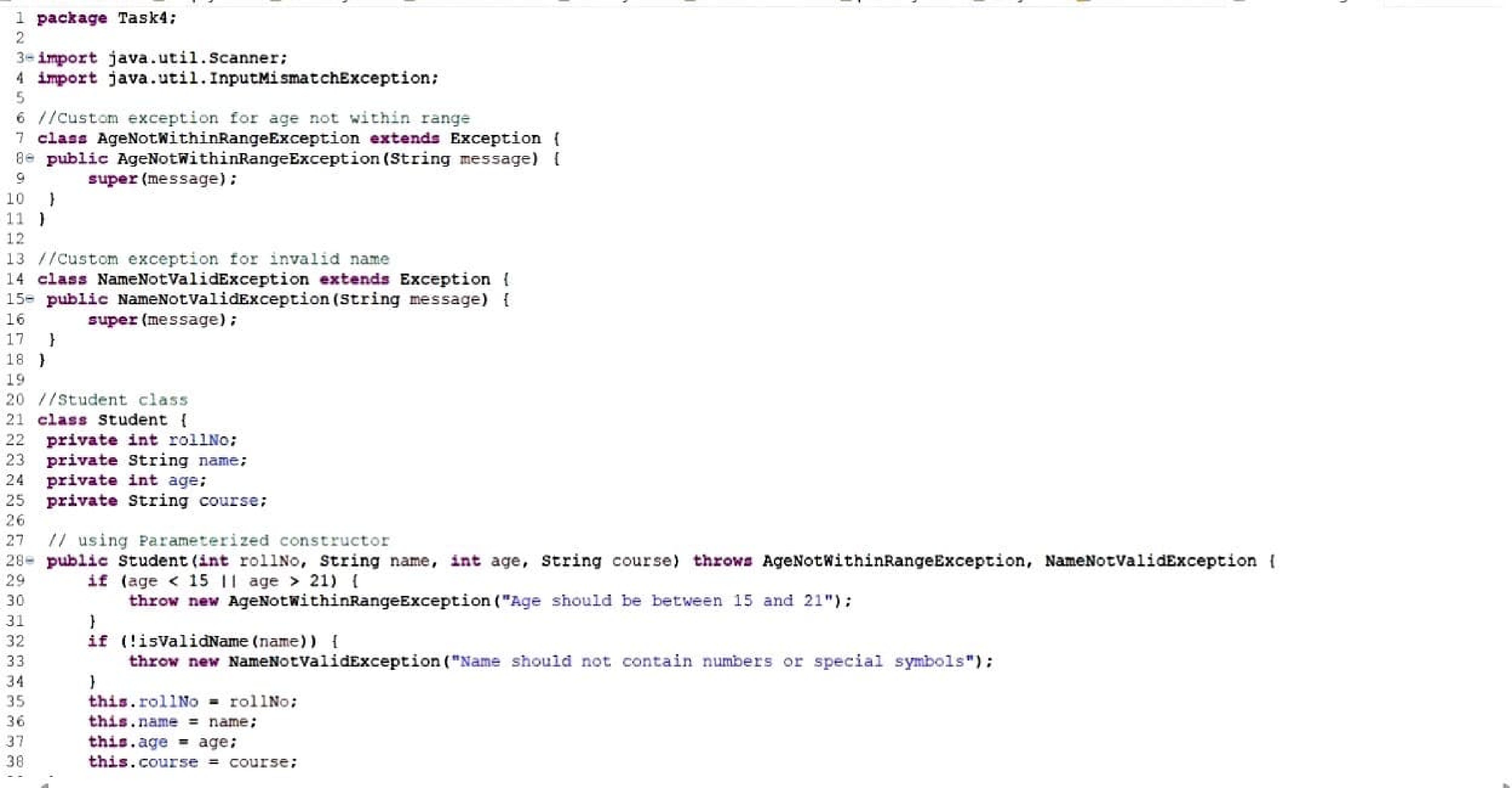
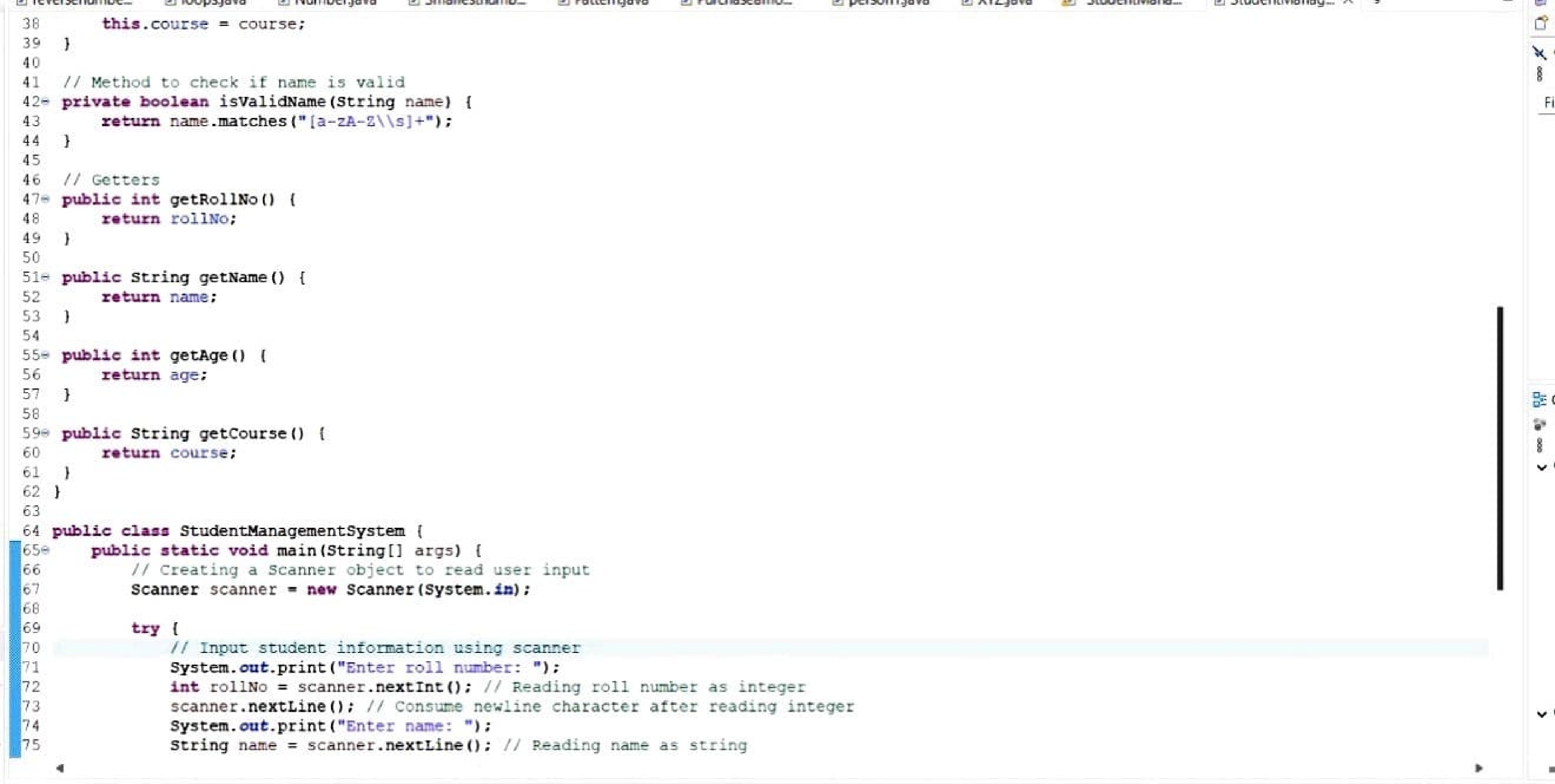
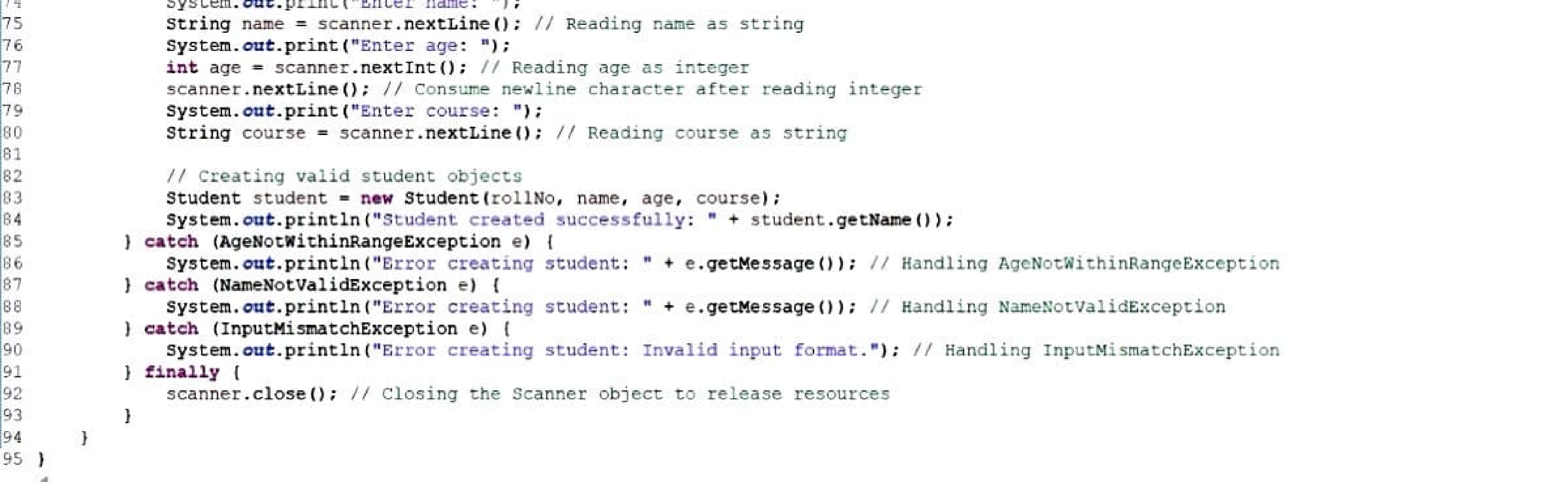


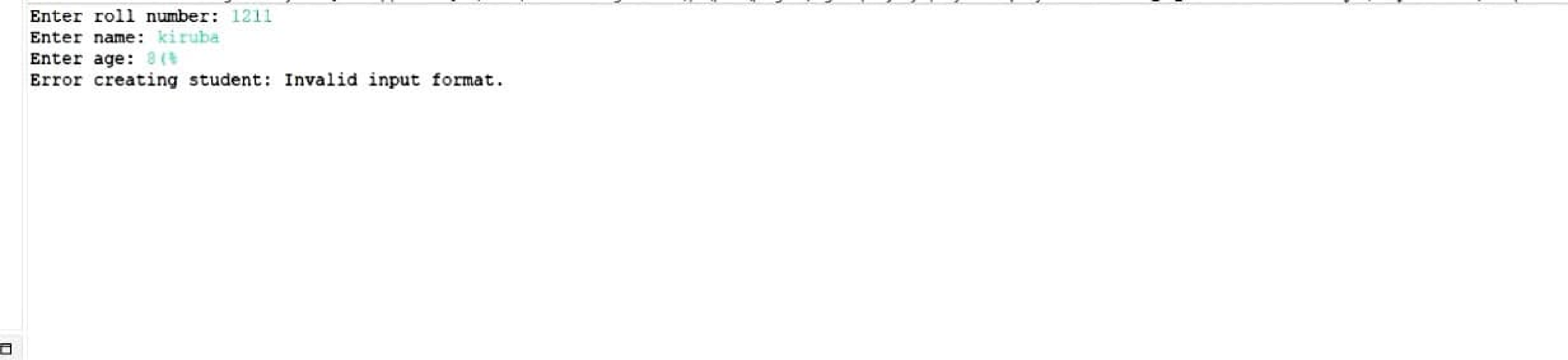
```java
package Task4;

import java.util.Scanner;
import java.util.InputMismatchException;

//Custom exception for age not within range
class AgeNotWithinRangeException extends Exception {
 public AgeNotWithinRangeException(String message) {
     super(message);
 }
}

//Custom exception for invalid name
class NameNotValidException extends Exception {
 public NameNotValidException(String message) {
     super(message);
 }
}

//Student class
class Student {
 private int rollNo;
 private String name;
 private int age;
 private String course;

 // using Parameterized constructor
 public Student(int rollNo, String name, int age, String course) throws AgeNotWithinRangeException, NameNotValidException {
     if (age < 15 || age > 21) {
         throw new AgeNotWithinRangeException("Age should be between 15 and 21");
     }
     if (!isValidName(name)) {
         throw new NameNotValidException("Name should not contain numbers or special symbols");
     }
     this.rollNo = rollNo;
     this.name = name;
     this.age = age;
     this.course = course;
```



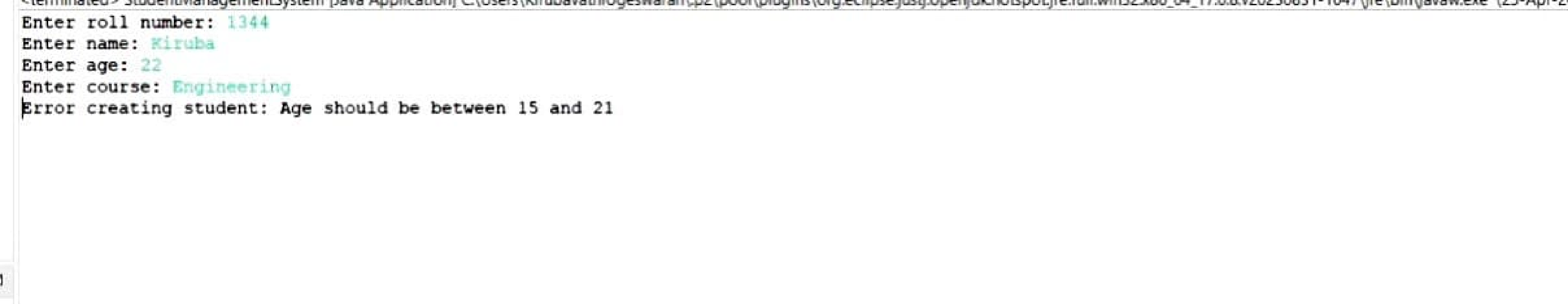
```
     this.course = course;
 }

 // Method to check if name is valid
 private boolean isValidName(String name) {
     return name.matches("[a-zA-Z\\s]+");
 }

 // Getters
 public int getRollNo() {
     return rollNo;
 }

 public String getName() {
     return name;
 }

 public int getAge() {
     return age;
 }

 public String getCourse() {
     return course;
 }
}

public class StudentManagementSystem {
    public static void main(String[] args) {
        // Creating a Scanner object to read user input
        Scanner scanner = new Scanner(System.in);

        try {
            // Input student information using scanner
            System.out.print("Enter roll number: ");
            int rollNo = scanner.nextInt(); // Reading roll number as integer
            scanner.nextLine(); // Consume newline character after reading integer
            System.out.print("Enter name: ");
            String name = scanner.nextLine(); // Reading name as string
```

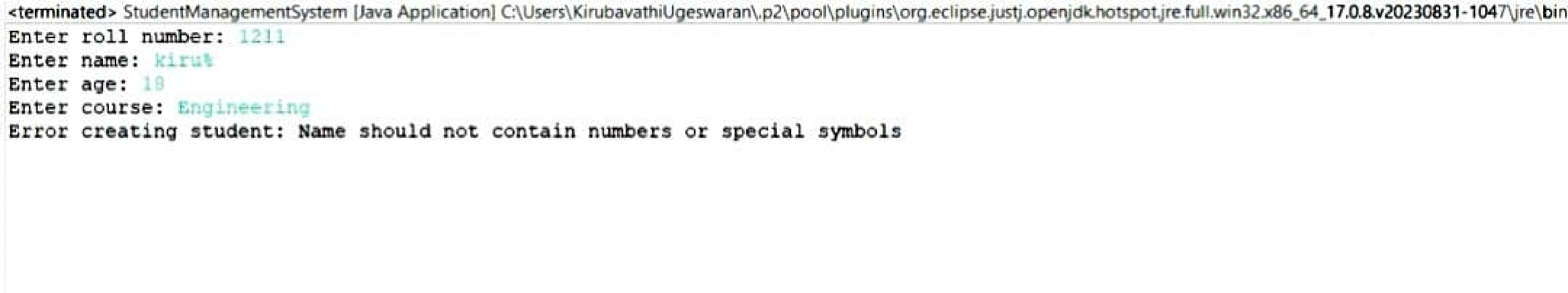


```java
            System.out.print("Enter name: ");
            String name = scanner.nextLine(); // Reading name as string
            System.out.print("Enter age: ");
            int age = scanner.nextInt(); // Reading age as integer
            scanner.nextLine(); // Consume newline character after reading integer
            System.out.print("Enter course: ");
            String course = scanner.nextLine(); // Reading course as string

            // Creating valid student objects
            Student student = new Student(rollNo, name, age, course);
            System.out.println("Student created successfully: " + student.getName());
        } catch (AgeNotWithinRangeException e) {
            System.out.println("Error creating student: " + e.getMessage()); // Handling AgeNotWithinRangeException
        } catch (NameNotValidException e) {
            System.out.println("Error creating student: " + e.getMessage()); // Handling NameNotValidException
        } catch (InputMismatchException e) {
            System.out.println("Error creating student: Invalid input format."); // Handling InputMismatchException
        } finally {
            scanner.close(); // Closing the Scanner object to release resources
        }
    }
}
```

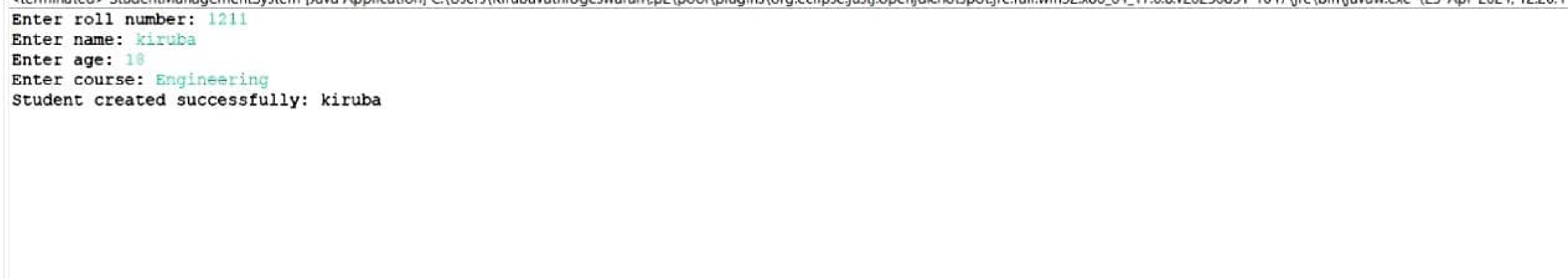


```
Enter roll number: 1211
Enter name: kiruba
Enter age: 8(%
Error creating student: Invalid input format.
```

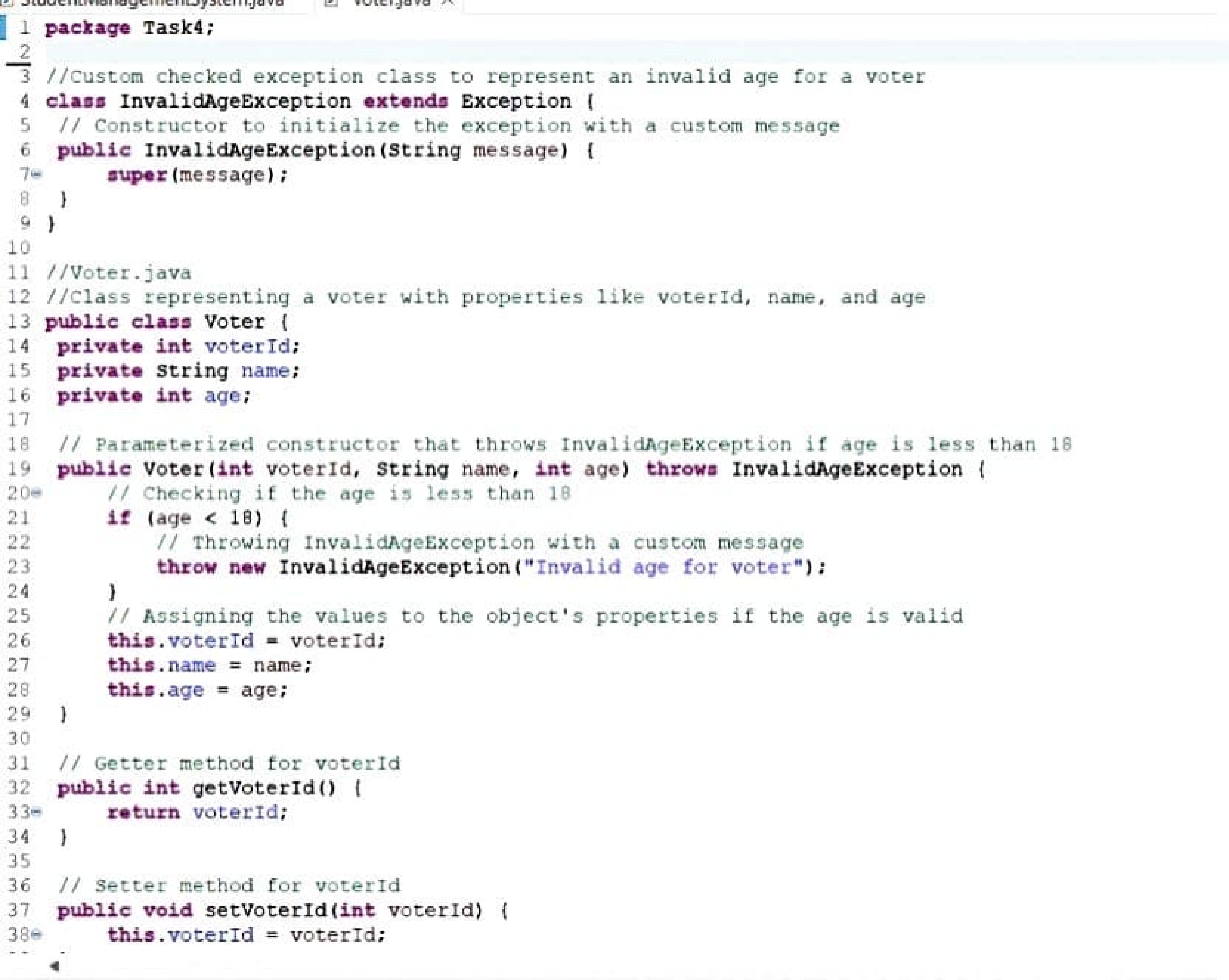

```
<terminated> StudentManagementSystem [Java Application] C:\Users\Kirubavathiyogeswaran\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (23-Apr-2
Enter roll number: 1344
Enter name: Kiruba
Enter age: 22
Enter course: Engineering
Error creating student: Age should be between 15 and 21
```



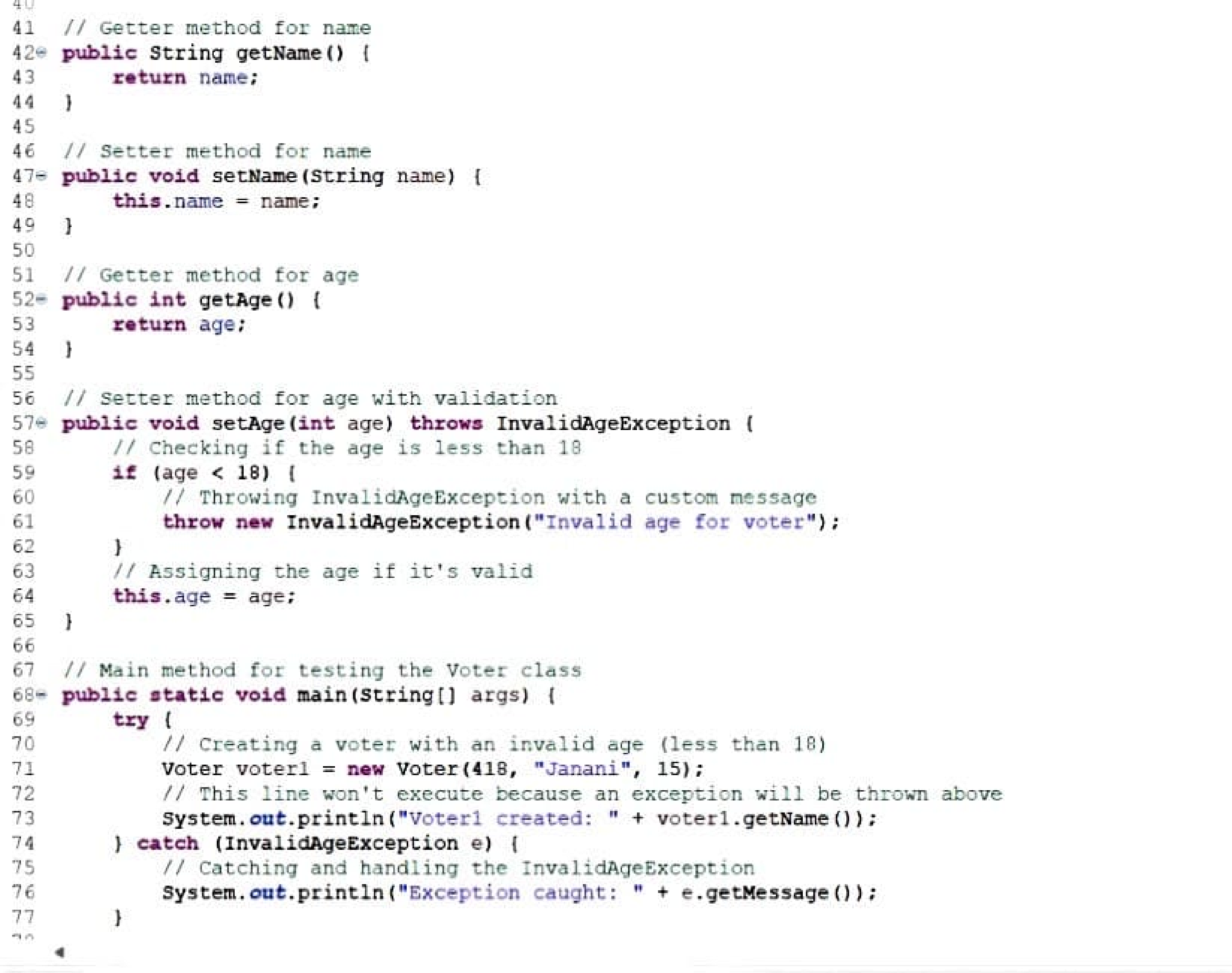
```
<terminated> StudentManagementSystem [Java Application] C:\Users\KirubavathiUgeswaran\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin
Enter roll number: 1211
Enter name: kiru%
Enter age: 18
Enter course: Engineering
Error creating student: Name should not contain numbers or special symbols
```



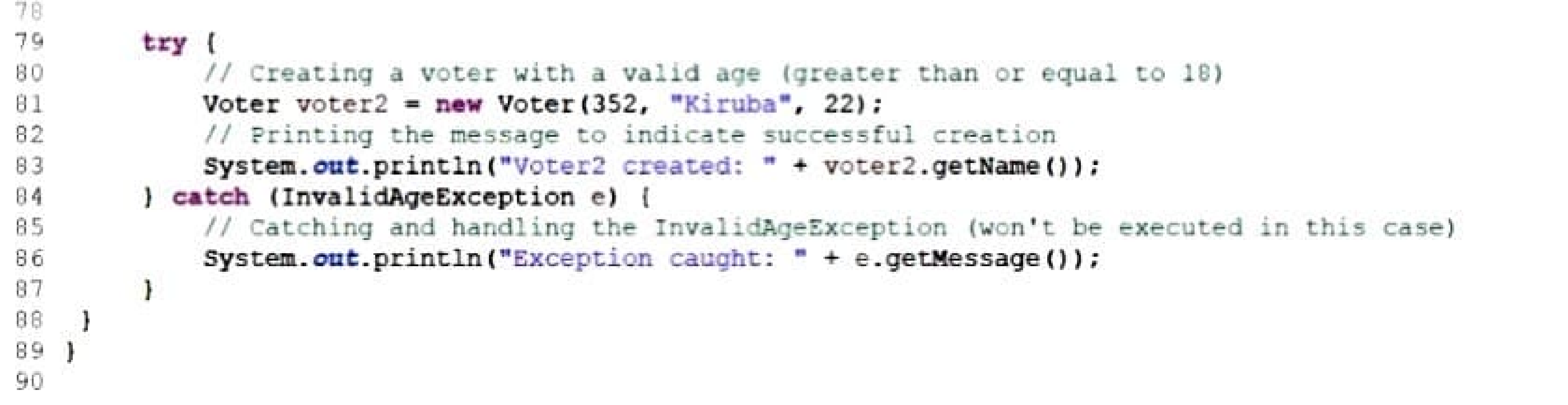
```
Enter roll number: 1211
Enter name: kiruba
Enter age: 18
Enter course: Engineering
Student created successfully: kiruba
```



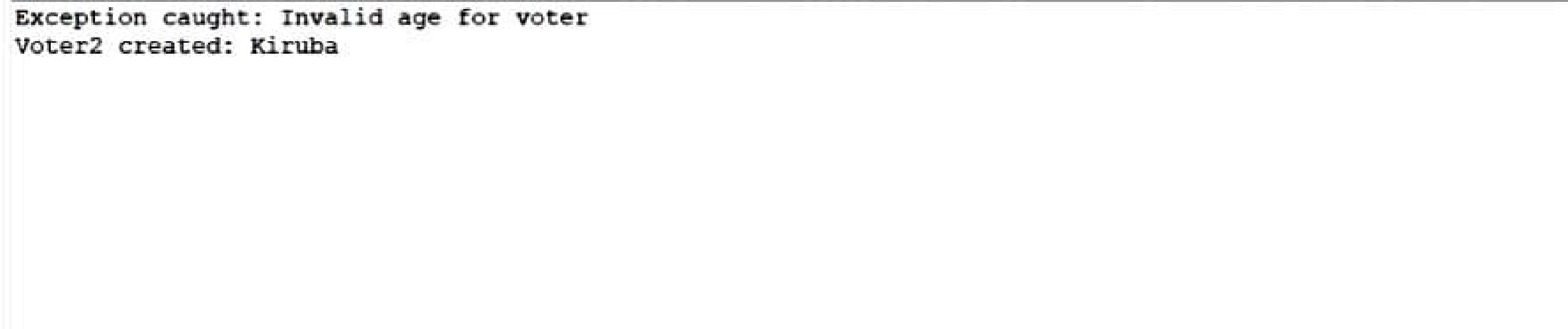
```java
package Task4;

//Custom checked exception class to represent an invalid age for a voter
class InvalidAgeException extends Exception {
 // Constructor to initialize the exception with a custom message
 public InvalidAgeException(String message) {
     super(message);
 }
}

//Voter.java
//Class representing a voter with properties like voterId, name, and age
public class Voter {
 private int voterId;
 private String name;
 private int age;

 // Parameterized constructor that throws InvalidAgeException if age is less than 18
 public Voter(int voterId, String name, int age) throws InvalidAgeException {
     // Checking if the age is less than 18
     if (age < 18) {
         // Throwing InvalidAgeException with a custom message
         throw new InvalidAgeException("Invalid age for voter");
     }
     // Assigning the values to the object's properties if the age is valid
     this.voterId = voterId;
     this.name = name;
     this.age = age;
 }

 // Getter method for voterId
 public int getVoterId() {
     return voterId;
 }

 // Setter method for voterId
 public void setVoterId(int voterId) {
     this.voterId = voterId;
```



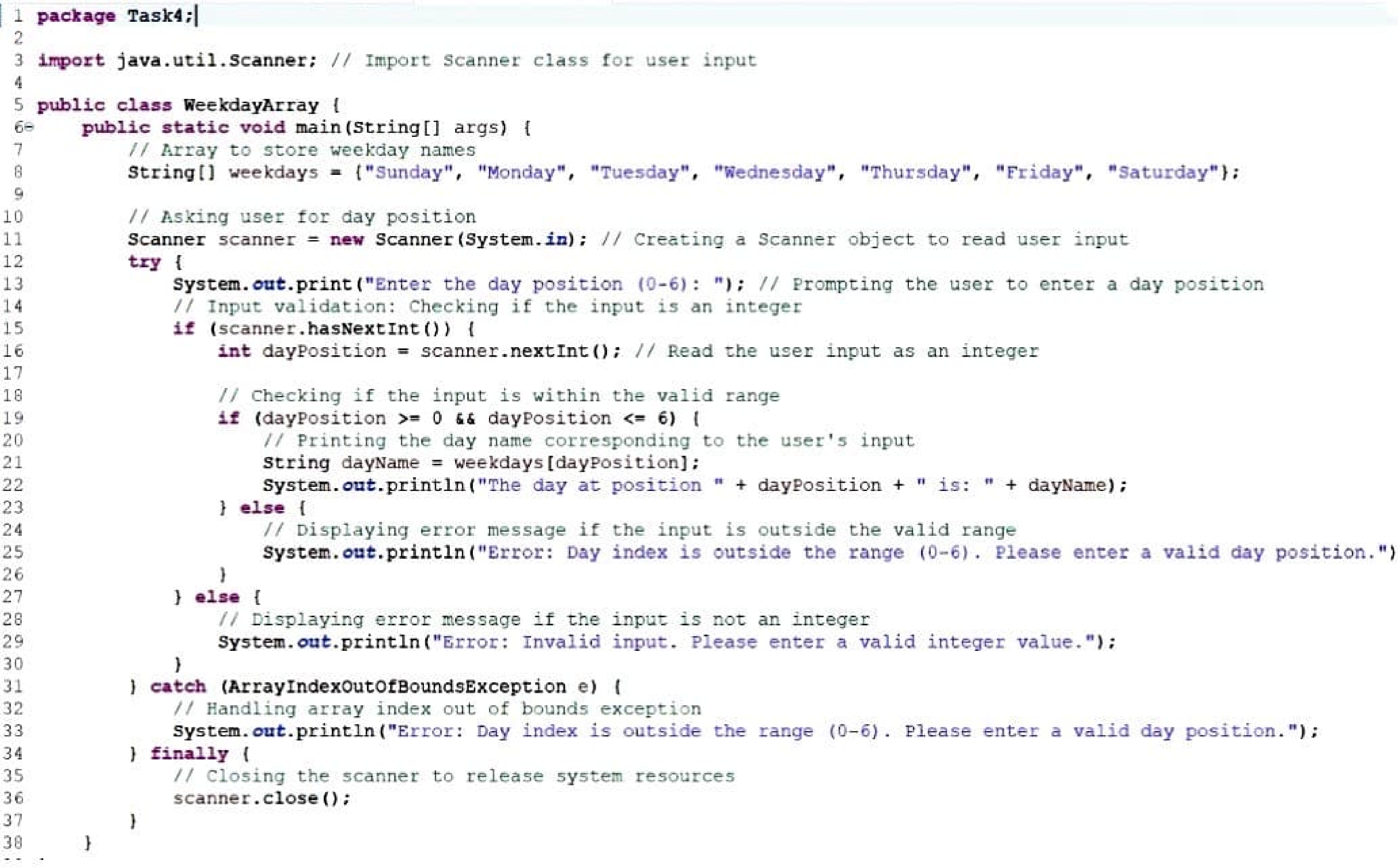
```java
// Getter method for name
public String getName() {
    return name;
}

// Setter method for name
public void setName(String name) {
    this.name = name;
}

// Getter method for age
public int getAge() {
    return age;
}

// Setter method for age with validation
public void setAge(int age) throws InvalidAgeException {
    // Checking if the age is less than 18
    if (age < 18) {
        // Throwing InvalidAgeException with a custom message
        throw new InvalidAgeException("Invalid age for voter");
    }
    // Assigning the age if it's valid
    this.age = age;
}

// Main method for testing the Voter class
public static void main(String[] args) {
    try {
        // Creating a voter with an invalid age (less than 18)
        Voter voter1 = new Voter(418, "Janani", 15);
        // This line won't execute because an exception will be thrown above
        System.out.println("Voter1 created: " + voter1.getName());
    } catch (InvalidAgeException e) {
        // Catching and handling the InvalidAgeException
        System.out.println("Exception caught: " + e.getMessage());
    }
```



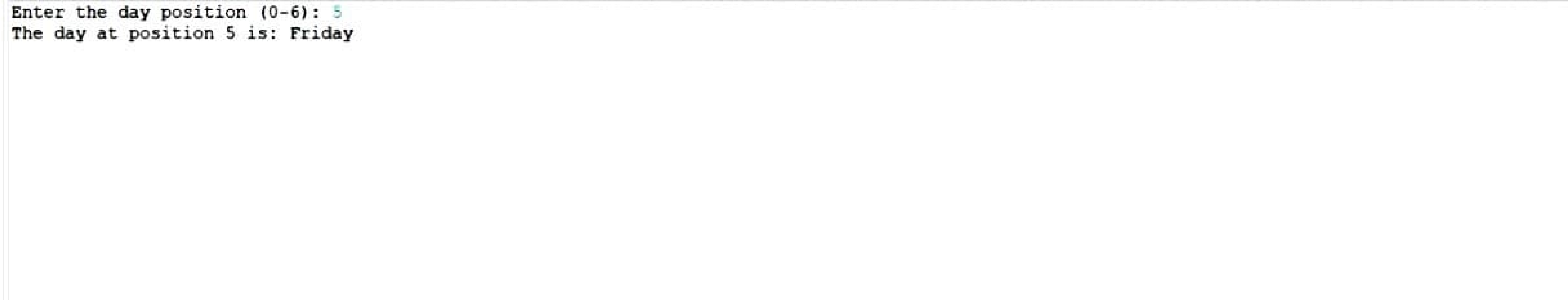
```java

        try {
            // Creating a voter with a valid age (greater than or equal to 18)
            Voter voter2 = new Voter(352, "Kiruba", 22);
            // Printing the message to indicate successful creation
            System.out.println("Voter2 created: " + voter2.getName());
        } catch (InvalidAgeException e) {
            // Catching and handling the InvalidAgeException (won't be executed in this case)
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}

```



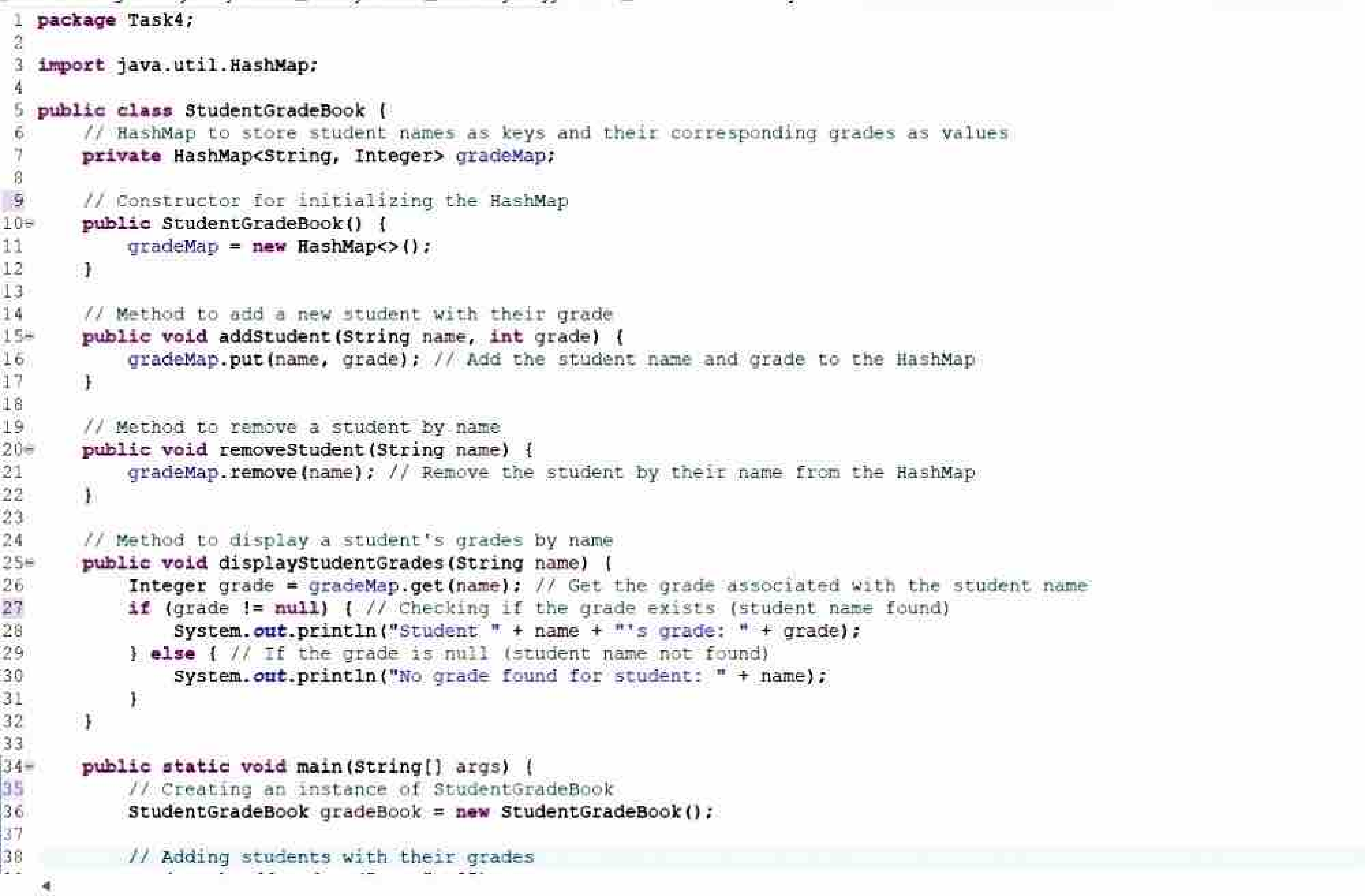
```
Exception caught: Invalid age for voter
Voter2 created: Kiruba
```



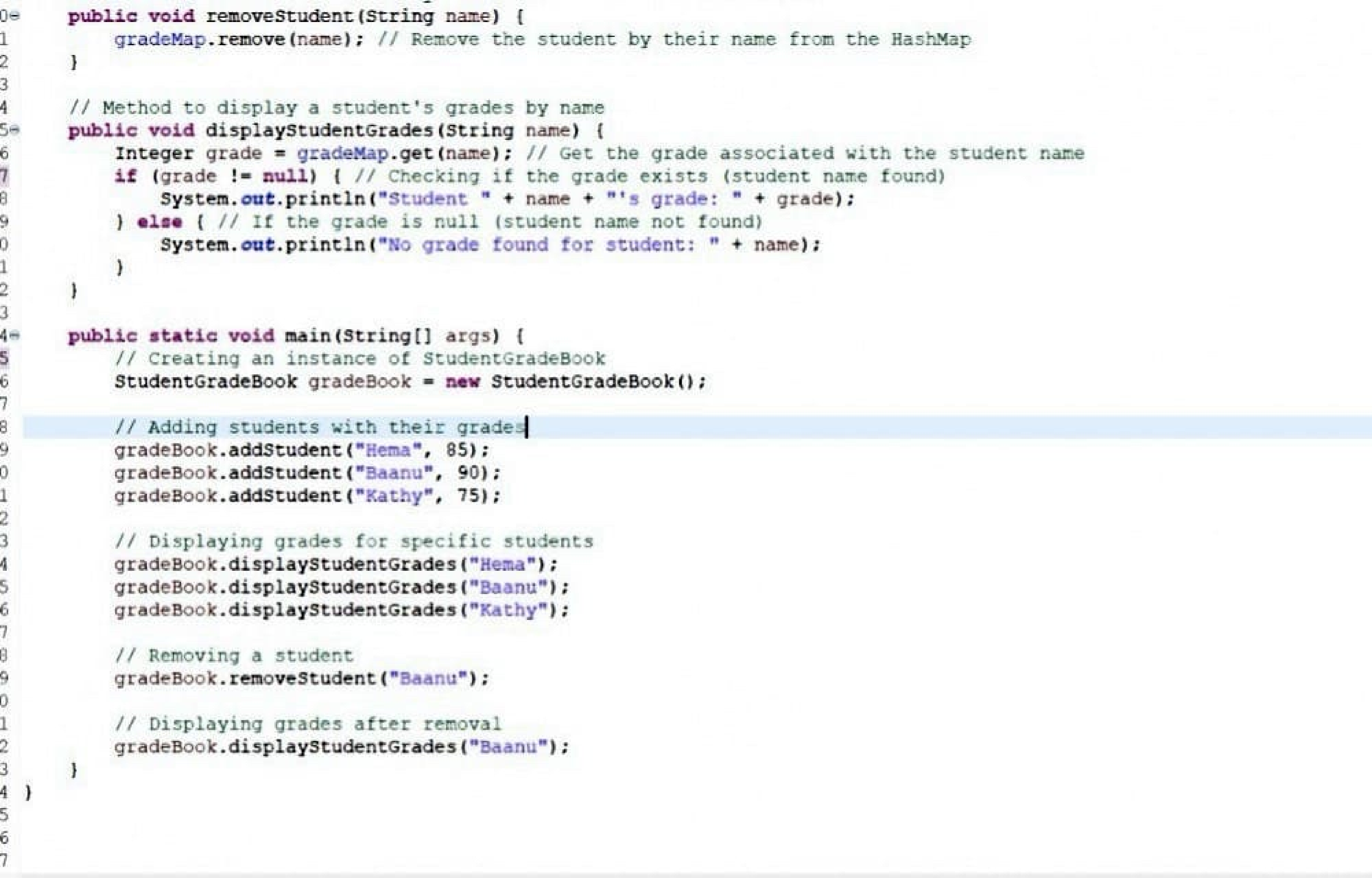
```java
package Task4;

import java.util.Scanner; // Import Scanner class for user input

public class WeekdayArray {
    public static void main(String[] args) {
        // Array to store weekday names
        String[] weekdays = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

        // Asking user for day position
        Scanner scanner = new Scanner(System.in); // Creating a Scanner object to read user input
        try {
            System.out.print("Enter the day position (0-6): "); // Prompting the user to enter a day position
            // Input validation: Checking if the input is an integer
            if (scanner.hasNextInt()) {
                int dayPosition = scanner.nextInt(); // Read the user input as an integer

                // Checking if the input is within the valid range
                if (dayPosition >= 0 && dayPosition <= 6) {
                    // Printing the day name corresponding to the user's input
                    String dayName = weekdays[dayPosition];
                    System.out.println("The day at position " + dayPosition + " is: " + dayName);
                } else {
                    // Displaying error message if the input is outside the valid range
                    System.out.println("Error: Day index is outside the range (0-6). Please enter a valid day position.")
                }
            } else {
                // Displaying error message if the input is not an integer
                System.out.println("Error: Invalid input. Please enter a valid integer value.");
            }
        } catch (ArrayIndexOutOfBoundsException e) {
            // Handling array index out of bounds exception
            System.out.println("Error: Day index is outside the range (0-6). Please enter a valid day position.");
        } finally {
            // Closing the scanner to release system resources
            scanner.close();
        }
    }
```



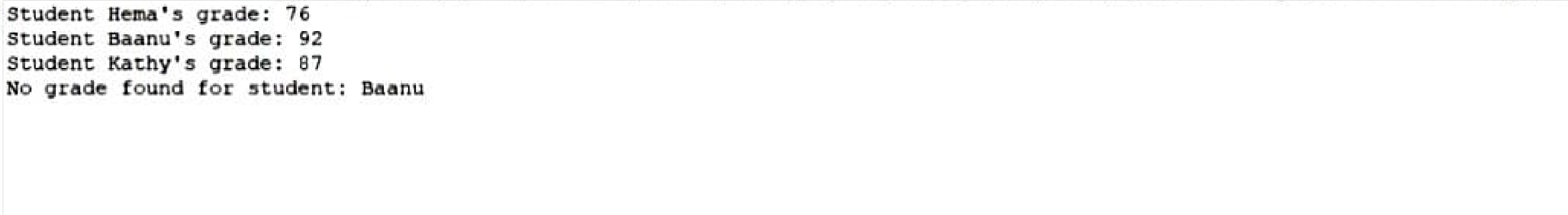
```
Enter the day position (0-6): 5
The day at position 5 is: Friday
```



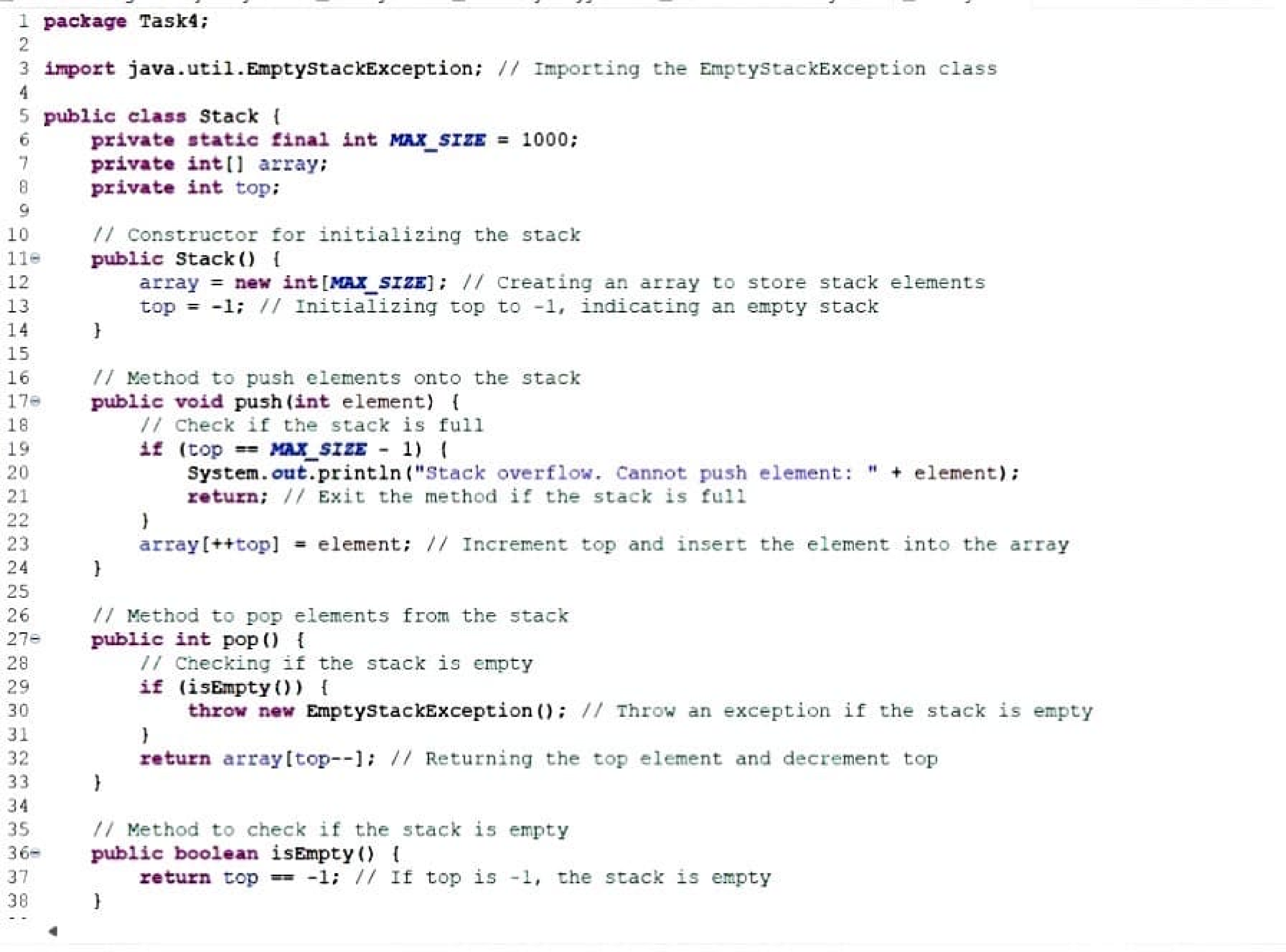
```java
package Task4;

import java.util.HashMap;

public class StudentGradeBook {
    // HashMap to store student names as keys and their corresponding grades as values
    private HashMap<String, Integer> gradeMap;

    // Constructor for initializing the HashMap
    public StudentGradeBook() {
        gradeMap = new HashMap<>();
    }

    // Method to add a new student with their grade
    public void addStudent(String name, int grade) {
        gradeMap.put(name, grade); // Add the student name and grade to the HashMap
    }

    // Method to remove a student by name
    public void removeStudent(String name) {
        gradeMap.remove(name); // Remove the student by their name from the HashMap
    }

    // Method to display a student's grades by name
    public void displayStudentGrades(String name) {
        Integer grade = gradeMap.get(name); // Get the grade associated with the student name
        if (grade != null) { // Checking if the grade exists (student name found)
            System.out.println("Student " + name + "'s grade: " + grade);
        } else { // If the grade is null (student name not found)
            System.out.println("No grade found for student: " + name);
        }
    }

    public static void main(String[] args) {
        // Creating an instance of StudentGradeBook
        StudentGradeBook gradeBook = new StudentGradeBook();

        // Adding students with their grades
```



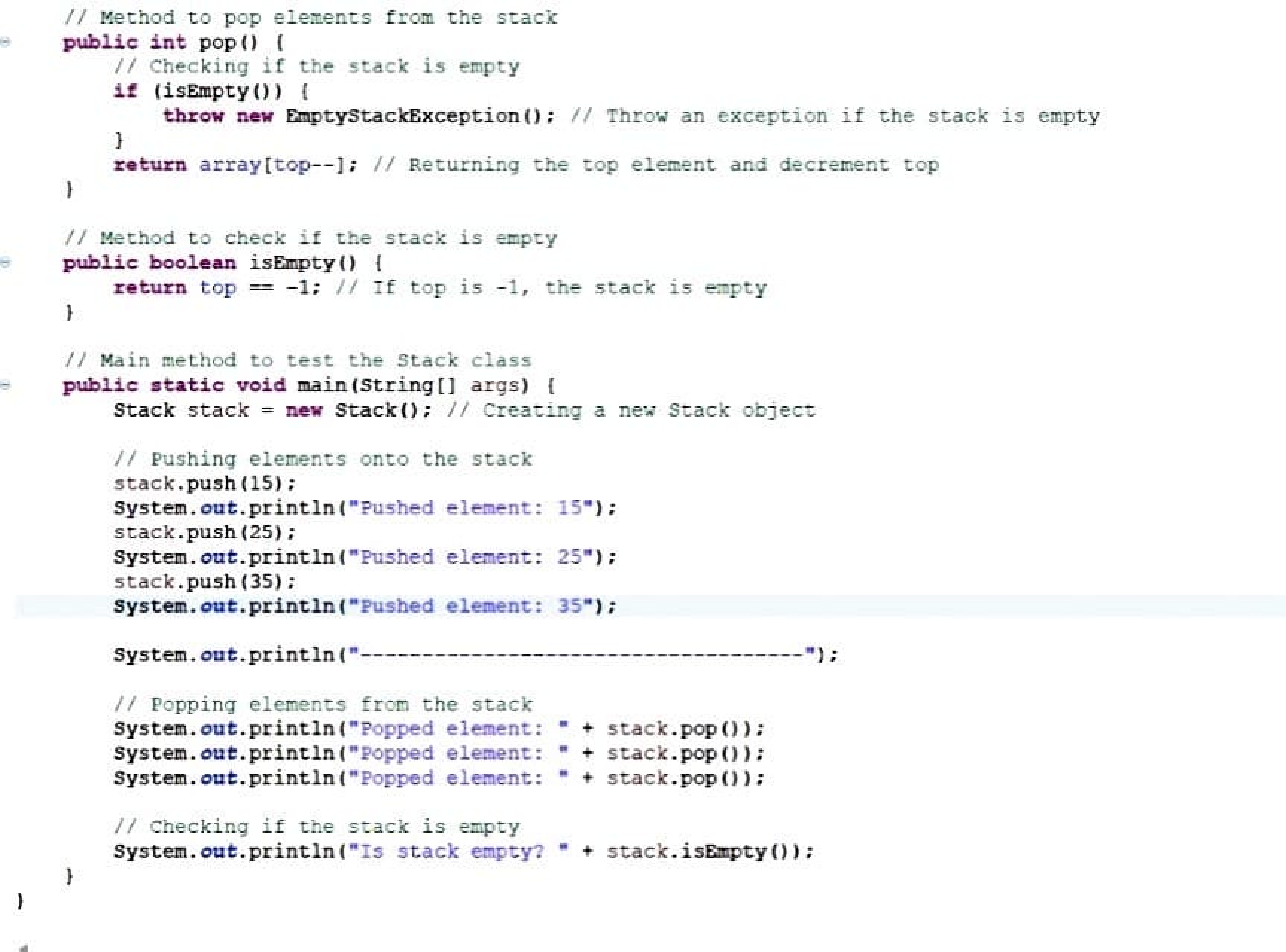
```java
    public void removeStudent(String name) {
        gradeMap.remove(name); // Remove the student by their name from the HashMap
    }

    // Method to display a student's grades by name
    public void displayStudentGrades(String name) {
        Integer grade = gradeMap.get(name); // Get the grade associated with the student name
        if (grade != null) { // Checking if the grade exists (student name found)
            System.out.println("Student " + name + "'s grade: " + grade);
        } else { // If the grade is null (student name not found)
            System.out.println("No grade found for student: " + name);
        }
    }

    public static void main(String[] args) {
        // Creating an instance of StudentGradeBook
        StudentGradeBook gradeBook = new StudentGradeBook();

        // Adding students with their grades
        gradeBook.addStudent("Hema", 85);
        gradeBook.addStudent("Baanu", 90);
        gradeBook.addStudent("Kathy", 75);

        // Displaying grades for specific students
        gradeBook.displayStudentGrades("Hema");
        gradeBook.displayStudentGrades("Baanu");
        gradeBook.displayStudentGrades("Kathy");

        // Removing a student
        gradeBook.removeStudent("Baanu");

        // Displaying grades after removal
        gradeBook.displayStudentGrades("Baanu");
    }
}
```



```
Student Hema's grade: 76
Student Baanu's grade: 92
Student Kathy's grade: 87
No grade found for student: Baanu
```



```java
package Task4;

import java.util.EmptyStackException; // Importing the EmptyStackException class

public class Stack {
    private static final int MAX_SIZE = 1000;
    private int[] array;
    private int top;

    // Constructor for initializing the stack
    public Stack() {
        array = new int[MAX_SIZE]; // Creating an array to store stack elements
        top = -1; // Initializing top to -1, indicating an empty stack
    }

    // Method to push elements onto the stack
    public void push(int element) {
        // Check if the stack is full
        if (top == MAX_SIZE - 1) {
            System.out.println("Stack overflow. Cannot push element: " + element);
            return; // Exit the method if the stack is full
        }
        array[++top] = element; // Increment top and insert the element into the array
    }

    // Method to pop elements from the stack
    public int pop() {
        // Checking if the stack is empty
        if (isEmpty()) {
            throw new EmptyStackException(); // Throw an exception if the stack is empty
        }
        return array[top--]; // Returning the top element and decrement top
    }

    // Method to check if the stack is empty
    public boolean isEmpty() {
        return top == -1; // If top is -1, the stack is empty
    }
```



```java
    // Method to pop elements from the stack
    public int pop() {
        // Checking if the stack is empty
        if (isEmpty()) {
            throw new EmptyStackException(); // Throw an exception if the stack is empty
        }
        return array[top--]; // Returning the top element and decrement top
    }

    // Method to check if the stack is empty
    public boolean isEmpty() {
        return top == -1; // If top is -1, the stack is empty
    }

    // Main method to test the Stack class
    public static void main(String[] args) {
        Stack stack = new Stack(); // Creating a new Stack object

        // Pushing elements onto the stack
        stack.push(15);
        System.out.println("Pushed element: 15");
        stack.push(25);
        System.out.println("Pushed element: 25");
        stack.push(35);
        System.out.println("Pushed element: 35");

        System.out.println("------------------------------------");

        // Popping elements from the stack
        System.out.println("Popped element: " + stack.pop());
        System.out.println("Popped element: " + stack.pop());
        System.out.println("Popped element: " + stack.pop());

        // Checking if the stack is empty
        System.out.println("Is stack empty? " + stack.isEmpty());
    }
}
```

```
Pushed element: 15
Pushed element: 25
Pushed element: 35
------------------------------------
Popped element: 35
Popped element: 25
Popped element: 15
Is stack empty? true
```