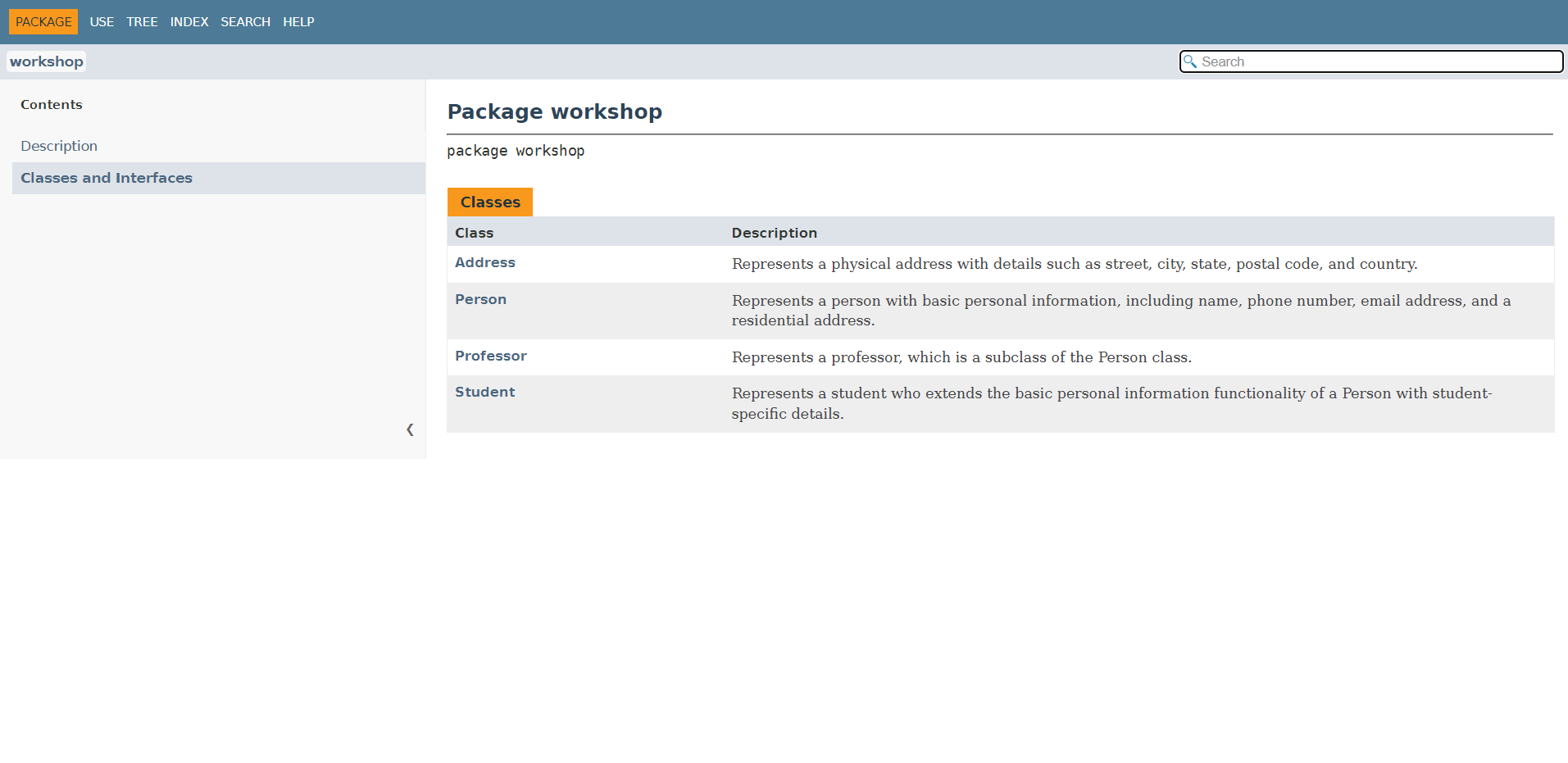
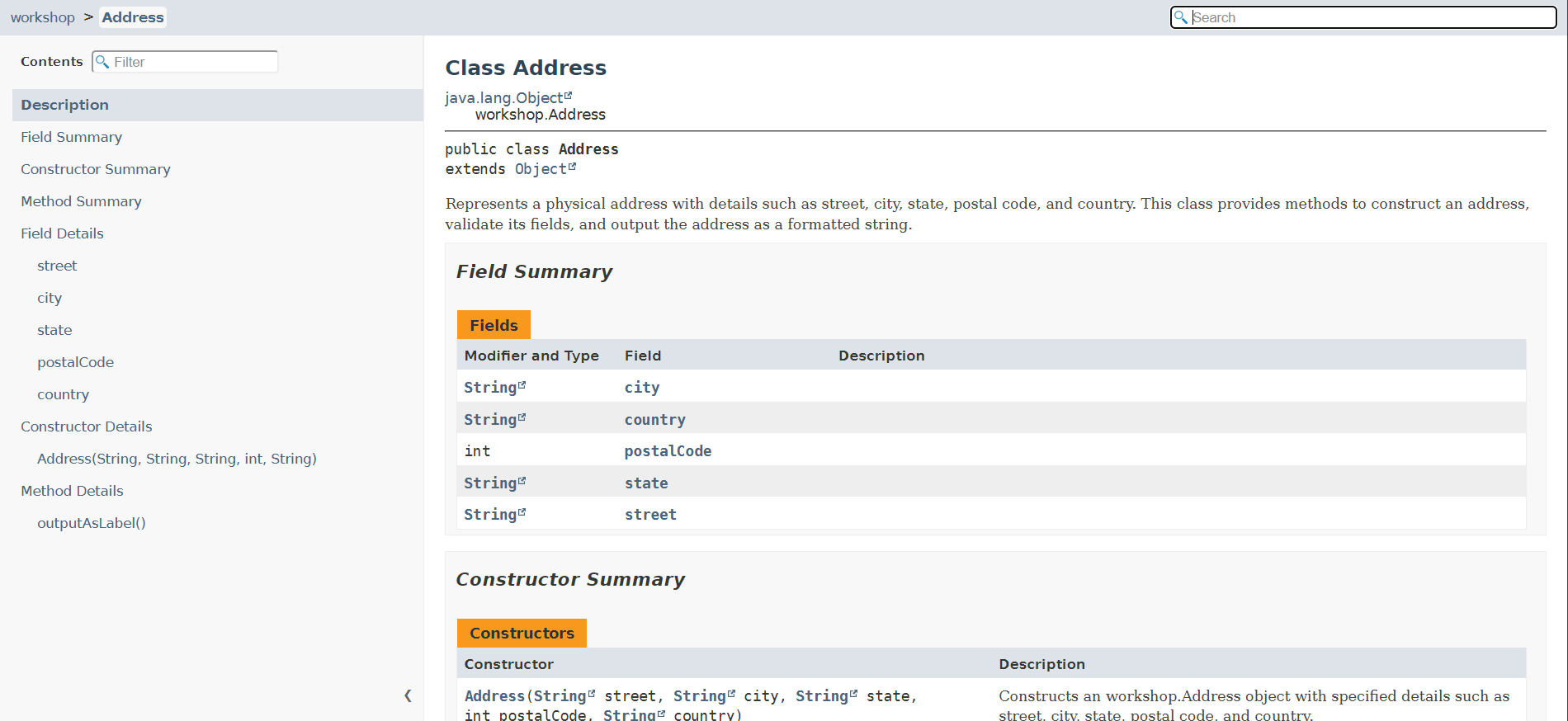
### **Question-1**

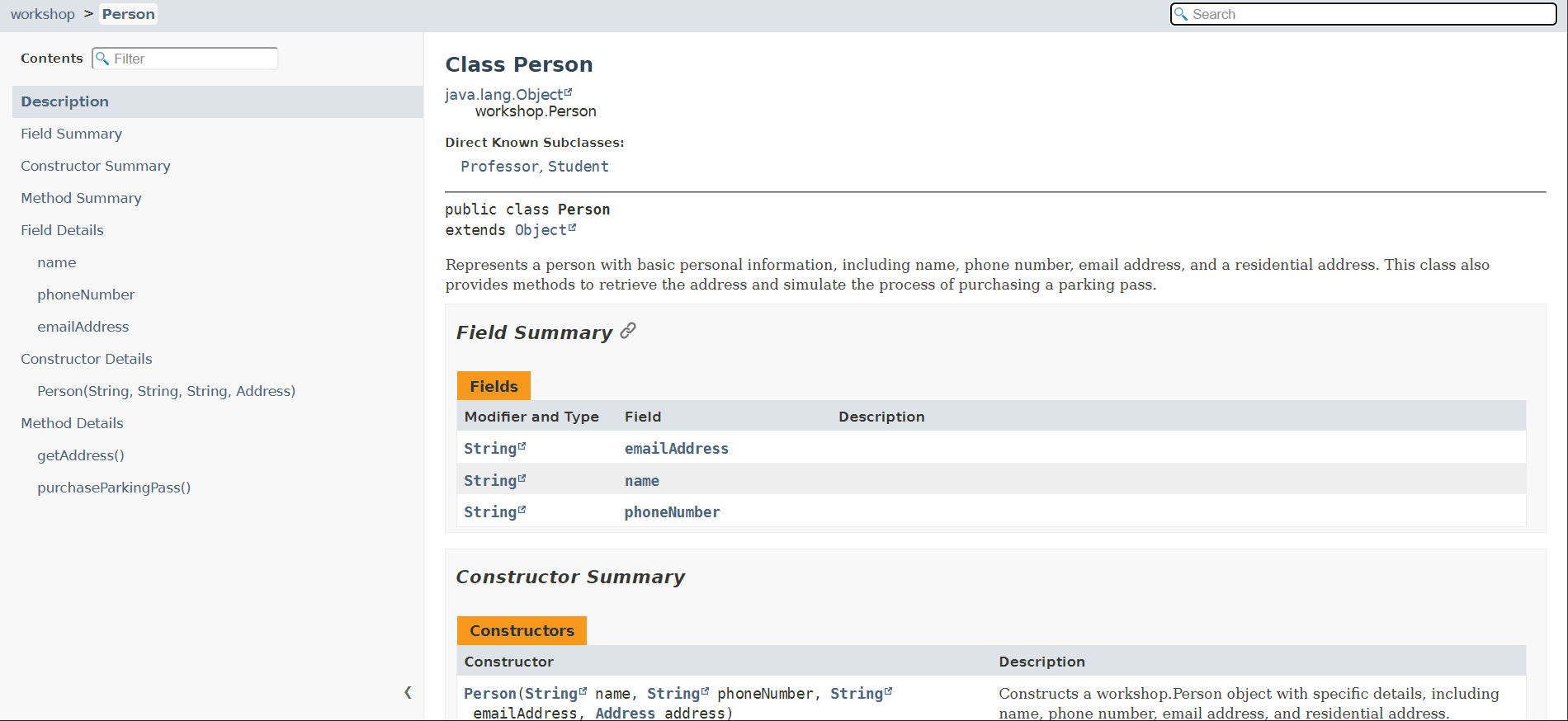
### Create classes and methods from the class diagram. Also, generate javadocs for them.

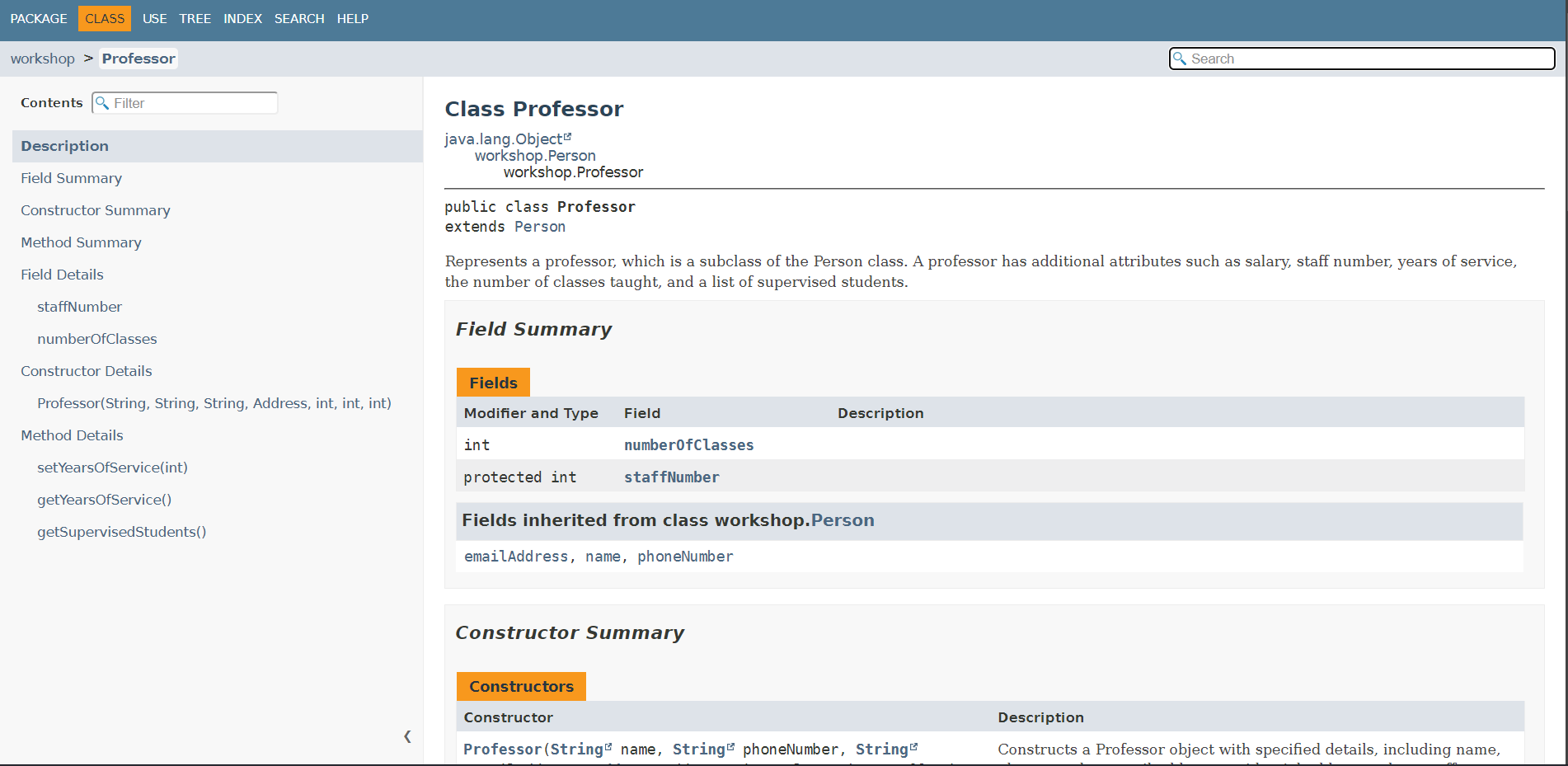
### 

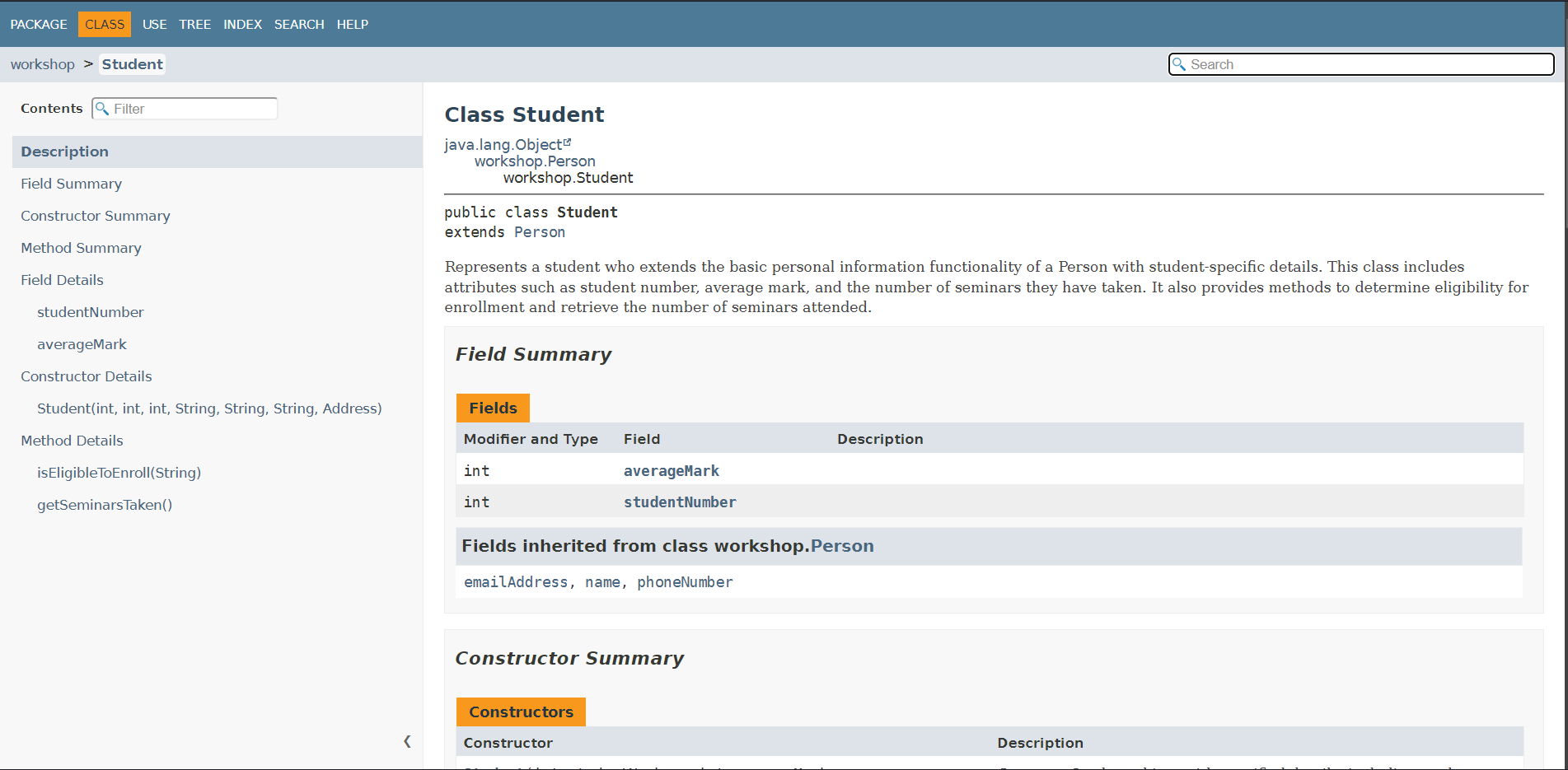
DOCS:











### 

### 

### **Case Study One: Hospital Management System**

A hospital needs a system to manage its operations, which include managing patients, doctors, appointments, and treatments. Patients can book appointments with doctors, and doctors can prescribe treatments.

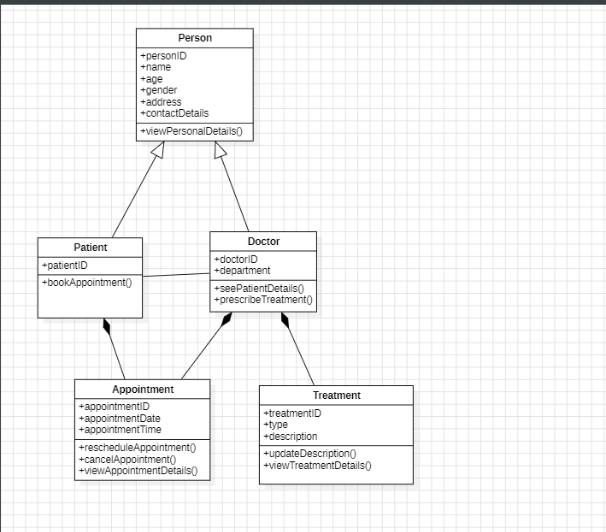
Design a **class diagram** representing the system and generate **Javadocs** for classes and methods. Also, implement the following functionalities:

### **Requirements:**

1. **Patients:** Patients can book appointments with doctors.
2. **Appointments:** Appointments record the date, time, and patient details.
3. **Doctors**: Doctors can prescribe treatments/medications for patients during an appointment.

### **Classes, Attributes and their Relationships:**

1. **Person(Parent Class)**
   * Attributes: personID, name, age, gender, address, contactDetails
   * Methods:viewPersonalDetails()
2. **Patient(Child Class)**
   * Attributes: patientID
   * Methods: bookAppointment()
3. **Doctor(Child Class)**
   * Attributes: doctorID, department
   * Methods: seePatientDetails(), prescribeTreatment()
4. **Appointment**
   * Attributes: appointmentID, appointmentDate, appointmentTime
   * Methods: rescheduleAppointment(), cancelAppointment(), viewAppointmentDetails()
5. **Treatment**
   * Attributes: treatmentID, type, description
   * Methods: updateDescription(), viewTreatmentDetails()



### **Case Study Two: E-Commerce Shopping Cart System**

You are developing a system that should allow customers to browse products, add products to a shopping cart, and proceed with the checkout.

Design a **class diagram** representing the system and generate **Javadocs** for classes and methods. Also, implement the following functionalities:

**Requirements:**

1. **Customer**: A customer can create an account, log in, and view products. A customer can also add products to their cart and proceed to checkout.
2. **Product**: The system should manage a list of products with details like name, price, and stock availability.
3. **Shopping Cart**: A customer can add products to the shopping cart, remove products, and view the total price of the cart. The cart should keep track of the quantities for each product.

### **Classes:**

1. **Customer**
   * Attributes: customerId, name, email, password
   * Methods: register(), login(), viewProducts(), addToCart(), checkout()
2. **Product**
   * Attributes: productId, name, price, stockQuantity
   * Methods: getDetails(), checkAvailability()
3. **ShoppingCart**
   * Attributes: cartItems (a list of products with their quantities), totalPrice
   * Methods: addProduct(), removeProduct(), calculateTotalPrice(), viewCartItems()

