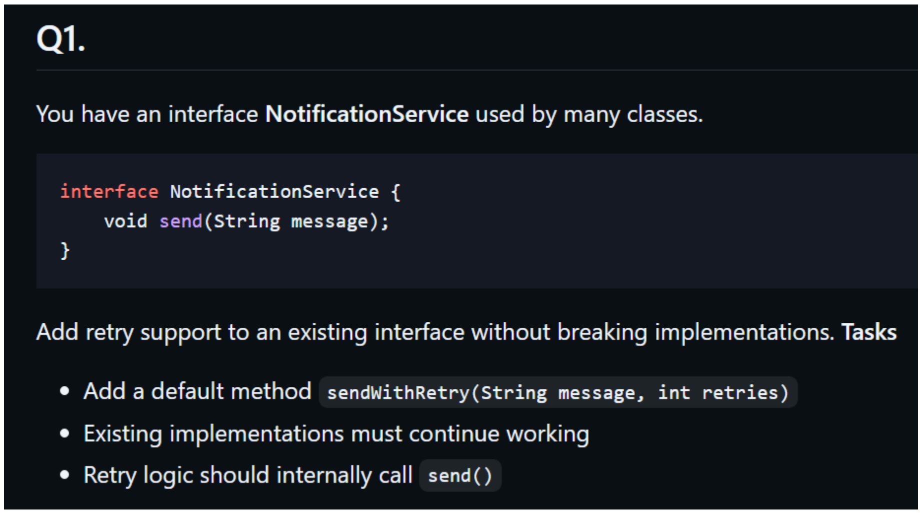
Workshop Assignment



Q1.

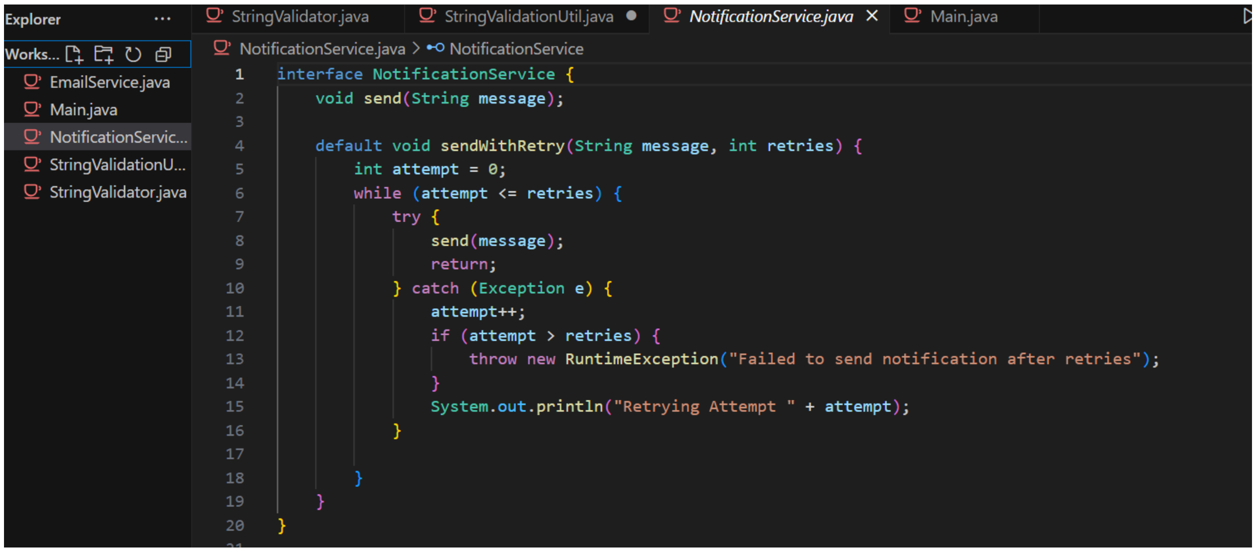
You have an interface **NotificationService** used by many classes.

```
interface NotificationService {
    void send(String message);
}
```

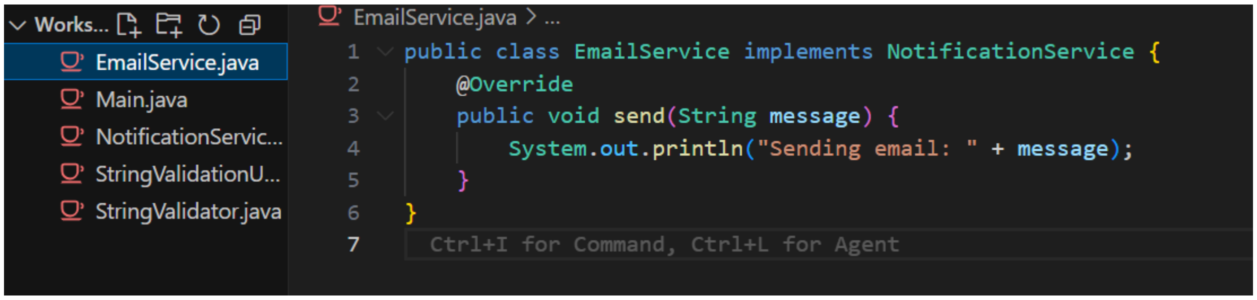
Add retry support to an existing interface without breaking implementations. **Tasks**

- Add a default method `sendWithRetry(String message, int retries)`
- Existing implementations must continue working
- Retry logic should internally call `send()`

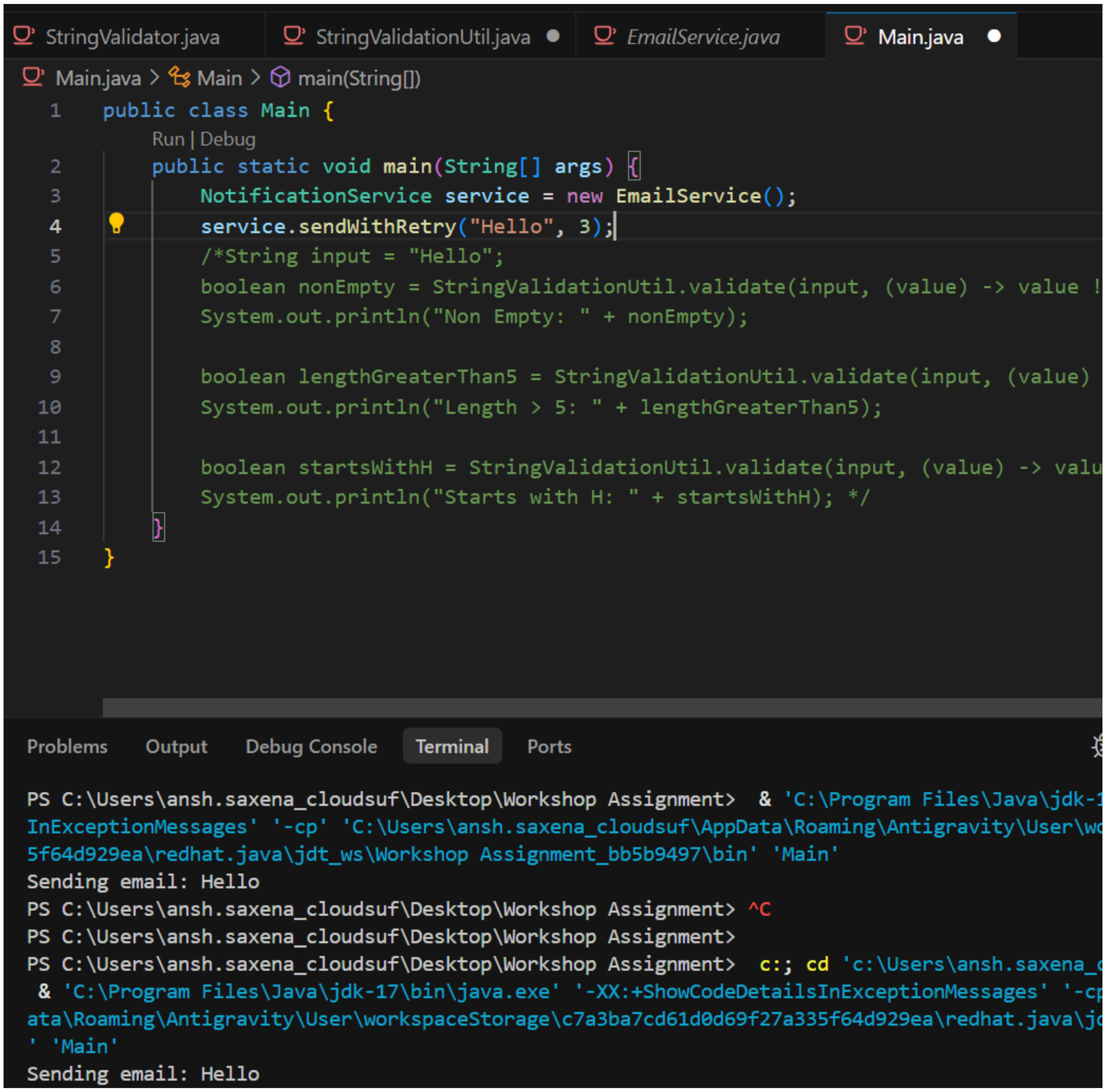
Sol-



```
interface NotificationService {
    void send(String message);

    default void sendWithRetry(String message, int retries) {
        int attempt = 0;
        while (attempt <= retries) {
            try {
                send(message);
                return;
            } catch (Exception e) {
                attempt++;
                if (attempt > retries) {
                    throw new RuntimeException("Failed to send notification after retries");
                }
                System.out.println("Retrying Attempt " + attempt);
            }

        }
    }
}
```



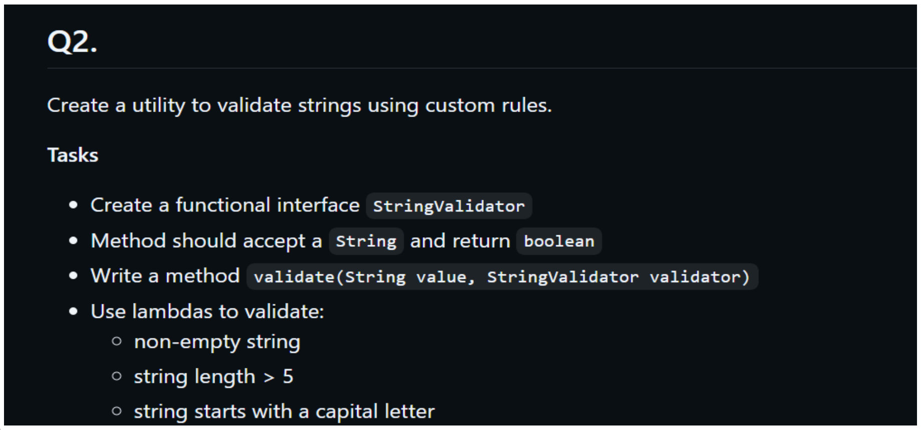
```
public class EmailService implements NotificationService {
    @Override
    public void send(String message) {
        System.out.println("Sending email: " + message);
    }
}
```



```java
public class Main {
    public static void main(String[] args) {
        NotificationService service = new EmailService();
        service.sendWithRetry("Hello", 3);
        /*String input = "Hello";
        boolean nonEmpty = StringValidationUtil.validate(input, (value) -> value !
        System.out.println("Non Empty: " + nonEmpty);

        boolean lengthGreaterThan5 = StringValidationUtil.validate(input, (value)
        System.out.println("Length > 5: " + lengthGreaterThan5);

        boolean startsWithH = StringValidationUtil.validate(input, (value) -> valu
        System.out.println("Starts with H: " + startsWithH); */
    }
}
```

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  & 'C:\Program Files\Java\jdk-1
InExceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\AppData\Roaming\Antigravity\User\wo
5f64d929ea\redhat.java\jdt_ws\Workshop Assignment_bb5b9497\bin' 'Main'
Sending email: Hello
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> ^C
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  c:; cd 'c:\Users\ansh.saxena_c
 & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp
ata\Roaming\Antigravity\User\workspaceStorage\c7a3ba7cd61d0d69f27a335f64d929ea\redhat.java\jd
' 'Main'
Sending email: Hello
```

Q2



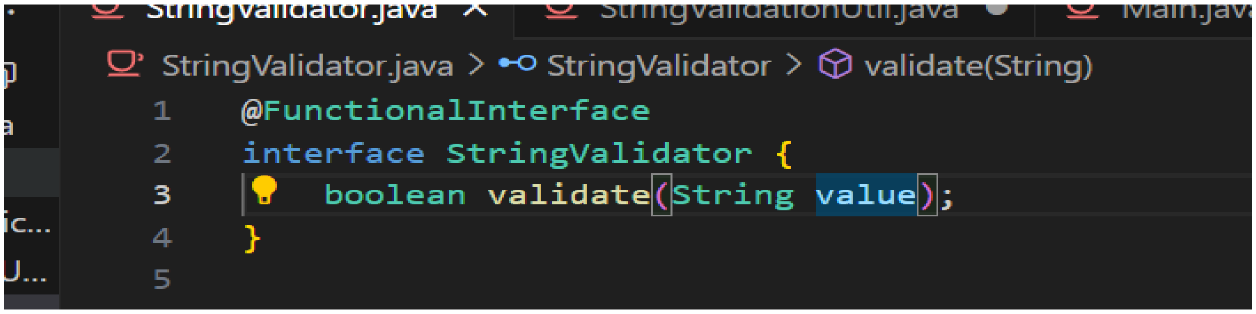
Q2.

Create a utility to validate strings using custom rules.

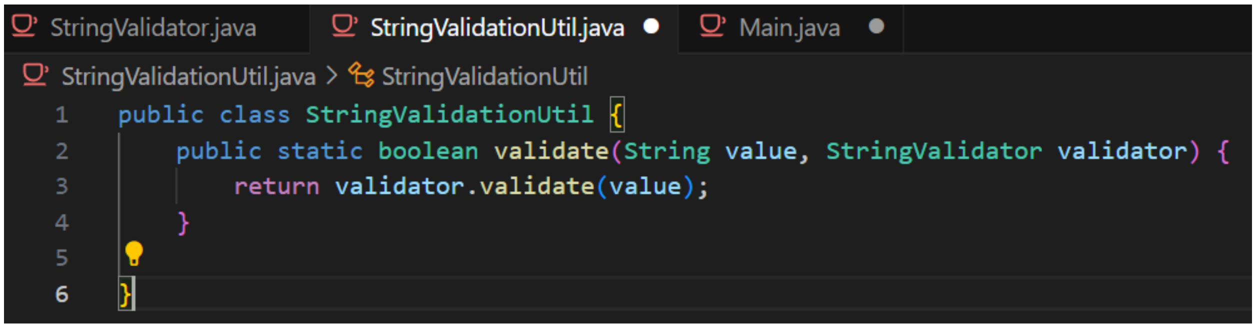
Tasks

- Create a functional interface `StringValidator`
- Method should accept a `String` and return `boolean`
- Write a method `validate(String value, StringValidator validator)`
- Use lambdas to validate:
  - non-empty string
  - string length > 5
  - string starts with a capital letter

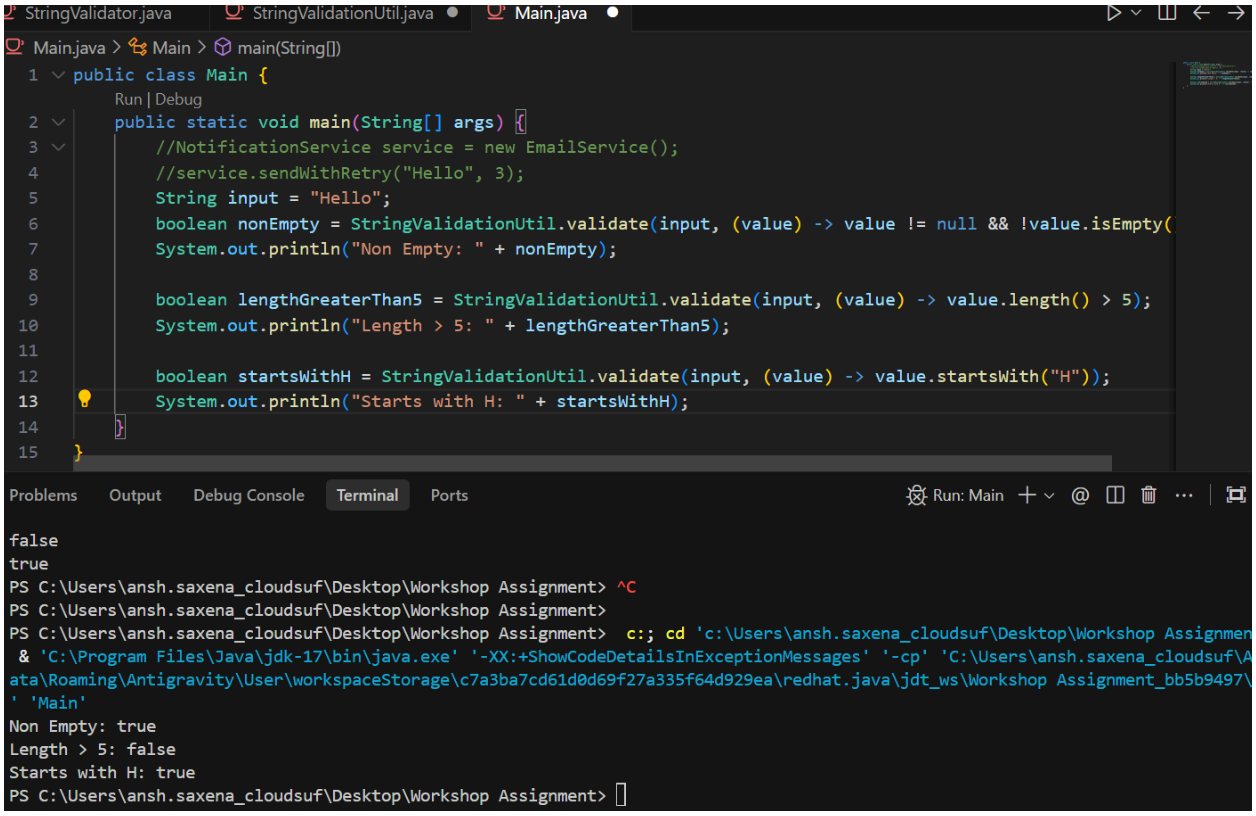
Sol-


StringValidator.java
StringValidationUtil.java
Main.java
StringValidator.java > StringValidator > validate(String)

```java
@FunctionalInterface
interface StringValidator {
    boolean validate(String value);
}

```


StringValidator.java
StringValidationUtil.java
Main.java
StringValidationUtil.java > StringValidationUtil

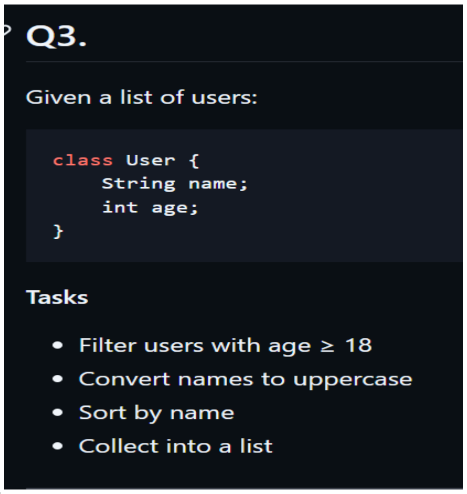
```java
public class StringValidationUtil {
    public static boolean validate(String value, StringValidator validator) {
        return validator.validate(value);
    }

}
```



```java
public class Main {
    public static void main(String[] args) {
        //NotificationService service = new EmailService();
        //service.sendWithRetry("Hello", 3);
        String input = "Hello";
        boolean nonEmpty = StringValidationUtil.validate(input, (value) -> value != null && !value.isEmpty(
        System.out.println("Non Empty: " + nonEmpty);

        boolean lengthGreaterThan5 = StringValidationUtil.validate(input, (value) -> value.length() > 5);
        System.out.println("Length > 5: " + lengthGreaterThan5);

        boolean startsWithH = StringValidationUtil.validate(input, (value) -> value.startsWith("H"));
        System.out.println("Starts with H: " + startsWithH);
    }
}
```

```
false
true
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> ^C
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  c:; cd 'c:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignmen
 & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\A
ata\Roaming\Antigravity\User\workspaceStorage\c7a3ba7cd61d0d69f27a335f64d929ea\redhat.java\jdt_ws\Workshop Assignment_bb5b9497\
' 'Main'
Non Empty: true
Length > 5: false
Starts with H: true
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> 
```

Q3



Q3.

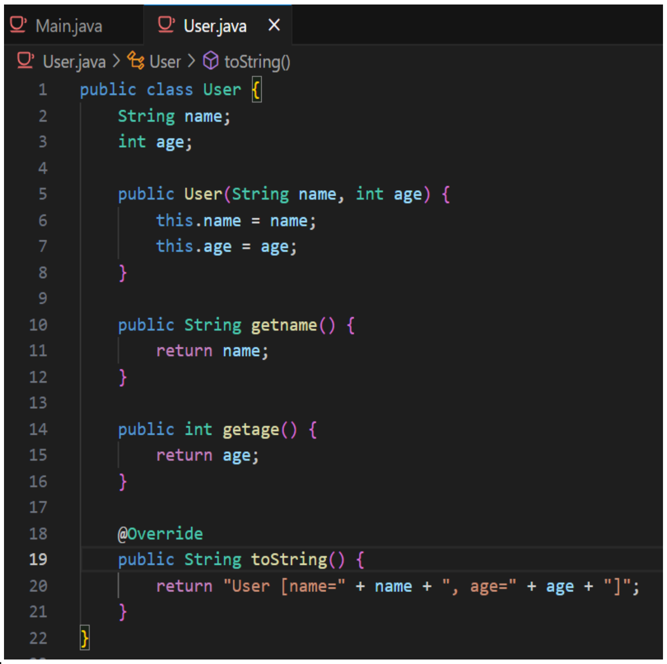
Given a list of users:

```
class User {
    String name;
    int age;
}
```

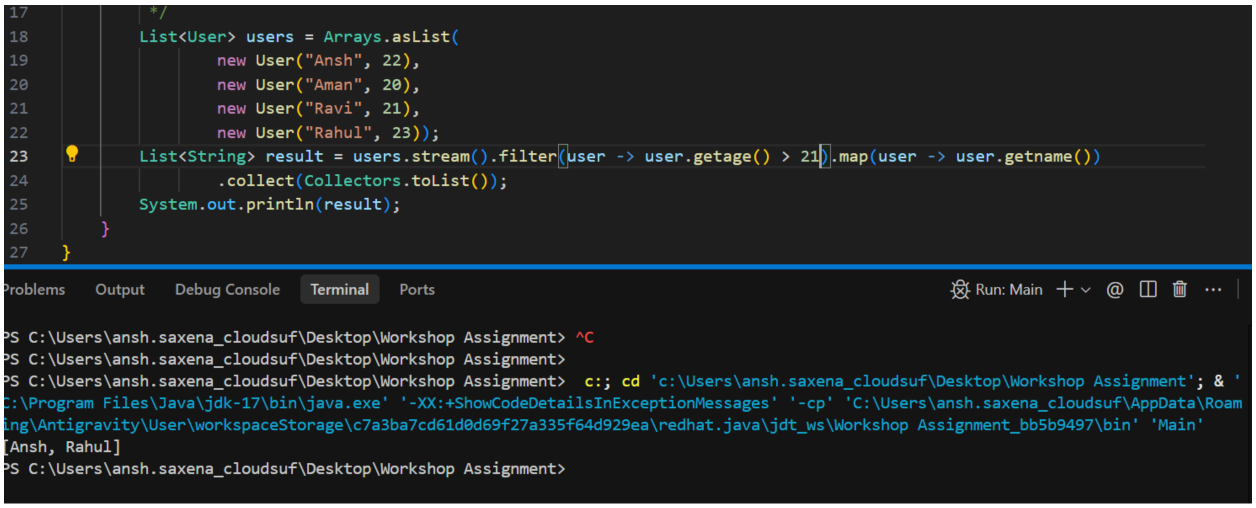
Tasks

- Filter users with age ≥ 18
- Convert names to uppercase
- Sort by name
- Collect into a list

Sol-

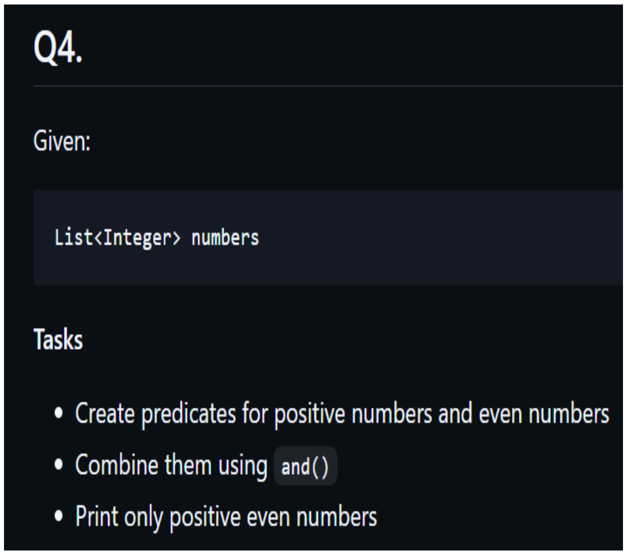


```java
public class User {
    String name;
    int age;

    public User(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getname() {
        return name;
    }

    public int getage() {
        return age;
    }

    @Override
    public String toString() {
        return "User [name=" + name + ", age=" + age + "]";
    }
}
```



```java
        */
        List<User> users = Arrays.asList(
                new User("Ansh", 22),
                new User("Aman", 20),
                new User("Ravi", 21),
                new User("Rahul", 23));
        List<String> result = users.stream().filter(user -> user.getage() > 21).map(user -> user.getname())
                .collect(Collectors.toList());
        System.out.println(result);
    }
}
```

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> ^C
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  c:; cd 'c:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment'; & '
C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\AppData\Roam
ing\Antigravity\User\workspaceStorage\c7a3ba7cd61d0d69f27a335f64d929ea\redhat.java\jdt_ws\Workshop Assignment_bb5b9497\bin' 'Main'
[Ansh, Rahul]
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
```



Q4.

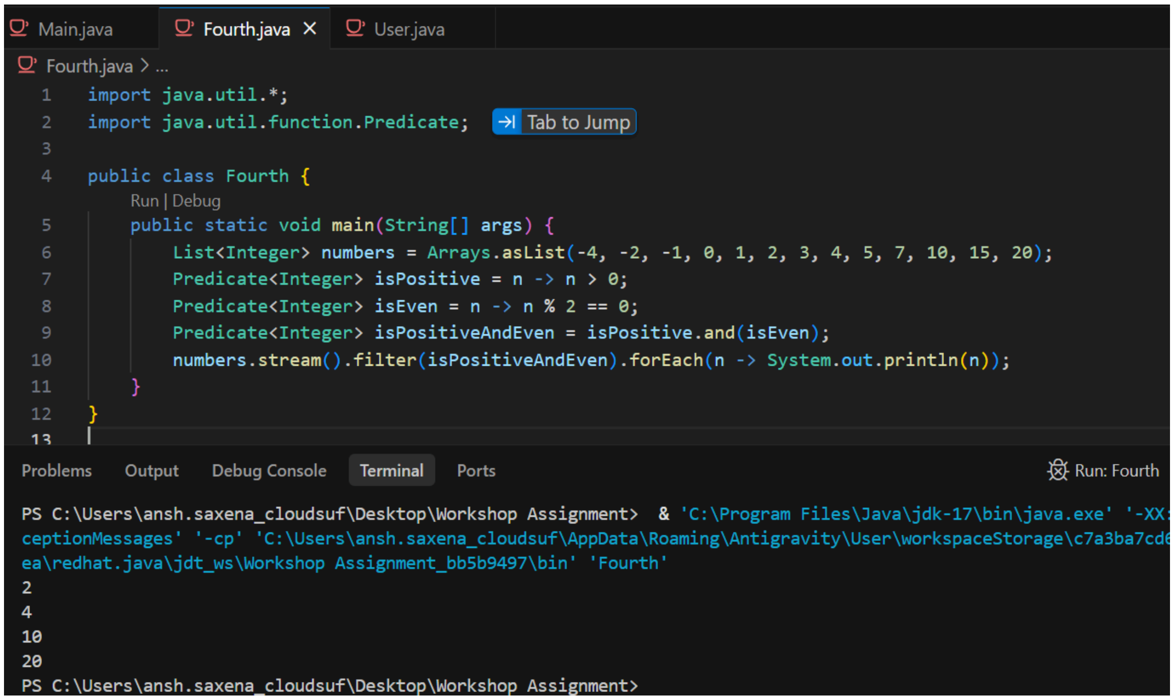
Given:

```
List<Integer> numbers
```

Tasks

- Create predicates for positive numbers and even numbers
- Combine them using `and()`
- Print only positive even numbers

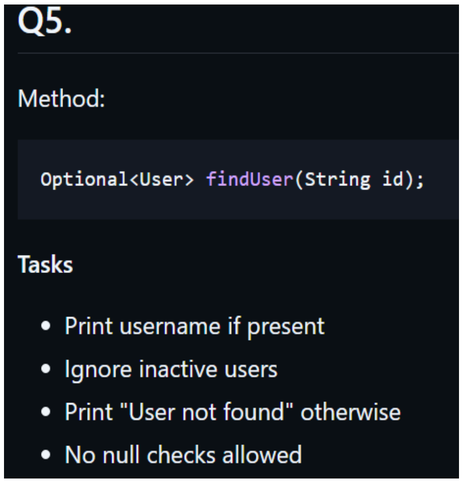
<u>Q4 Sol-</u>



```java
import java.util.*;
import java.util.function.Predicate;

public class Fourth {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(-4, -2, -1, 0, 1, 2, 3, 4, 5, 7, 10, 15, 20);
        Predicate<Integer> isPositive = n -> n > 0;
        Predicate<Integer> isEven = n -> n % 2 == 0;
        Predicate<Integer> isPositiveAndEven = isPositive.and(isEven);
        numbers.stream().filter(isPositiveAndEven).forEach(n -> System.out.println(n));
    }
}
```

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:
ceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\AppData\Roaming\Antigravity\User\workspaceStorage\c7a3ba7cd
ea\redhat.java\jdt_ws\Workshop Assignment_bb5b9497\bin' 'Fourth'
2
4
10
20
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
```

Q5-



Q5.

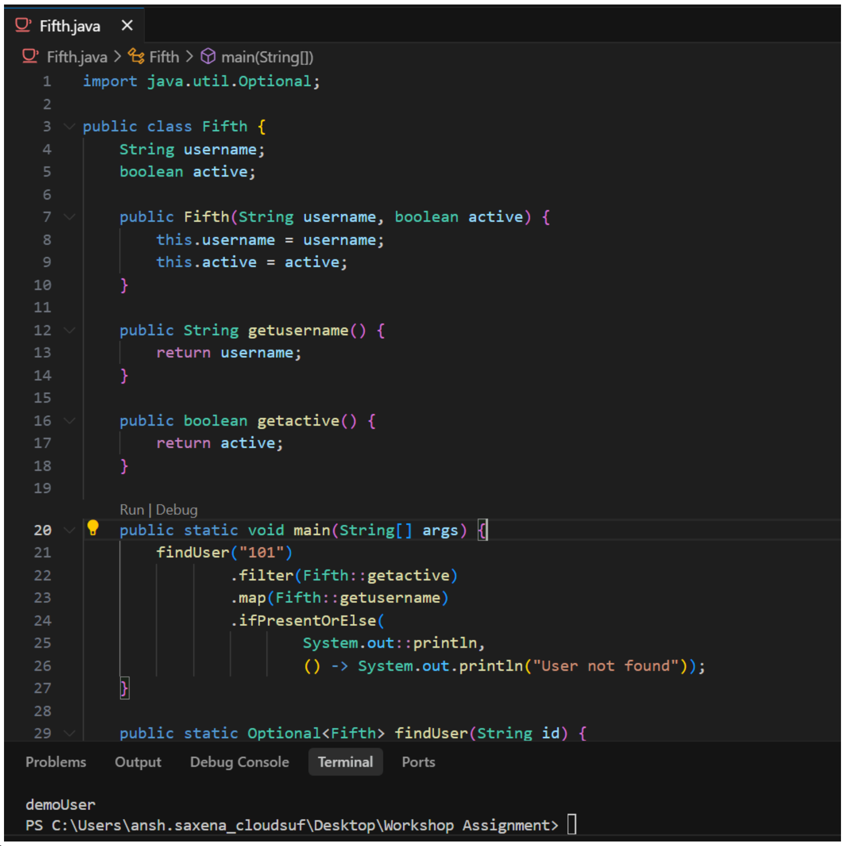
Method:

```
Optional<User> findUser(String id);
```

Tasks

- Print username if present
- Ignore inactive users
- Print "User not found" otherwise
- No null checks allowed

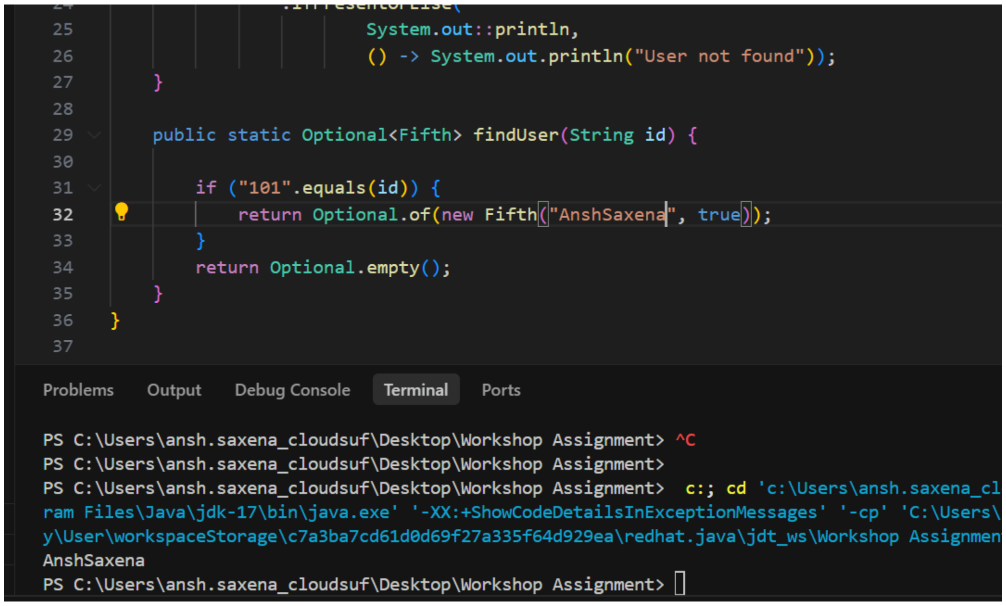
Sol-



Fifth.java
Fifth.java > Fifth > main(String[])
Run | Debug
Problems Output Debug Console Terminal Ports

```
import java.util.Optional;

public class Fifth {
    String username;
    boolean active;

    public Fifth(String username, boolean active) {
        this.username = username;
        this.active = active;
    }

    public String getusername() {
        return username;
    }

    public boolean getactive() {
        return active;
    }

    public static void main(String[] args) {
        findUser("101")
                .filter(Fifth::getactive)
                .map(Fifth::getusername)
                .ifPresentOrElse(
                        System.out::println,
                        () -> System.out.println("User not found"));
    }

    public static Optional<Fifth> findUser(String id) {
```

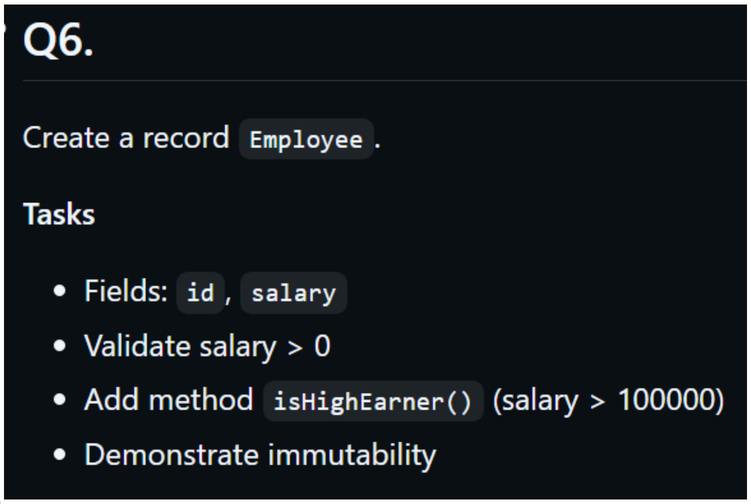
```
demoUser
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
```



```
                .ifPresentOrElse(
                        System.out::println,
                        () -> System.out.println("User not found"));
    }

    public static Optional<Fifth> findUser(String id) {

        if ("101".equals(id)) {
            return Optional.of(new Fifth("AnshSaxena", true));
        }
        return Optional.empty();
    }
}

```

Problems Output Debug Console Terminal Ports

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> ^C
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  c:; cd 'c:\Users\ansh.saxena_cl
ram Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
y\User\workspaceStorage\c7a3ba7cd61d0d69f27a335f64d929ea\redhat.java\jdt_ws\Workshop Assignmen
AnshSaxena
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> 
```



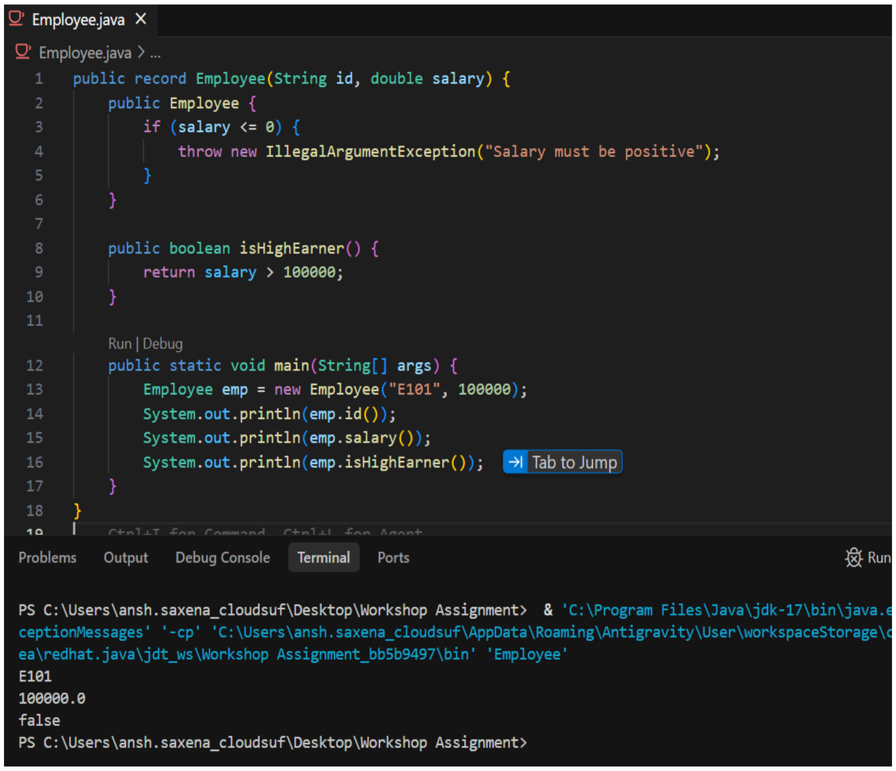
Q6.

Create a record `Employee`.

Tasks

- Fields: `id`, `salary`
- Validate salary > 0
- Add method `isHighEarner()` (salary > 100000)
- Demonstrate immutability

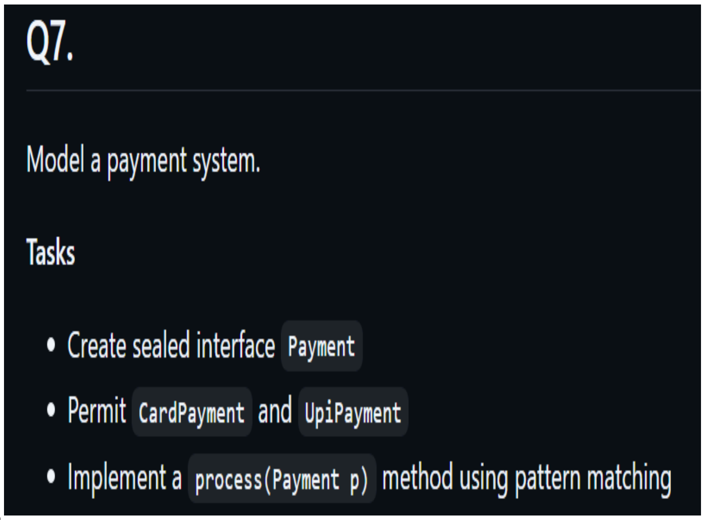
Q6



```java
public record Employee(String id, double salary) {
    public Employee {
        if (salary <= 0) {
            throw new IllegalArgumentException("Salary must be positive");
        }
    }

    public boolean isHighEarner() {
        return salary > 100000;
    }

    public static void main(String[] args) {
        Employee emp = new Employee("E101", 100000);
        System.out.println(emp.id());
        System.out.println(emp.salary());
        System.out.println(emp.isHighEarner());
    }
}
```

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  & 'C:\Program Files\Java\jdk-17\bin\java.e
ceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\AppData\Roaming\Antigravity\User\workspaceStorage\c
ea\redhat.java\jdt_ws\Workshop Assignment_bb5b9497\bin' 'Employee'
E101
100000.0
false
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
```

Q7-

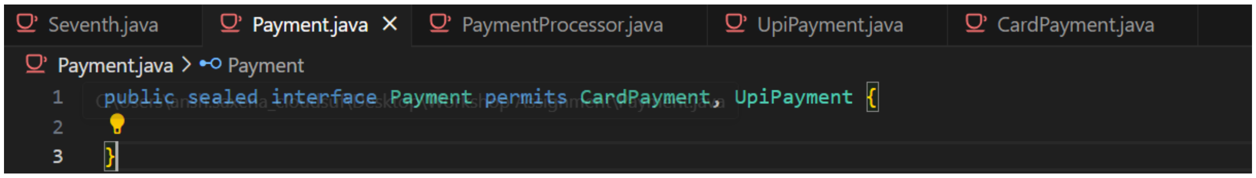


Q7.

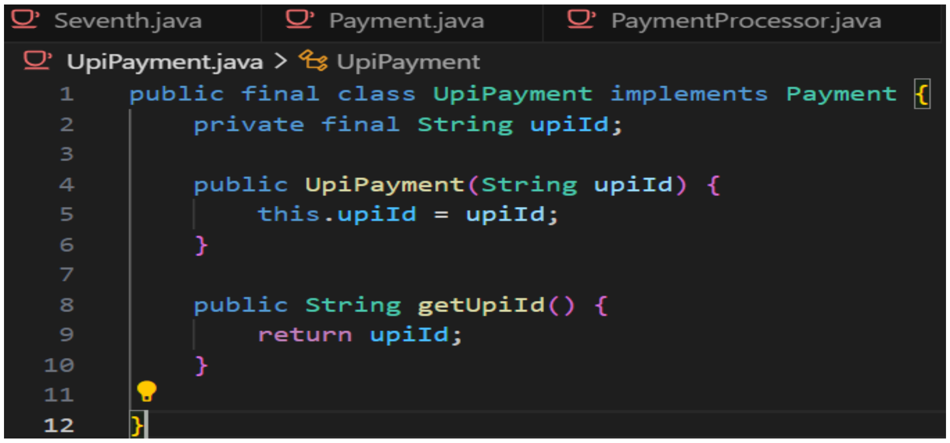
Model a payment system.

Tasks

- Create sealed interface `Payment`
- Permit `CardPayment` and `UpiPayment`
- Implement a `process(Payment p)` method using pattern matching

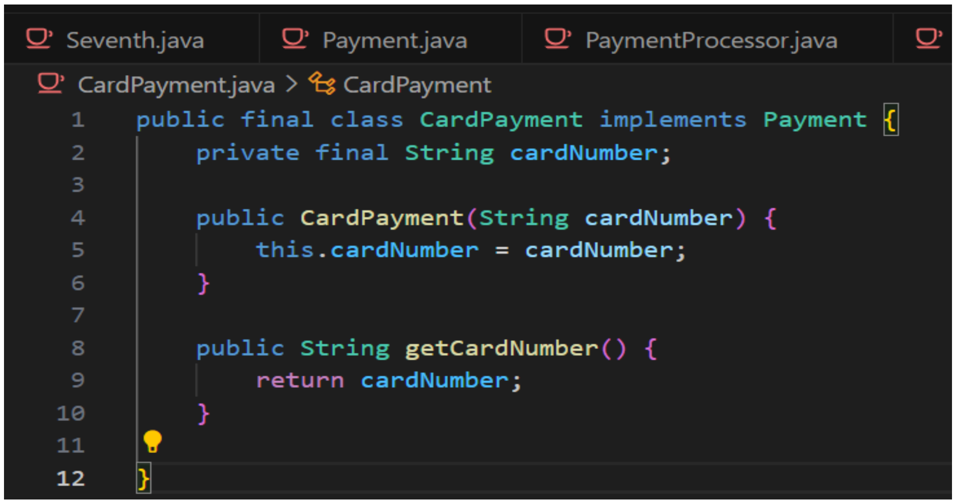


```java
public sealed interface Payment permits CardPayment, UpiPayment {

}
```


Seventh.java   Payment.java   PaymentProcessor.java

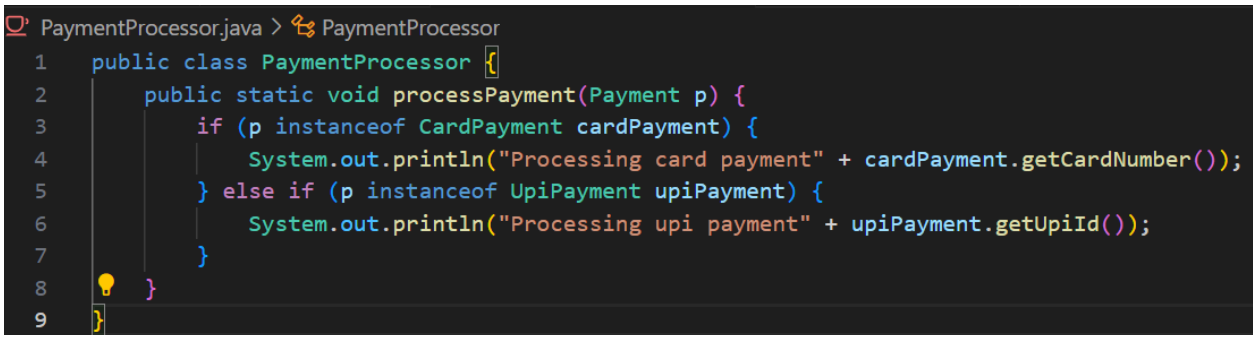
UpiPayment.java > UpiPayment

```java
public final class UpiPayment implements Payment {
    private final String upiId;

    public UpiPayment(String upiId) {
        this.upiId = upiId;
    }

    public String getUpiId() {
        return upiId;
    }

}
```

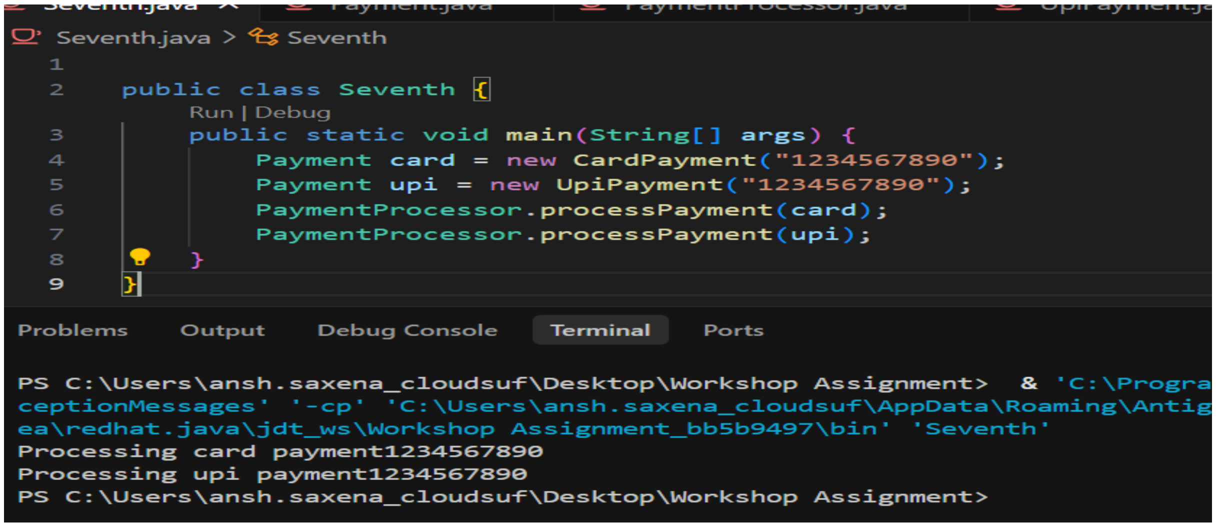

Seventh.java   Payment.java   PaymentProcessor.java

CardPayment.java > CardPayment

```java
public final class CardPayment implements Payment {
    private final String cardNumber;

    public CardPayment(String cardNumber) {
        this.cardNumber = cardNumber;
    }

    public String getCardNumber() {
        return cardNumber;
    }

}
```

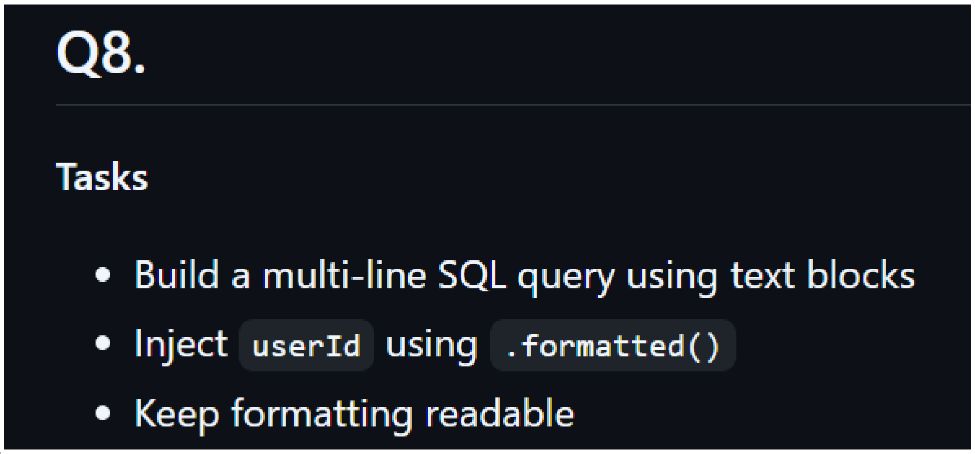

PaymentProcessor.java > PaymentProcessor

```java
public class PaymentProcessor {
    public static void processPayment(Payment p) {
        if (p instanceof CardPayment cardPayment) {
            System.out.println("Processing card payment" + cardPayment.getCardNumber());
        } else if (p instanceof UpiPayment upiPayment) {
            System.out.println("Processing upi payment" + upiPayment.getUpiId());
        }
    }
}
```



```java
public class Seventh {
    public static void main(String[] args) {
        Payment card = new CardPayment("1234567890");
        Payment upi = new UpiPayment("1234567890");
        PaymentProcessor.processPayment(card);
        PaymentProcessor.processPayment(upi);
    }
}
```

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  & 'C:\Progra
ceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\AppData\Roaming\Antig
ea\redhat.java\jdt_ws\Workshop Assignment_bb5b9497\bin' 'Seventh'
Processing card payment1234567890
Processing upi payment1234567890
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
```



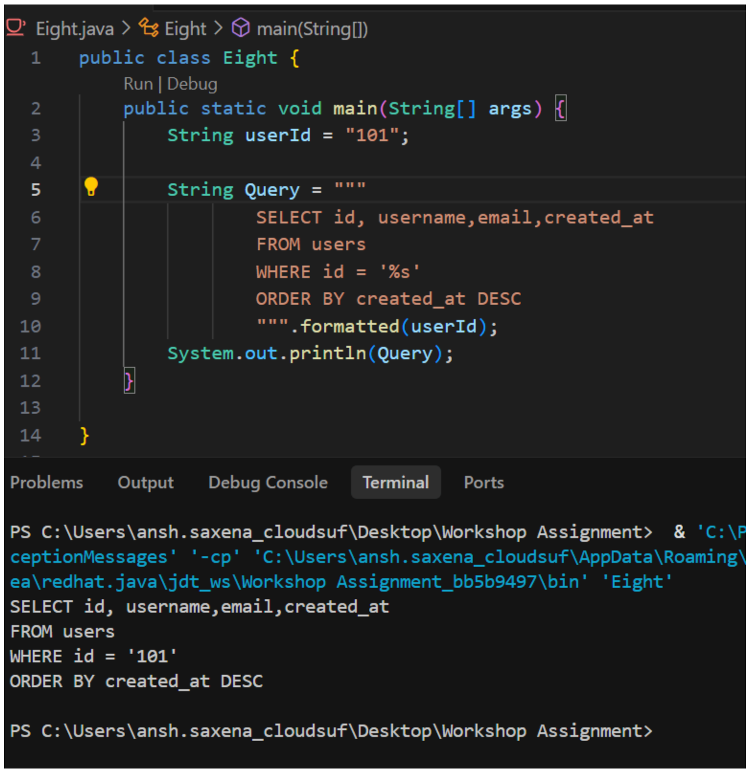
Q8.

Tasks

- Build a multi-line SQL query using text blocks
- Inject `userId` using `.formatted()`
- Keep formatting readable

Q8-

Sol-



```java
public class Eight {
    public static void main(String[] args) {
        String userId = "101";

        String Query = """
                SELECT id, username,email,created_at
                FROM users
                WHERE id = '%s'
                ORDER BY created_at DESC
                """.formatted(userId);
        System.out.println(Query);
    }

}
```

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>  & 'C:\P
ceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\AppData\Roaming\
ea\redhat.java\jdt_ws\Workshop Assignment_bb5b9497\bin' 'Eight'
SELECT id, username,email,created_at
FROM users
WHERE id = '101'
ORDER BY created_at DESC

PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
```