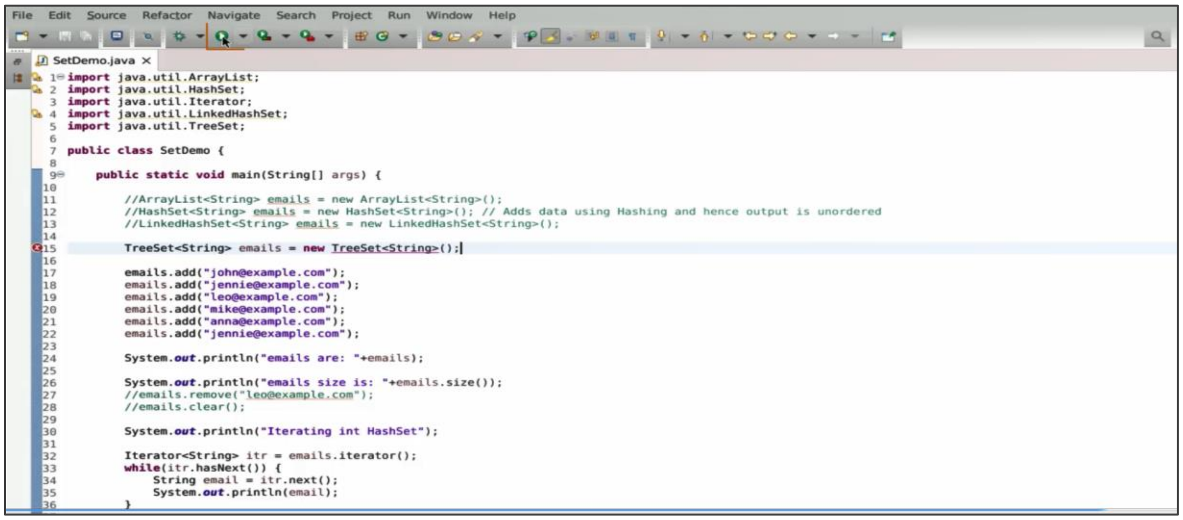
8.3 Comment out the **emails.remove** part if present. The data will still be displayed in the same order as it was added.



```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashSet;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        //HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hen
        LinkedHashSet<String> emails = new LinkedHashSet<String>();

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        //emails.remove("leo@example.com");
        //emails.clear();

        System.out.println("Iterating int HashSet");

        Iterator<String> itr = emails.iterator();
        while(itr.hasNext()) {
            String email = itr.next();
            System.out.println(email);
```

```
<terminated> SetDemo [Java Application] /usr/eclipse/plugins/org.eclipse.ju
emails are: [john@example.com, jennie@example.com, leo@example
emails size is: 5
Iterating int HashSet
john@example.com
jennie@example.com
leo@example.com
mike@example.com
anna@example.com
```

8.4 Additionally, you can use TreeSet to sort the data. TreeSet ensures uniqueness and provides a sorted arrangement.


File Edit Source Refactor Navigate Search Project Run Window Help
SetDemo.java

```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.TreeSet;

public class SetDemo {

    public static void main(String[] args) {

        //ArrayList<String> emails = new ArrayList<String>();
        //HashSet<String> emails = new HashSet<String>(); // Adds data using Hashing and hence output is unordered
        //LinkedHashSet<String> emails = new LinkedHashSet<String>();

        TreeSet<String> emails = new TreeSet<String>();

        emails.add("john@example.com");
        emails.add("jennie@example.com");
        emails.add("leo@example.com");
        emails.add("mike@example.com");
        emails.add("anna@example.com");
        emails.add("jennie@example.com");

        System.out.println("emails are: "+emails);

        System.out.println("emails size is: "+emails.size());
        //emails.remove("leo@example.com");
        //emails.clear();

        System.out.println("Iterating int HashSet");

        Iterator<String> itr = emails.iterator();
        while(itr.hasNext()) {
            String email = itr.next();
            System.out.println(email);
        }
```

Note: TreeSet is suitable for sorting data, but it requires elements to be either integers or strings. For custom sorting, Comparable and Comparator interfaces need to be implemented.