University of Bordj Bou Arreridj
Faculty of Mathematics and Informatics

Computer Science Department
3rd year 2023/2024

Software Engineering

Practical work 3

Exercise 1: ***E-Commerce System Java Code Implementation***

***Objective*:** Write a Java code based on the provided class diagram and explanations.

***Instructions*:**

1. ***User Class:***
   - Create an abstract class ***User*** with attributes ***username*** and ***password***.
   - Implement an abstract constructor ***User(username: String, password: String)***.
   - Include public methods ***getUsername(): String*** and ***getPassword(): String***.
2. ***Customer Class:***
   - Extend the ***User*** class to create a concrete class ***Customer***.
   - Add private attributes ***name***, ***address***, ***email***, and ***phoneNumber***.
   - Implement a constructor ***Customer(username: String, password: String, name: String, address: String, email: String, phoneNumber: String)***.
   - Provide public getter and setter methods for the additional attributes.
3. ***Product Class:***
   - Create a class ***Product*** with private attributes ***name***, ***price***, and ***quantity***.
   - Implement a constructor ***Product(name: String, price: double, quantity: int)***.
   - Include public getter and setter methods for each attribute.
4. ***ShoppingCart Class:***
   - Create a class ***ShoppingCart*** with a private attribute ***products*** (a list of ***Product*** objects).
   - Implement a constructor ***ShoppingCart()***.
   - Include public methods ***addProduct(product: Product): void*** and ***removeProduct(product: Product): void***.

5. ***Order Class:***

    - Create a class ***Order*** with private attributes ***customer*** (a ***Customer*** object), ***products*** (a list of ***Product*** objects), ***totalPrice***, and ***status*** (an ***OrderStatus*** enum).
    - Implement a constructor ***Order(customer: Customer, products: List<Product>)***.
    - Include public methods ***calculateTotalPrice(): double*** and various getter and setter methods.

6. ***OrderStatus Enum:***

    - Define an enum ***OrderStatus*** with values: ***NEW***, ***PROCESSING***, ***SHIPPED***, ***DELIVERED***.

7. ***Relationships and Associations:***

    - Define relationships and associations between classes as explained in the class diagram.

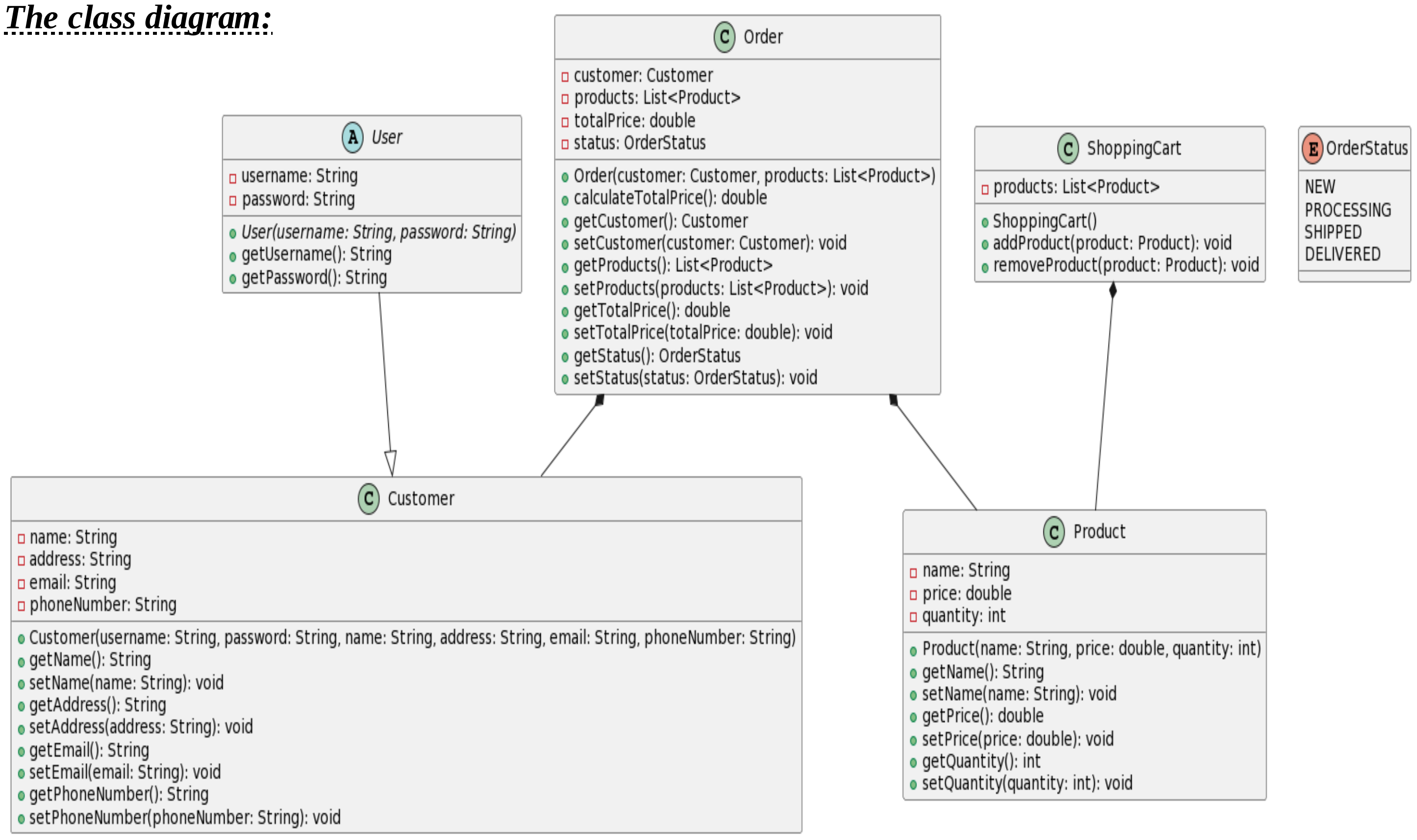
8. ***Summary:***

    - Summarize the purpose of each class and its relationships.
    - Ensure that each class and method has meaningful names and follows proper encapsulation principles.

Challenge (Optional):

- Implement additional methods and features based on your understanding of an e-commerce system.
- Consider adding error handling, validation, or database integration.

The class diagram:



Order
customer: Customer
products: List<Product>
totalPrice: double
status: OrderStatus
Order(customer: Customer, products: List<Product>)
calculateTotalPrice(): double
getCustomer(): Customer
setCustomer(customer: Customer): void
getProducts(): List<Product>
setProducts(products: List<Product>): void
getTotalPrice(): double
setTotalPrice(totalPrice: double): void
getStatus(): OrderStatus
setStatus(status: OrderStatus): void
User
username: String
password: String
User(username: String, password: String)
getUsername(): String
getPassword(): String
ShoppingCart
products: List<Product>
ShoppingCart()
addProduct(product: Product): void
removeProduct(product: Product): void
OrderStatus
NEW
PROCESSING
SHIPPED
DELIVERED
Customer
name: String
address: String
email: String
phoneNumber: String
Customer(username: String, password: String, name: String, address: String, email: String, phoneNumber: String)
getName(): String
setName(name: String): void
getAddress(): String
setAddress(address: String): void
getEmail(): String
setEmail(email: String): void
getPhoneNumber(): String
setPhoneNumber(phoneNumber: String): void
Product
name: String
price: double
quantity: int
Product(name: String, price: double, quantity: int)
getName(): String
setName(name: String): void
getPrice(): double
setPrice(price: double): void
getQuantity(): int
setQuantity(quantity: int): void

Exercise 1: *Medical Appointment System Java Implementation*

Objective: Write a Java code based on the provided class diagram and explanations.

Instructions:

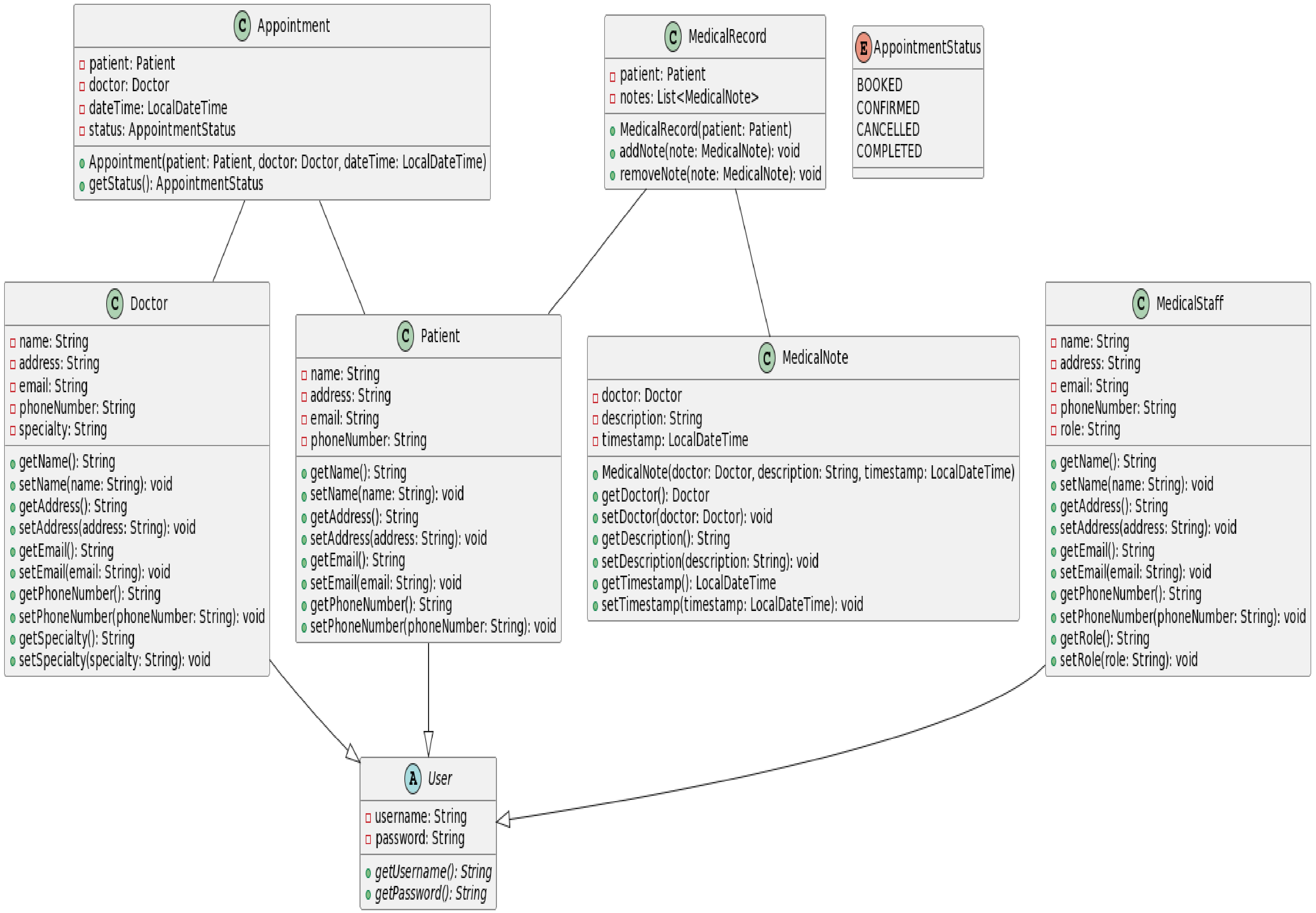
1. ***User Class:***
   - Create an abstract class ***User*** with attributes ***username*** and ***password***.
   - Implement an abstract constructor ***User(username: String, password: String)***.
   - Include public methods ***getUsername(): String and getPassword(): String***.
2. ***Patient Class:***
   - Extend the ***User*** class to create a concrete class ***Patient***.
   - Add private attributes ***name***, ***address***, ***email***, and ***phoneNumber***.
   - Implement a constructor ***Patient(username: String, password: String, name: String, address: String, email: String, phoneNumber: String)***.
   - Provide public getter and setter methods for the additional attributes.
3. ***Doctor Class:***
   - Extend the ***User*** class to create a concrete class ***Doctor***.
   - Add private attributes ***name***, ***address***, ***email***, ***phoneNumber***, and ***specialty***.
   - Implement a constructor ***Doctor(username: String, password: String, name: String, address: String, email: String, phoneNumber: String, specialty: String)***.
   - Provide public getter and setter methods for the additional attributes.
4. ***MedicalStaff Class:***
   - Extend the ***User*** class to create a concrete class ***MedicalStaff***.
   - Add private attributes ***name***, ***address***, ***email***, ***phoneNumber***, and ***role***.
   - Implement a constructor ***MedicalStaff(username: String, password: String, name: String, address: String, email: String, phoneNumber: String, role: String)***.
   - Provide public getter and setter methods for the additional attributes.
5. ***Appointment Class:***
   - Create a class ***Appointment*** with private attributes ***patient*** (a ***Patient*** object), ***doctor*** (a ***Doctor*** object), ***dateTime*** (a ***LocalDateTime*** object), and ***status*** (an ***AppointmentStatus*** enum).

- Implement a constructor ***Appointment(patient: Patient, doctor: Doctor, dateTime: LocalDateTime)***.
- Include public methods ***getStatus(): AppointmentStatus*** and various getter and setter methods.

6. ***AppointmentStatus Enum:***
   - Define an enum ***AppointmentStatus*** with values: ***BOOKED***, ***CONFIRMED***, ***CANCELLED***, ***COMPLETED***.
7. ***MedicalRecord Class:***
   - Create a class ***MedicalRecord*** with private attributes ***patient*** (a ***Patient*** object) and ***notes*** (a list of ***MedicalNote*** objects).
   - Implement a constructor ***MedicalRecord(patient: Patient)***.
   - Include public methods ***addNote(note: MedicalNote): void*** and ***removeNote(note: MedicalNote): void***.
8. ***MedicalNote Class:***
   - Create a class ***MedicalNote*** with private attributes ***doctor*** (a ***Doctor*** object), ***description***, and ***timestamp*** (a ***LocalDateTime*** object).
   - Implement a constructor ***MedicalNote(doctor: Doctor, description: String, timestamp: LocalDateTime)***.
   - Include public getter and setter methods for the attributes.
9. ***Relationships and Associations:***
   - Define relationships and associations between classes as explained in the class diagram.
10. ***Summary:***
    - Summarize the purpose of each class and its relationships.
    - Ensure that each class and method has meaningful names and follows proper encapsulation principles.

Challenge (Optional):

- Implement additional methods and features based on your understanding of a medical appointment system.
- Consider adding error handling, validation, or database integration.


C Appointment
patient: Patient
doctor: Doctor
dateTime: LocalDateTime
status: AppointmentStatus
Appointment(patient: Patient, doctor: Doctor, dateTime: LocalDateTime)
getStatus(): AppointmentStatus
C MedicalRecord
patient: Patient
notes: List<MedicalNote>
MedicalRecord(patient: Patient)
addNote(note: MedicalNote): void
removeNote(note: MedicalNote): void
E AppointmentStatus
BOOKED
CONFIRMED
CANCELLED
COMPLETED
C Doctor
name: String
address: String
email: String
phoneNumber: String
specialty: String
getName(): String
setName(name: String): void
getAddress(): String
setAddress(address: String): void
getEmail(): String
setEmail(email: String): void
getPhoneNumber(): String
setPhoneNumber(phoneNumber: String): void
getSpecialty(): String
setSpecialty(specialty: String): void
C Patient
name: String
address: String
email: String
phoneNumber: String
getName(): String
setName(name: String): void
getAddress(): String
setAddress(address: String): void
getEmail(): String
setEmail(email: String): void
getPhoneNumber(): String
setPhoneNumber(phoneNumber: String): void
C MedicalNote
doctor: Doctor
description: String
timestamp: LocalDateTime
MedicalNote(doctor: Doctor, description: String, timestamp: LocalDateTime)
getDoctor(): Doctor
setDoctor(doctor: Doctor): void
getDescription(): String
setDescription(description: String): void
getTimestamp(): LocalDateTime
setTimestamp(timestamp: LocalDateTime): void
C MedicalStaff
name: String
address: String
email: String
phoneNumber: String
role: String
getName(): String
setName(name: String): void
getAddress(): String
setAddress(address: String): void
getEmail(): String
setEmail(email: String): void
getPhoneNumber(): String
setPhoneNumber(phoneNumber: String): void
getRole(): String
setRole(role: String): void
A User
username: String
password: String
getUsername(): String
getPassword(): String