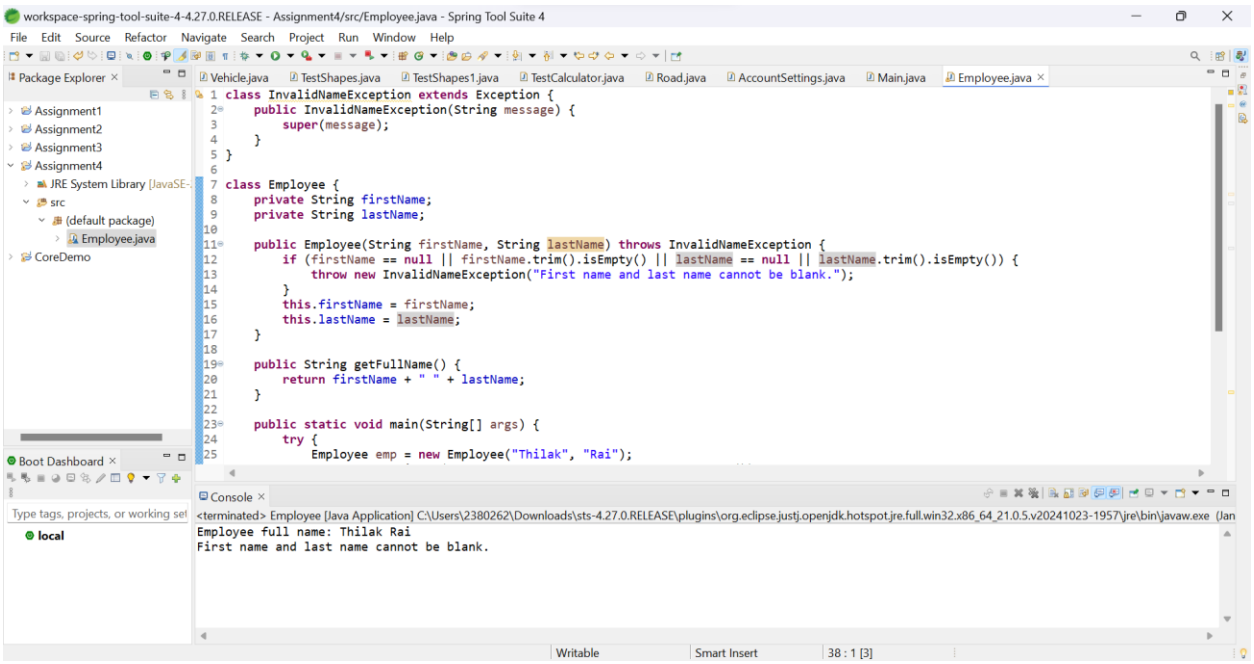
## TASK1:

```
class InvalidNameException extends Exception {
  public InvalidNameException(String message) {
    super(message);
  }
}
class Employee {
  private String firstName;
  private String lastName;
  public Employee(String firstName, String lastName) throws InvalidNameException {
    if (firstName == null || firstName.trim().isEmpty() || lastName == null || lastName.trim().isEmpty()) {
      throw new InvalidNameException("First name and last name cannot be blank.");
    }
    this.firstName = firstName;
    this.lastName = lastName;
  }
  public String getFullName() {
    return firstName + " " + lastName;
  }
  public static void main(String[] args) {
    try {
      Employee emp = new Employee("Thilak", "Rai");
      System.out.println("Employee full name: " + emp.getFullName());
    } catch (InvalidNameException e) {
      System.out.println(e.getMessage());
    }
    try {
      Employee emp2 = new Employee("", "Smith");
    } catch (InvalidNameException e) {
```

```
        System.out.println(e.getMessage());
    }
  }
}
```

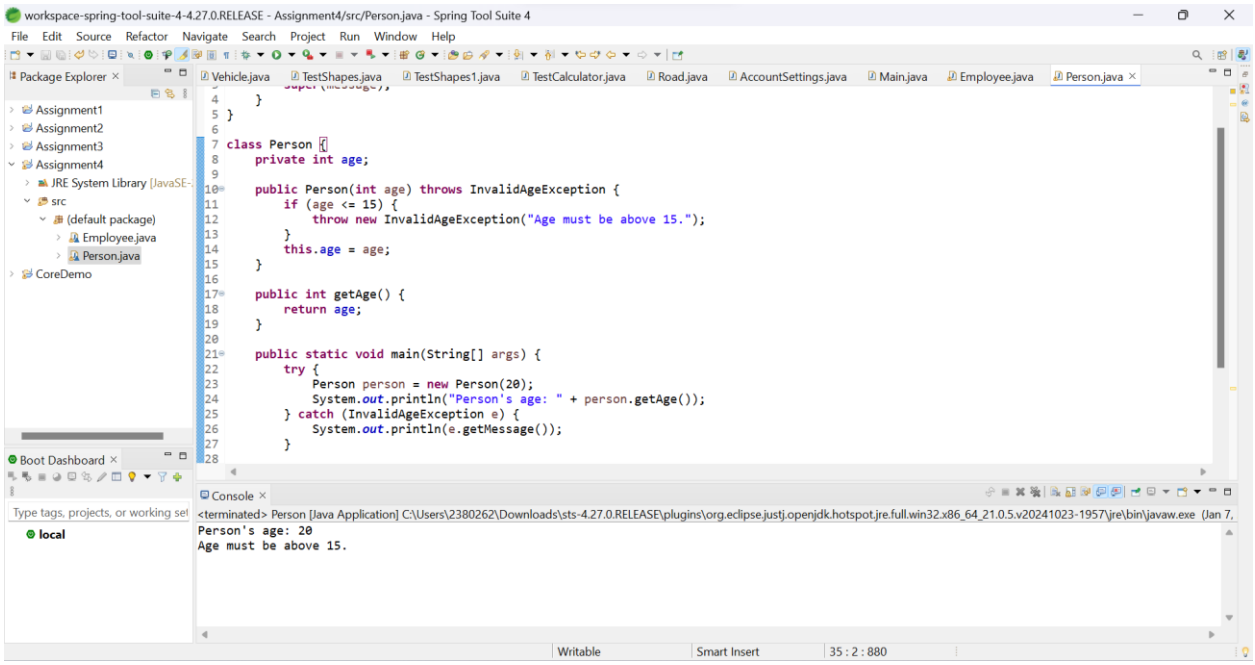



## TASK2:

```
class InvalidAgeException extends Exception {
  public InvalidAgeException(String message) {
    super(message);
  }
}
class Person {
  private int age;
  public Person(int age) throws InvalidAgeException {
    if (age <= 15) {
```

```
        throw new InvalidAgeException("Age must be above 15.");
    }
    this.age = age;
}
public int getAge() {
    return age;
}
public static void main(String[] args) {
    try {
        Person person = new Person(20);
        System.out.println("Person's age: " + person.getAge());
    } catch (InvalidAgeException e) {
        System.out.println(e.getMessage());
    }

    try {
        Person person2 = new Person(10);
    } catch (InvalidAgeException e) {
        System.out.println(e.getMessage());
    }
}
```



## TASK3:

```
package com.demo.exception;


class EmployeeException extends Exception {
   public EmployeeException(String message) {
      super(message);
   }
}


class Employee {
   private double salary;


   public Employee(double salary) throws EmployeeException {
      if (salary < 3000) {
         throw new EmployeeException("Salary cannot be below 3000.");
      }
      this.salary = salary;
   }


   public double getSalary() {
      return salary;
   }
   public static void main(String[] args) {
      try {
         Employee emp = new Employee(3500);
         System.out.println("Employee's salary: " + emp.getSalary());
      } catch (EmployeeException e) {
         System.out.println(e.getMessage());
      }
```

```
        try {
            Employee emp2 = new Employee(2500);
        } catch (EmployeeException e) {
            System.out.println(e.getMessage());
        }
    }
}
```