

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-

The screenshot shows a Java development environment with two code editors. The top editor displays the `NotificationService.java` interface, which includes a new default method `sendWithRetry`. The bottom editor displays the `EmailService.java` class, which implements the `NotificationService` interface and provides a concrete implementation for the `send` method.

```
NotificationService.java:  
1  interface NotificationService {  
2      void send(String message);  
3  
4      default void sendWithRetry(String message, int retries) {  
5          int attempt = 0;  
6          while (attempt <= retries) {  
7              try {  
8                  send(message);  
9                  return;  
10             } catch (Exception e) {  
11                 attempt++;  
12                 if (attempt > retries) {  
13                     throw new RuntimeException("Failed to send notification after retries");  
14                 }  
15                 System.out.println("Retrying Attempt " + attempt);  
16             }  
17         }  
18     }  
19 }  
20  
EmailService.java:  
1  public class EmailService implements NotificationService {  
2      @Override  
3      public void send(String message) {  
4          System.out.println("Sending email: " + message);  
5      }  
6  }  
7
```

The screenshot shows a Java IDE interface with several tabs at the top: StringValidator.java, StringValidationUtil.java, EmailService.java, Main.java (selected), and Main.java > main(String[]). The Main.java code is as follows:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         NotificationService service = new EmailService();
5         service.sendWithRetry("Hello", 3);
6         /*String input = "Hello";
7         boolean nonEmpty = StringValidationUtil.validate(input, (value) -> value != null);
8         System.out.println("Non Empty: " + nonEmpty);
9
10        boolean lengthGreaterThan5 = StringValidationUtil.validate(input, (value) -> value.length() > 5);
11        System.out.println("Length > 5: " + lengthGreaterThan5);
12
13        boolean startsWithH = StringValidationUtil.validate(input, (value) -> value.startsWith("H"));
14        System.out.println("Starts with H: " + startsWithH); */
15    }
}
```

Below the code editor is a terminal window showing command-line output:

```
PS 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\Roaming\Antigravity\User\workspaceStorage\c7a3ba7cd61d0d69f27a335f64d929ea\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_cloudsuf\Desktop\Workshop Assignment' & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment_bb5b9497\bin' 'Main'
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> Sending email: Hello
```

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

Q2

Sol-

The screenshot shows an IDE interface with three tabs open:

- StringValidator.java**: Contains a functional interface definition:

```
1  @FunctionalInterface
2  interface StringValidator {
3      boolean validate(String value);
4  }
```
- StringValidationUtil.java**: Contains a static method that delegates validation to another object:

```
1  public class StringValidationUtil {
2      public static boolean validate(String value, StringValidator validator) {
3          return validator.validate(value);
4      }
5  }
```
- Main.java**: Contains the main application logic:

```
1  public class Main {
2      public static void main(String[] args) {
3          //NotificationService service = new EmailService();
4          //service.sendWithRetry("Hello", 3);
5          String input = "Hello";
6          boolean nonEmpty = StringValidationUtil.validate(input, (value) -> value != null && !value.isEmpty());
7          System.out.println("Non Empty: " + nonEmpty);
8
9          boolean lengthGreaterThan5 = StringValidationUtil.validate(input, (value) -> value.length() > 5);
10         System.out.println("Length > 5: " + lengthGreaterThan5);
11
12         boolean startsWithH = StringValidationUtil.validate(input, (value) -> value.startsWith("H"));
13         System.out.println("Starts with H: " + startsWithH);
14     }
15 }
```

The terminal at the bottom shows the execution of the application:

```
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 Assignment' & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ansh.saxena_cloudsuf\Anata\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.

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

Q3

```
Main.java User.java X
User.java > User > toString()
1 public class User {
2     String name;
3     int age;
4
5     public User(String name, int age) {
6         this.name = name;
7         this.age = age;
8     }
9
10    public String getName() {
11        return name;
12    }
13
14    public int getAge() {
15        return age;
16    }
17
18    @Override
19    public String toString() {
20        return "User [name=" + name + ", age=" + age + "]";
21    }
22 }
```

Sol-

```
/*
18 List<User> users = Arrays.asList(
19     new User("Ansh", 22),
20     new User("Aman", 20),
21     new User("Ravi", 21),
22     new User("Rahul", 23));
23 List<String> result = users.stream().filter(user -> user.getAge() > 21).map(user -> user.getName())
24     .collect(Collectors.toList());
25 System.out.println(result);
26 }
27 }
```

Problems Output Debug Console Terminal Ports Run: Main + @ ... |

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

Q4

Sol-

The screenshot shows a Java IDE interface with the following details:

- Project Structure:** Shows files Main.java, Fourth.java (active), and User.java.
- Code Editor:** Displays the content of Fourth.java:

```
1 import java.util.*;
2 import java.util.function.Predicate; →I Tab to Jump
3
4 public class Fourth {
    Run | Debug
    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));
    }
}
```
- Toolbars and Menus:** Includes Problems, Output, Debug Console, Terminal, and Ports.
- Terminal:** Shows the command run and its output:

```
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\c7a3ba7cd6ea\redhat.java\jdt_ws\Workshop Assignment_bb5b9497\bin' 'Fourth'
2
4
10
20
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>
```

Q5.

Method:

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

Tasks

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

Q5-

The screenshot shows an IDE interface with a code editor and a terminal. The code editor displays a Java file named 'Fifth.java' containing the following code:

```
1  import java.util.Optional;
2
3  public class Fifth {
4      String username;
5      boolean active;
6
7      public Fifth(String username, boolean active) {
8          this.username = username;
9          this.active = active;
10     }
11
12     public String getUsername() {
13         return username;
14     }
15
16     public boolean getactive() {
17         return active;
18     }
19
20     public static void main(String[] args) {
21         findUser("101")
22             .filter(Fifth::getactive)
23             .map(Fifth::getusername)
24             .ifPresentOrElse(
25                 System.out::println,
26                 () -> System.out.println("User not found"));
27     }
28
29     public static Optional<Fifth> findUser(String id) {
```

The terminal below the code editor shows the command 'demoUser' and the path 'PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment>'. The terminal window is currently empty.

Sol-

```
24     .ifPresentOrElse(
25         System.out::println,
26         () -> System.out.println("User not found"));
27     }
28
29     public static Optional<Fifth> findUser(String id) {
30
31         if ("101".equals(id)) {
32             return Optional.of(new Fifth("AnshSaxena", true));
33         }
34         return Optional.empty();
35     }
36
37 }
```

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 Assignment
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

```

Employee.java X
Employee.java > ...
1  public record Employee(String id, double salary) {
2      public Employee {
3          if (salary <= 0) {
4              throw new IllegalArgumentException("Salary must be positive");
5          }
6      }
7
8      public boolean isHighEarner() {
9          return salary > 100000;
10     }
11
12     Run | Debug
13     public static void main(String[] args) {
14         Employee emp = new Employee("E101", 100000);
15         System.out.println(emp.id());
16         System.out.println(emp.salary());
17         System.out.println(emp.isHighEarner()); → Tab to Jump
18     }
19
20 }

```

Problems Output Debug Console Terminal Ports Run

```

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

```

Q7.

Model a payment system.

Tasks

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

Q7-

```

Seventh.java Payment.java X PaymentProcessor.java UpiPayment.java CardPayment.java
Payment.java > Payment
1  public sealed interface Payment permits CardPayment, UpiPayment {
2
3 }

```

Seventh.java | Payment.java | PaymentProcessor.java

UpIPayment.java > UpIPayment

```
1 public final class UpIPayment implements Payment {
2     private final String upiId;
3
4     public UpIPayment(String upiId) {
5         this.upiId = upiId;
6     }
7
8     public String getUpiId() {
9         return upiId;
10    }
11
12 }
```

Seventh.java | Payment.java | PaymentProcessor.java |

CardPayment.java > CardPayment

```
1 public final class CardPayment implements Payment {
2     private final String cardNumber;
3
4     public CardPayment(String cardNumber) {
5         this.cardNumber = cardNumber;
6     }
7
8     public String getCardNumber() {
9         return cardNumber;
10    }
11
12 }
```

PaymentProcessor.java > PaymentProcessor

```
1 public class PaymentProcessor {
2     public static void processPayment(Payment p) {
3         if (p instanceof CardPayment cardPayment) {
4             System.out.println("Processing card payment" + cardPayment.getCardNumber());
5         } else if (p instanceof UpIPayment upiPayment) {
6             System.out.println("Processing upi payment" + upiPayment.getUpiId());
7         }
8     }
9 }
```

The screenshot shows a Java development environment with several tabs at the top: Seventh.java, Payment.java, PaymentProcessor.java, and Upipayment.java. The active tab is Seventh.java, which contains the following code:

```
1  public class Seventh {
2      Run | Debug
3      public static void main(String[] args) {
4          Payment card = new CardPayment("1234567890");
5          Payment upi = new Upipayment("1234567890");
6          PaymentProcessor.processPayment(card);
7          PaymentProcessor.processPayment(upi);
8      }
9 }
```

Below the code editor is a navigation bar with tabs: Problems, Output, Debug Console, Terminal (which is selected), and Ports.

The terminal window displays the following command-line output:

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> & 'C:\Program Files\Red Hat\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-

□ Eight.java > ⚙ Eight > ⏪ main(String[])

```
1  public class Eight {
2      Run | Debug
3      public static void main(String[] args) {
4          String userId = "101";
5
6          String Query = """
7              SELECT id, username,email,created_at
8              FROM users
9              WHERE id = '%s'
10             ORDER BY created_at DESC
11             """.formatted(userId);
12
13         System.out.println(Query);
14     }
15 }
```

Problems Output Debug Console **Terminal** Ports

```
PS C:\Users\ansh.saxena_cloudsuf\Desktop\Workshop Assignment> & 'C:\Program Files\Red Hat\JDK\bin\java' '-cp' 'C:\Users\ansh.saxena_cloudsuf\AppData\Roaming\Red Hat\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>
```