**CST-239 Activity 5**

**Sept 14 ,2023**

**Taofik Otiotio**

Part 1: Java Generics

**Theory of Operation**

### **Overview:**

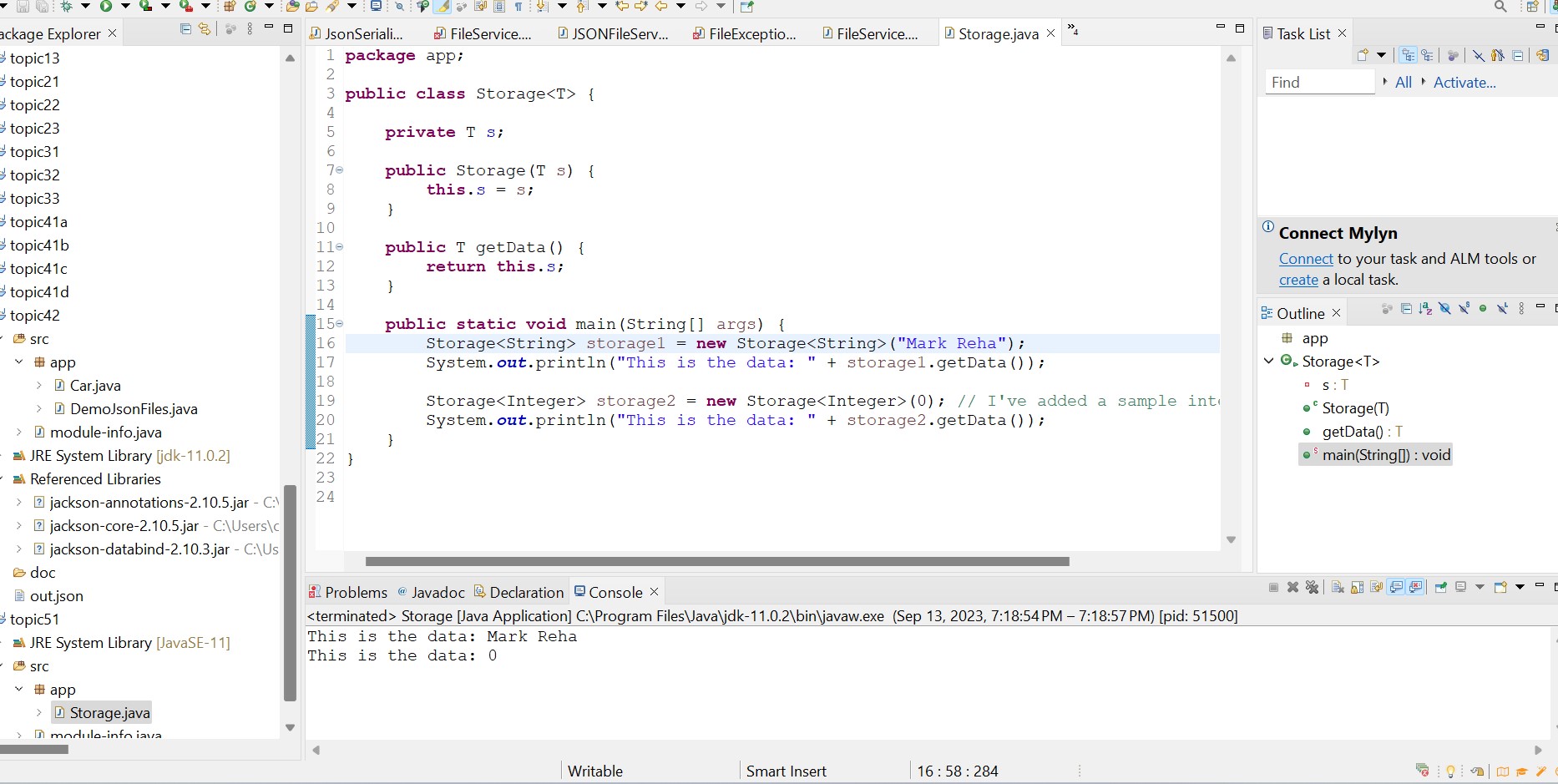
The provided code demonstrates the implementation and usage of generic classes and methods in Java. Generics are a powerful feature in Java, which allows the creation of classes, interfaces, and methods that operate on parameterized types. The primary purpose of generics is to ensure type safety and to allow operations on objects of various types while retaining the benefits of strong type checking at compile-time.

**1. Storage Class:**

**Purpose**:  
This class offers a generic way to store and retrieve a single item of a specified type.

**Key Features**:  
a) **Generic Type T**: Represents the type of item to be stored. It allows the class to hold any type of data (e.g., String, Integer, etc.).  
b) **Constructor**: Accepts an item of type T and initializes the internal storage.  
c) **getData Method**: Retrieves the stored item of type T.

**Sample Usage**:  
In the main method, two instances of Storage class are created: One for storing a String named "Mark Reha" and another for storing an Integer with a value of 0.

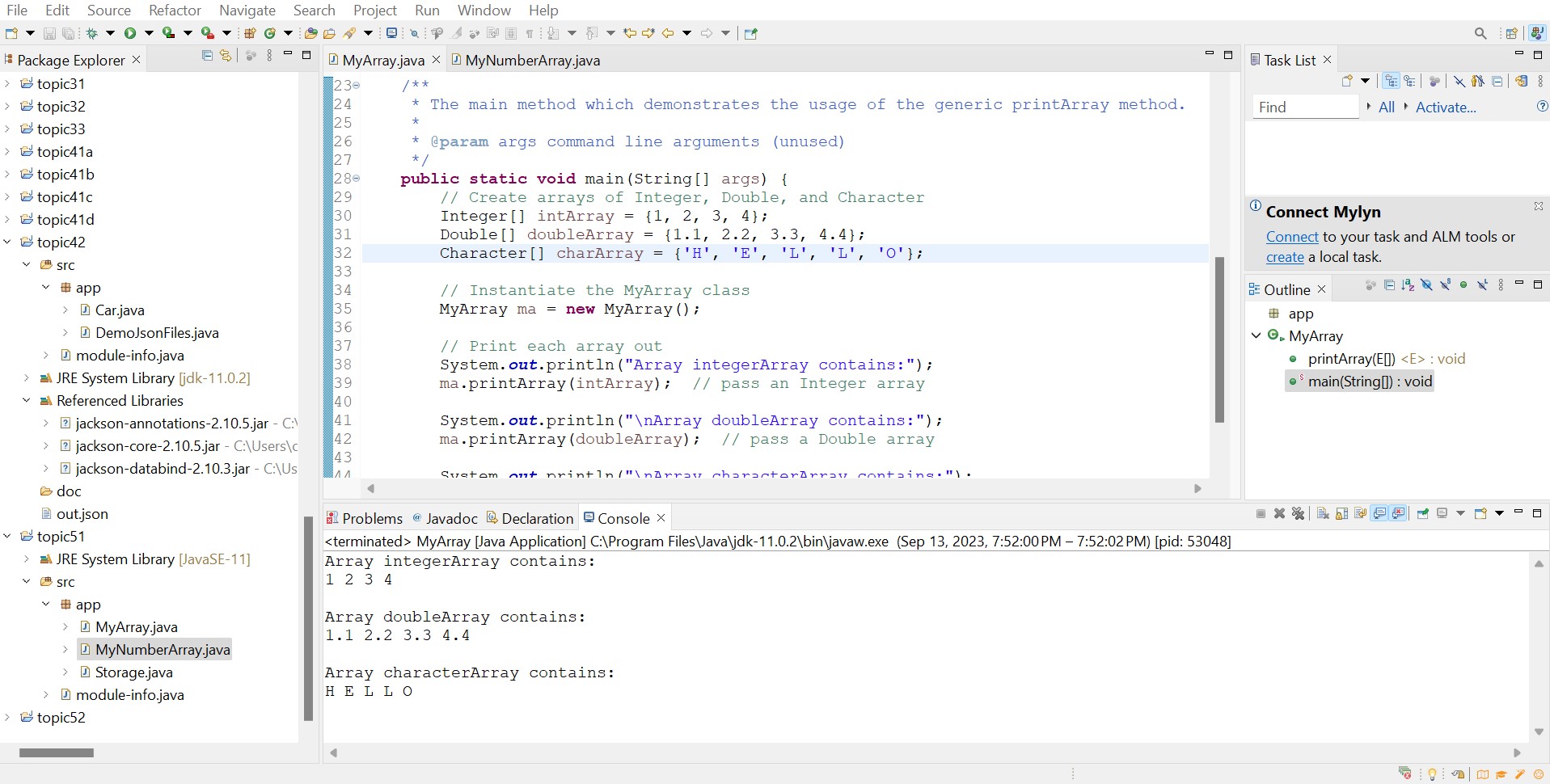


**2. MyArray Class:**

**Purpose**:  
This class demonstrates the use of a generic method that prints elements of any array, regardless of its element type.

**Key Features**:  
a) **Generic Method printArray**: Accepts an array of any type and prints its elements. The type of the array's elements is denoted by <E>.

**Sample Usage**:  
In the main method, three arrays of different types (Integer, Double, and Character) are created. The printArray method is then invoked for each array, displaying their elements.

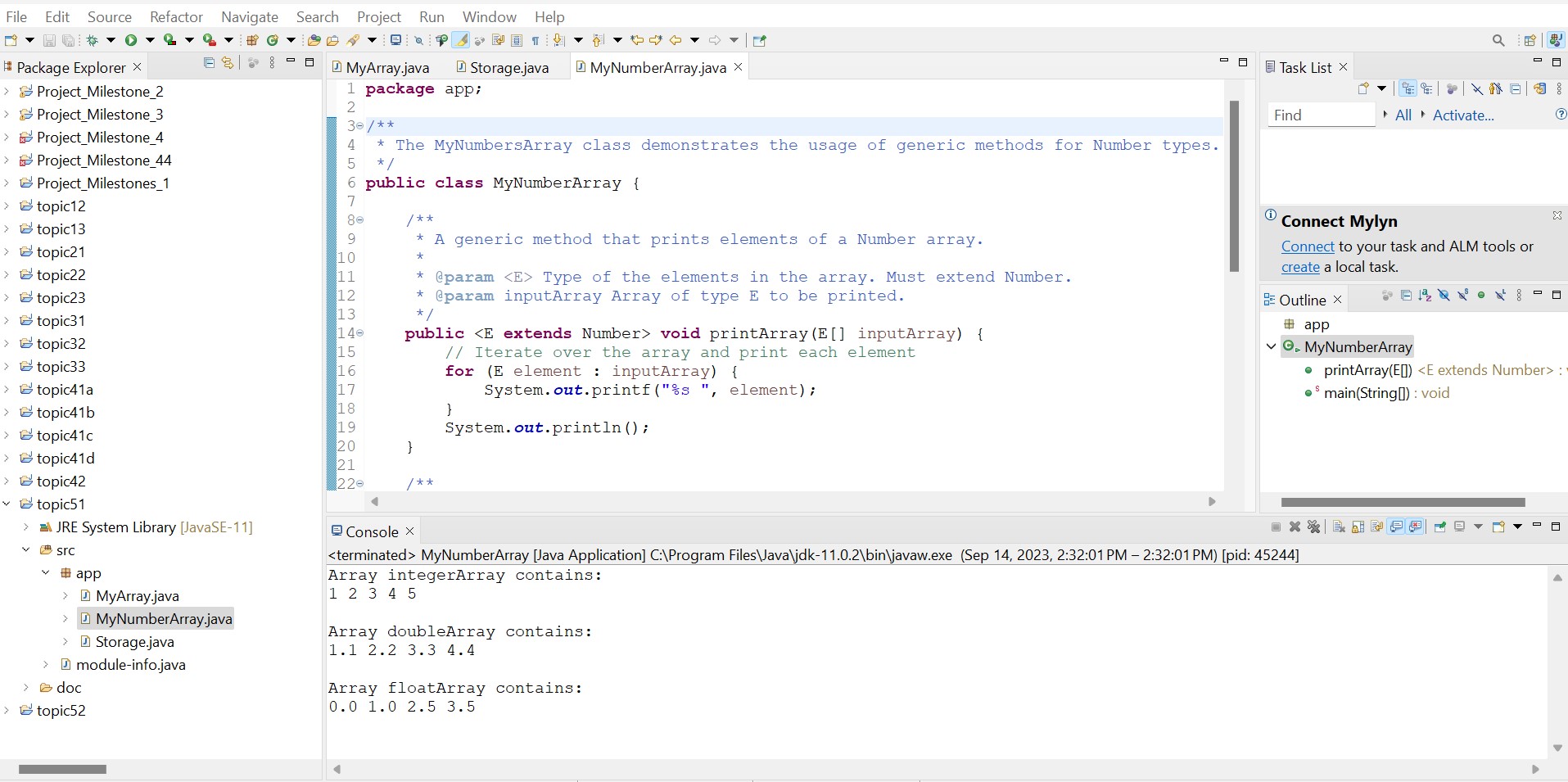


### **3. MyNumberArray Class:**

**Purpose**:  
This class is a variation of the MyArray class, but it restricts the operation to arrays whose elements are subclasses of Number (e.g., Integer, Double, Float, etc.).

**Key Features**:  
a) **Generic Method print Array with a Bounded Type Parameter**: The method accepts an array whose elements are of type E, where E extends Number. This ensures that only arrays with numeric types can be passed.

**Sample Usage**:  
The main method creates three arrays (Integer, Double, and Float). The print Array method is called for each array, showcasing that the method works with all numeric types but would not accept, for example, a String array.



**Part 2: Java Collections Framework**

## **Theory of Operation**

### **Overview**

The provided code showcases basic operations of Java collections: ArrayLists, HashMaps, LinkedLists (as a Queue), and Stacks. The operations on these collections cover common tasks like addition, retrieval, iteration, and removal of elements. The collection elements span types including Integer and String.

### **PlayList Class**

This class demonstrates how to perform basic operations using ArrayLists of both Integer and String types.

**Creation**: The ArrayLists are instantiated.

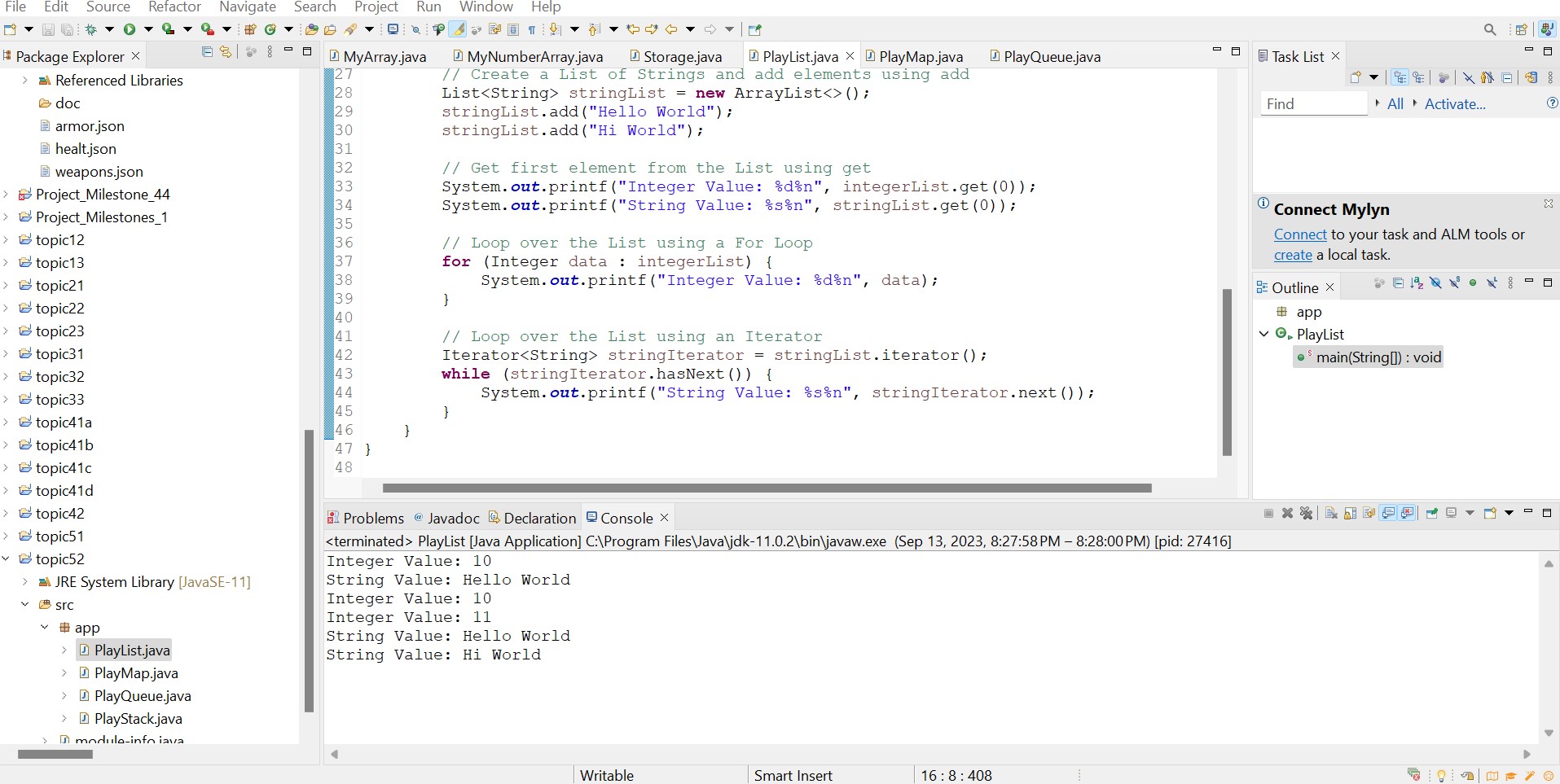
**Addition**: Elements are added to the ArrayLists using the add method.

**Retrieval**:

The first element from each ArrayList is accessed using the get method.

**Iteration**

A for-each loop iterates over the Integer ArrayList and an Iterator iterates over the String ArrayList.



### **PlayMap Class**

This class highlights fundamental operations on HashMaps.

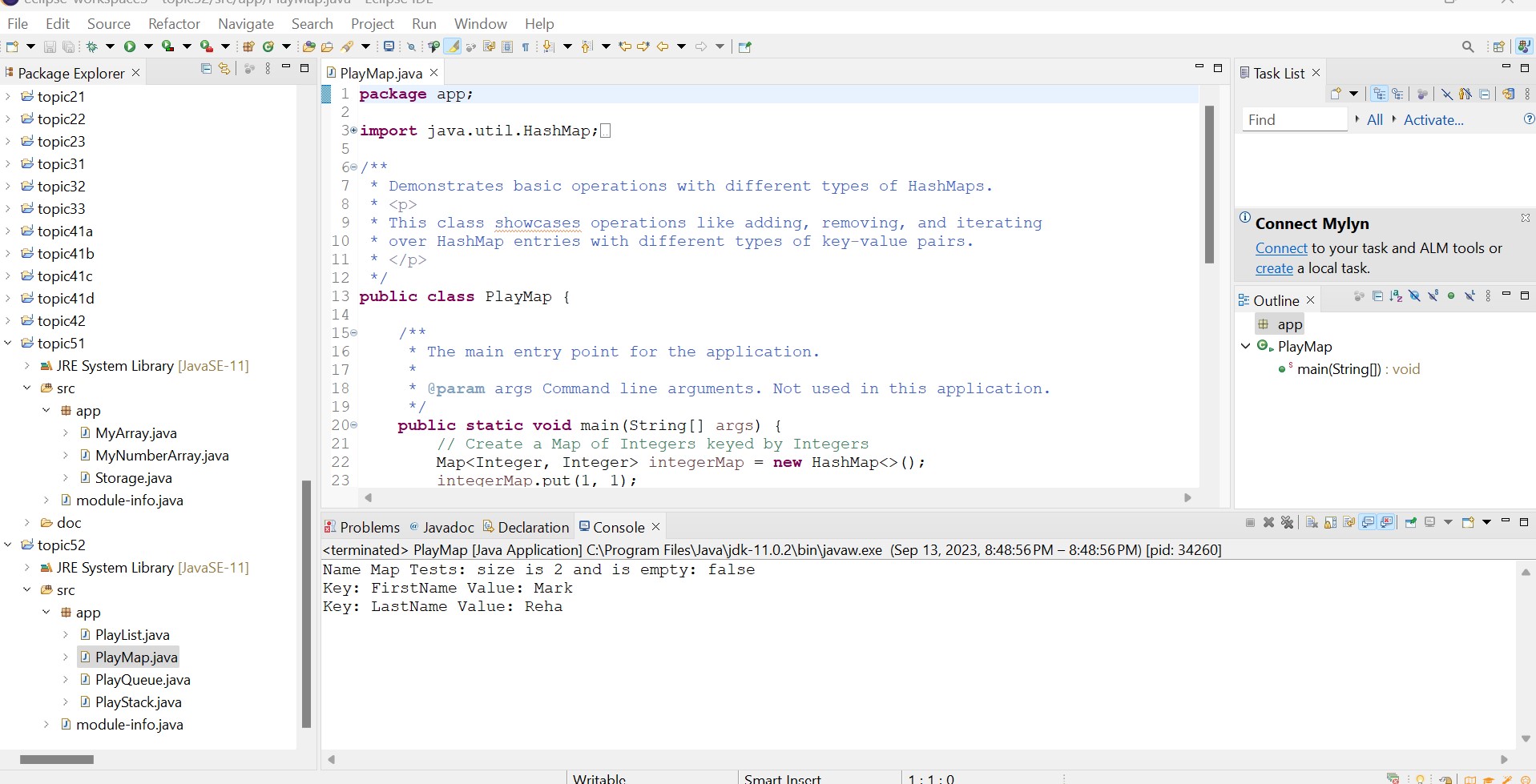
**Creation**: Three different HashMaps are created. One with Integer keys and values, one with Integer keys and String values, and the last with String keys and values.

**Addition**: Elements are added to the Maps using the put method.

**Size & Emptiness Check**: The size method fetches the number of entries in a map, and **isEmpty** checks if the map has no entries.

**Iteration**: A for-each loop is used to iterate over the Map's entries, extracting keys and values.

**Removal & Clear**: Entries from the Maps are removed individually using the remove method, and entire Maps are cleared using the clear method.



### **PlayQueue Class**

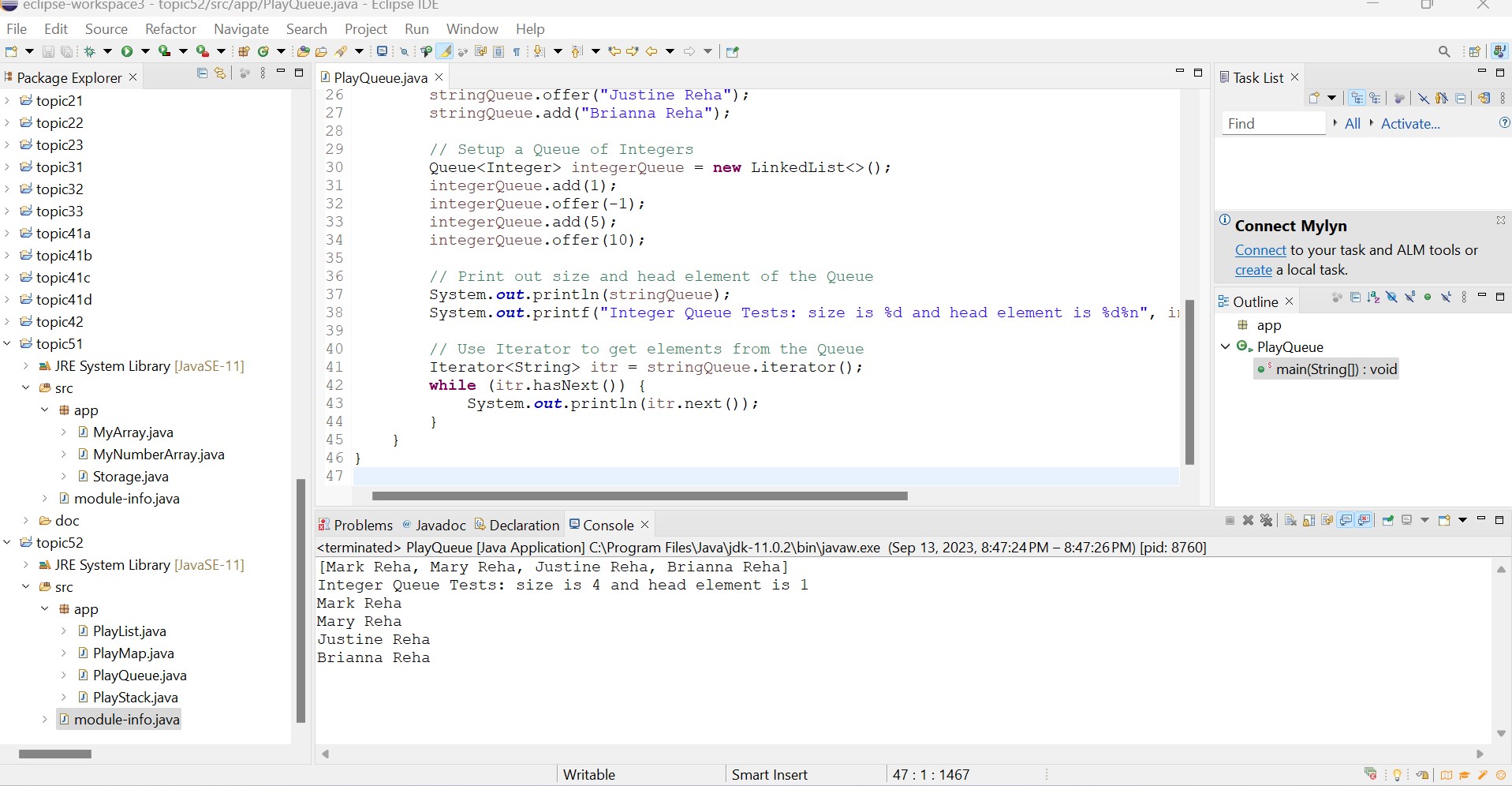
This class details fundamental operations using Queues.

**Creation**: Two LinkedLists, serving as Queues, are instantiated. One for Strings and one for Integers.

**Addition**: Elements are added to the Queues using both the add and offer methods.

**Peek**: The peek method is utilized to see the head of the Queue without removing it.

**Iteration**: An Iterator is used to access and display the elements from the String Queue.



### **PlayStack Class**

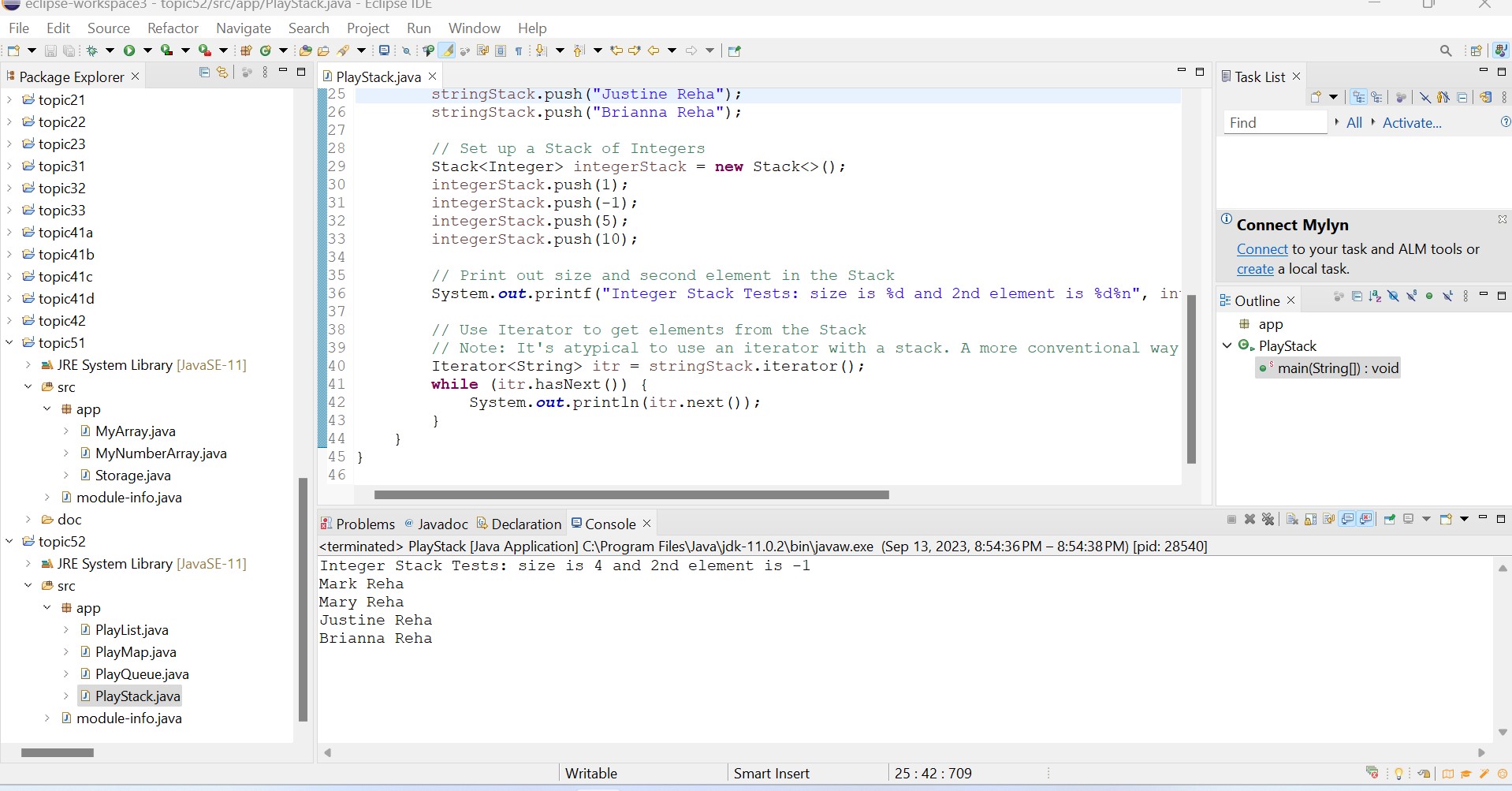
This class explains core operations using Stacks.

**Creation**: Two Stacks are instantiated, one for Strings and another for Integers.

**Push**: The push method adds elements onto the Stacks.

**Element Retrieval**: The elementAt method retrieves an element from a specific position without removing it. This is a non-standard operation for Stacks.

**Iteration**: Although not typical, an Iterator is used to access and display the elements from the String Stack. Conventionally, the pop method would be used until the Stack is empty.



**Complete Collection Quiz-1. Take a screenshot of your completed Quiz.**

