Java code refactoring with or without ChatGPT

Survey response 1

Response ID	38
Date submitted	2024-02-02 12:30:30
Last page	13
Start language	en
Seed	784106325
Date started	2024-02-02 12:14:04
Date last action	2024-02-02 12:30:30
Total time	993.54

Survey questionnaires (Part 1)

How old are you?

28

How many years of experience do you have with Java programming?

1

For how many years have you been programming for larger software projects e.g. in a company? Please enter a number between 0 and 30.

0

How many years of experience do you have with code refactoring?

0

Did you study programming or computer science at a university?

Yes

During your education, how many courses did you take where Java was the primary language?

1

On a scale from 1 to 5, how would you rate your Java programming expertise (e.g 1-very inexperienced, 5-very experienced)?

1

How would you compare your Java expertise to those with over 20 years of practical experience (e.g 1-very inexperienced, 5-very experienced)?

1

How would you rate your Java expertise in comparison to your peers or colleagues (e.g 1-very inexperienced, 5-very experienced)?

1

How often have you used Chat GPT (e.g 1-low, 5-high)?

1

Have you used ChatGPT for code refactoring tasks (e.g 1-low, 5-high)?

1

What is the average size of Java professional projects you typically work on, categorized as small-scale (up to 900 lines of code), medium-scale (900 to 40,000 lines of code), or large-scale (exceeding 40,000 lines of code)?

Small

Group time: Survey questionnaires (Part 1)

90.65

Task Explanation

Tasks Overview: In each of the two sections, you will encounter five Java code snippets that require refactoring. The first five snippets must be refactored without assistance from ChatGPT, while the last five snippets can be refactored with the aid of ChatGPT. Primarily, you have two alternatives: Without Assistance: Refactor the code on your own, relying on your existing knowledge and skills. With ChatGPT Assistance: Utilize the assistance of ChatGPT to receive suggestions and guidance for refactoring the code. Timing: Each assignment must be completed within a strict time constraint of 3 minutes. You must complete the work within a 3-minute timeframe, otherwise, timeouts will occur. Efficiently allocate your time to ensure timely completion of all jobs.   Instructions: Read the code: Begin by thoroughly understanding the provided Java code snippet. Refactor: Apply your refactoring skills to improve the code based on the given criteria (readability, efficiency, maintainability, etc.).

Group time: Task Explanation

12.3

Question 1 for Pretest (Part 2)

Refactor the below code snippet without ChatGPT within 3 minutes. public double getPayAmount() { double result; if (isDead) { result = deadAmount(); } else { if (isSeparated) { result = separatedAmount(); } else { if (isRetired) { result = retiredAmount(); } else { result = normalPayAmount(); } } } return result;}

Group time: Question 1 for Pretest (Part 2)

17.87

Question 2

Refactor the below code snippet without ChatGPT within 3 minutes. public class Customer { private String name; private String address; private double balance; public Customer(String name, String address) { this.name = name; this.address = address; this.balance = 0; } public void deposit(double amount) { this.balance += amount; } public void withdraw(double amount) { this.balance -= amount; } public double getBalance() { return balance; } }

Group time: Question 2

20.37

Question 3

Refactor the below code snippet without ChatGPT within 3 minutes. public class Customer { private String name; private String address; private double balance; public Customer(String name, String address, double initialBalance) { this.name = name; this.address = address; this.balance = initialBalance; } public void processPayment(double amount) { if (amount > balance) { throw new InsufficientFundsException(); } balance -= amount; } public void printStatement() { System.out.println("Customer name: " + name); System.out.println("Customer address: " + address); System.out.println("Customer balance: " + balance); }}

```
public class Customer {
    private String name;
    private String address;
    private double balance;

    public Customer(String name, String address, double initialBalance) {
        this.name = name;
        this.address = address;
        this.balance = initialBalance;
    }

    public void processPayment(double amount) {
        if (amount > balance) {
            throw new InsufficientFundsException("Insufficient funds for payment");
        }
        balance -= amount;
    }

    public void printStatement() {
        System.out.println("Customer Details:");
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
        System.out.println("Balance: $" + balance);
    }

    // Additional getter methods for encapsulation
    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

    public double getBalance() {
        return balance;
    }
}
```

Group time: Question 3
199.3

Question 4

Refactor the below code snippet without ChatGPT within 3 minutes. public class ShippingService { public double calculateShippingCost(Order order) { double totalPrice = order.getTotalPrice(); double weight = order.getWeight(); if (totalPrice > 100) { if (weight > 10) { return totalPrice * 0.2; } else { return totalPrice * 0.05; } } else { return 0; } }}

```
public class ShippingService {
    public double calculateShippingCost(Order order) {
        double totalPrice = order.getTotalPrice();
        double weight = order.getWeight();

        if (totalPrice > 100) {
            return (weight > 10) ? totalPrice * 0.2 : totalPrice * 0.05;
        } else {
            return 0;
        }
    }
}
```

Group time: Question 4
95.23

Question 5

Refactor the below code snippet without ChatGPT within 3 minutes. import java.util.concurrent.atomic.AtomicInteger; public class Order { private String customerName; private String productName; private double price; private int orderId; private static final AtomicInteger orderIdGenerator = new AtomicInteger(1000); public Order(String customerName, String productName, double price) { this.customerName = customerName; this.productName = productName; this.price = price; this.orderId = orderIdGenerator.incrementAndGet(); } public String toString() { String nameAndPrice = customerName + "," + String.valueOf(price); return nameAndPrice + "," + orderId; }}

```
import java.util.concurrent.atomic.AtomicInteger;

public class Order {
    private static final AtomicInteger orderIdGenerator = new AtomicInteger(1000);

    private String customerName;
    private String productName;
    private double price;
    private int orderId;

    public Order(String customerName, String productName, double price) {
        this.customerName = customerName;
        this.productName = productName;
        this.price = price;
        this.orderId = orderIdGenerator.incrementAndGet();
    }

    @Override
    public String toString() {
        return String.format("Customer: %s, Product: %s, Price: %.2f, OrderID: %d",
                customerName, productName, price, orderId);
    }

    // Additional getter methods for encapsulation
    public String getCustomerName() {
        return customerName;
    }

    public String getProductName() {
        return productName;
    }

    public double getPrice() {
        return price;
    }

    public int getOrderId() {
        return orderId;
    }
}
```

Group time: Question 5
114.86

Question 1 for Posttest (Part 3)

Refactor the below code snippet with ChatGPT within 3 minutes. public double getPayAmount() { double result; if (isDead) { result = deadAmount(); } else { if (isSeparated) { result = separatedAmount(); } else { if (isRetired) { result = retiredAmount(); } else { result = normalPayAmount(); } } } return result;}

```
public double getPayAmount() {
    if (isDead) {
        return deadAmount();
    }
    if (isSeparated) {
        return separatedAmount();
    }
    if (isRetired) {
        return retiredAmount();
    }
    return normalPayAmount();
}
```

Group time: Question 1 for Posttest (Part 3)

55.3

Question 2

Refactor the below code snippet with ChatGPT within 3 minutes. public class Customer { private String name; private String address; private double balance; public Customer(String name, String address) { this.name = name; this.address = address; this.balance = 0; } public void deposit(double amount) { this.balance += amount; } public void withdraw(double amount) { this.balance -= amount; } public double getBalance() { return balance; } }

```
public class Customer {
    private String name;
    private String address;
    private double balance;

    public Customer(String name, String address) {
        this.name = name;
        this.address = address;
        this.balance = 0;
    }

    public void deposit(double amount) {
        this.balance += amount;
    }

    public void withdraw(double amount) {
        if (amount > balance) {
            throw new InsufficientFundsException("Insufficient funds for withdrawal");
        }
        this.balance -= amount;
    }

    public double getBalance() {
        return balance;
    }

    // Additional getter methods for encapsulation
    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }
}
```

Group time: Question 2
46.74

Question 3

Refactor the below code snippet with ChatGPT within 3 minutes. public class Customer { private String name; private String address; private double balance; public Customer(String name, String address, double initialBalance) { this.name = name; this.address = address; this.balance = initialBalance; } public void processPayment(double amount) { if (amount > balance) { throw new InsufficientFundsException(); } balance -= amount; } public void printStatement() { System.out.println("Customer name: " + name); System.out.println("Customer address: " + address); System.out.println("Customer balance: " + balance); }}

```
public class Customer {
    private String name;
    private String address;
    private double balance;

    public Customer(String name, String address, double initialBalance) {
        this.name = name;
        this.address = address;
        this.balance = initialBalance;
    }

    public void processPayment(double amount) {
        if (amount > balance) {
            throw new InsufficientFundsException("Insufficient funds for payment");
        }
        balance -= amou
```

Group time: Question 3
39.99

Question 4

Refactor the below code snippet with ChatGPT within 3 minutes. public class ShippingService { public double calculateShippingCost(Order order) { double totalPrice = order.getTotalPrice(); double weight = order.getWeight(); if (totalPrice > 100) { if (weight > 10) { return totalPrice * 0.2; } else { return totalPrice * 0.05; } } else { return 0; } }}

```
public class ShippingService {
    public double calculateShippingCost(Order order) {
        double totalPrice = order.getTotalPrice();
        double weight = order.getWeight();

        if (totalPrice > 100) {
            return (weight > 10) ? totalPrice * 0.2 : totalPrice * 0.05;
        } else {
            return 0;
        }
    }
}
```

Group time: Question 4
49.95

Question 5

Refactor the below code snippet with ChatGPT within 3 minutes. import java.util.concurrent.atomic.AtomicInteger; public class Order { private String customerName; private String productName; private double price; private int orderId; private static final AtomicInteger orderIdGenerator = new AtomicInteger(1000); public Order(String customerName, String productName, double price) { this.customerName = customerName; this.productName = productName; this.price = price; this.orderId = orderIdGenerator.incrementAndGet(); } public String toString() { String nameAndPrice = customerName + "," + String.valueOf(price); return nameAndPrice + "," + orderId; }}

```
import java.util.concurrent.atomic.AtomicInteger;

public class Order {
    private static final AtomicInteger orderIdGenerator = new AtomicInteger(1000);

    private String customerName;
    private String productName;
    private double price;
    private int orderId;

    public Order(String customerName, String productName, double price) {
        this.customerName = customerName;
        this.productName = productName;
        this.price = price;
        this.orderId = orderIdGenerator.incrementAndGet();
    }

    @Override
    public String toString() {
        return String.format("Customer: %s, Product: %s, Price: %.2f, OrderID: %d",
                customerName, productName, price, orderId);
    }

    // Additional getter methods for encapsulation
    public String getCus
```

Group time: Question 5

46.16

Interview Question (Part 4)

Can you share your experiences using ChatGPT for code refactoring? What were the specific benefits or advantages you observed during the process?

Fast and reliable

In what ways did ChatGPT enhance your productivity and efficiency in completing code refactoring tasks? Please provide specific examples.

Certainly. It explains me the code, helps me Unterstand it better

Did ChatGPT help you discover new refactoring techniques or approaches that you were previously unaware of? If yes, please elaborate on these insights.

I don’t think so

How did ChatGPT contribute to the maintainability and readability of the code you produced during refactoring? Were there any notable improvements or challenges in this aspect?

Yes

Were there any specific challenges or limitations you encountered while using ChatGPT for code refactoring? How did you overcome them, if at all?

I did not encountered such things

In what scenarios do you believe AI assistance, like ChatGPT, is most beneficial for code refactoring? Conversely, are there situations where you think it might be less effective or not suitable at all?

Maybe in the cases of fulfilling very specific requirements.

How does ChatGPT's performance vary depending on the complexity of the code?

Correctness

Would you recommend ChatGPT to other Java programmers?
Yes

Group time: Interview Question (Part 4)
204.82