Java code refactoring with or without ChatGPT

Survey response 1

Response ID
37

Date submitted
2024-02-01 00:47:07

Last page
13

Start language
en

Seed
1514093398

Date started
2024-02-01 00:14:04

Date last action
2024-02-01 00:47:07

Total time
1989.38

Survey questionnaires (Part 1)

How old are you?
30

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.
1

How many years of experience do you have with code refactoring?
1

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?
2

On a scale from 1 to 5, how would you rate your Java programming expertise (e.g 1-very inexperienced, 5-very experienced)?
3

How would you compare your Java expertise to those with over 20 years of practical experience (e.g 1-very inexperienced, 5-very experienced)?
2

How would you rate your Java expertise in comparison to your peers or colleagues (e.g 1-very inexperienced, 5-very experienced)?
3

How often have you used Chat GPT (e.g 1-low, 5-high)?
4

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)?
10000

Group time: Survey questionnaires (Part 1)
120.15

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
5.88

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;}

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

Group time: Question 1 for Pretest (Part 2)
85.05

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; } }

```
import java.math.BigDecimal;

public class Customer {
    private String name;
    private String address;
    private BigDecimal balance;

    public Customer(String name, String address) {
        this.name = name;
        this.address = address;
        this.balance = BigDecimal.ZERO; // initializes balance to zero
    }

    public void deposit(BigDecimal amount) {
        if (amount.compareTo(BigDecimal.ZERO) > 0) {
            this.balance = this.balance.add(amount);
        } else {
            throw new IllegalArgumentException("Deposit amount must be positive.");
        }
    }

    public void withdraw(BigDecimal amount) {
        if (amount.compareTo(BigDecimal.ZERO) > 0 && this.balance.compareTo(amount) >= 0) {
            this.balance = this.balance.subtract(amount);
        } else {
            throw new IllegalArgumentException("Withdrawal amount must be positive and within available balance.");
        }
    }

    public BigDecimal getBalance() {
        return this.balance;
    }

    // Additional getter methods for name and address if needed
    public String getName() {
        return name;
    }

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

Group time: Question 2
92.63

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); }}

```
import java.math.BigDecimal;

public class Customer {
    private String name;
    private String address;
    private BigDecimal balance;

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

    public void withdraw(BigDecimal amount) {
        if (amount.compareTo(BigDecimal.ZERO) < 0) {
            throw new IllegalArgumentException("Amount must be positive");
        }
        if (balance.compareTo(amount) < 0) {
            throw new InsufficientFundsException("Insufficient funds for withdrawal");
        }
        balance = balance.subtract(amount);
    }

    public void printStatement() {
        System.out.println("Customer name: " + name);
        System.out.println("Customer address: " + address);
        System.out.println("Customer balance: " + balance);
    }

    // Assuming InsufficientFundsException is a custom exception that must be defined
    public static class InsufficientFundsException extends RuntimeException {
        public InsufficientFundsException(String message) {
            super(message);
        }
    }

    // Getters if needed
    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

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

Group time: Question 3
149.53

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();
        if (totalPrice  10) {
            return totalPrice * 0.2;
        } else {
            return totalPrice * 0.05;
        }
    }
}
```

Group time: Question 4
61.44

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.math.BigDecimal;
import java.util.concurrent.atomic.AtomicInteger;

public class Order {
    private String customerName;
    private String productName;
    private BigDecimal price;
    private int orderId;

    private static final AtomicInteger orderIdGenerator = new AtomicInteger(1000);

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

    @Override
    public String toString() {
        return String.format("Customer Name: %s, Product Name: %s, Price: %s, Order ID: %d",
                             customerName, productName, price.toPlainString(), orderId);
    }
}
```

Group time: Question 5
95.14

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)
77.22

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; } }

```
import java.math.BigDecimal;

public class Customer {
   private String name;
   private String address;
   private BigDecimal balance;

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

   public void deposit(BigDecimal amount) {
      if (amount.compareTo(BigDecimal.ZERO) < 0) {
         throw new IllegalArgumentException("Deposit amount cannot be negative.");
      }
      this.balance = this.balance.add(amount);
   }

   public void withdraw(BigDecimal amount) {
      if (amount.compareTo(BigDecimal.ZERO) < 0) {
         throw new IllegalArgumentException("Withdrawal amount cannot be negative.");
      }
      if (this.balance.compareTo(amount) < 0) {
         throw new IllegalArgumentException("Insufficient funds.");
      }
      this.balance = this.balance.subtract(amount);
   }

   public BigDecimal getBalance() {
      return this.balance;
   }
}
```

Group time: Question 2
109.67

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); }}

```
import java.math.BigDecimal;

public class Customer {
    private String name;
    private String address;
    private BigDecimal balance;

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

    public void processPayment(BigDecimal amount) {
        if (amount.compareTo(BigDecimal.ZERO) < 0) {
            throw new IllegalArgumentException("Amount must be positive");
        }

        if (balance.compareTo(amount) < 0) {
            throw new InsufficientFundsException("Insufficient funds to process the payment");
        }
        balance = balance.subtract(amount);
    }

    public BigDecimal getBalance() {
        return balance;
    }

    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

    @Override
    public String toString() {
        return "Customer name: " + name + ", Customer address: " + address + ", Customer balance: " + balance.toPlainString();
    }

    // Assuming InsufficientFundsException is a custom exception that must be defined
    public static class InsufficientFundsException extends RuntimeException {
        public InsufficientFundsException(String message) {
            super(message);
        }
    }
}
```

Group time: Question 3
77.34

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 {
  private static final double HEAVY_WEIGHT_THRESHOLD = 10; // weight in kilograms
  private static final double PRICE_THRESHOLD = 100; // price in currency unit
  private static final double HEAVY_WEIGHT_SHIPPING_RATE = 0.2;
  private static final double STANDARD_SHIPPING_RATE = 0.05;

  public double calculateShippingCost(Order order) {
    double totalPrice = order.getTotalPrice();
    if (totalPrice  HEAVY_WEIGHT_THRESHOLD) {
      return totalPrice * HEAVY_WEIGHT_SHIPPING_RATE;
    }

    return totalPrice * STANDARD_SHIPPING_RATE;
  }
}
```

Group time: Question 4
53.84

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;
import java.math.BigDecimal;

public class Order {
  private String customerName;
  private String productName;
  private BigDecimal 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 = BigDecimal.valueOf(price); // Convert double to BigDecimal
    this.orderId = orderIdGenerator.incrementAndGet();
  }

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

Group time: Question 5
77.55

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?
ChatGPT has been instrumental in streamlining the code refactoring process, significantly boosting productivity by providing instant, intelligent suggestions.

In what ways did ChatGPT enhance your productivity and efficiency in completing code refactoring tasks? Please provide specific examples.
ChatGPT enhances productivity and efficiency in code refactoring by providing instant feedback and suggestions. For instance, when working with nested if-else statements, ChatGPT suggested converting them into a cleaner structure using early returns, which made the code more readable and easier to maintain. Another example is its proposal to replace primitive data types with BigDecimal for financial calculations, ensuring precision and preventing common rounding errors.

Did ChatGPT help you discover new refactoring techniques or approaches that you were previously unaware of? If yes, please elaborate on these insights.
Yes, ChatGPT introduced new refactoring approaches such as the use of design patterns to simplify complex conditional logic. For example, the Strategy pattern was suggested to encapsulate varying algorithms, enhancing the flexibility and testability of the code. Additionally, ChatGPT recommended applying the Single Responsibility Principle for clearer, more modular functions, thus promoting code that is easier to debug and extend.

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?
ChatGPT significantly contributed to the maintainability and readability of code by encouraging adherence to best practices such as modular design, which makes future changes easier and less risky. It suggested improvements like extracting methods from complex blocks of code, thereby reducing cognitive load and making the codebase more navigable. Notable improvements included the replacement of magic numbers with named constants, which made the code self-documenting and easier to understand at a glance.

Were there any specific challenges or limitations you encountered while using ChatGPT for code refactoring? How did you overcome them, if at all?
One challenge encountered with using ChatGPT for code refactoring could be its limitations in understanding the full context of a larger codebase when provided with only snippets of code. This might lead to suggestions that don't align perfectly with the existing architecture or dependencies. To overcome this, it's important to review the AI's suggestions critically and integrate them thoughtfully, ensuring they fit into the broader system. Additionally, using ChatGPT in conjunction with comprehensive testing and version control can mitigate the risks of integrating new refactorings.

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?
AI assistance like ChatGPT is most beneficial for code refactoring in scenarios where the codebase has well-defined patterns and practices, and when the refactoring tasks are straightforward, such as simplifying conditional statements, removing duplication, or renaming variables for better clarity. Conversely, AI might be less effective or not suitable in highly specialized domains or in legacy systems where the intricacies and dependencies are not clearly communicated through the code snippets provided. It may also struggle with understanding the context of larger architectural decisions, where human expertise and domain knowledge are crucial.

How does ChatGPT's performance vary depending on the complexity of the code?
ChatGPT's performance in code refactoring can vary with the complexity of the code. For simple to moderately complex code, it can effectively suggest refactoring strategies, identify anti-patterns, and propose improvements in line with best practices. However, as the complexity of the code increases, particularly with systems that have intricate logic, deep interdependencies, or domain-specific requirements, ChatGPT might not fully grasp the nuances, potentially leading to less optimal suggestions. Therefore, while it can still provide general guidance, the insights require more careful consideration and human judgment to ensure they are contextually appropriate.

Would you recommend ChatGPT to other Java programmers?
Yes, I would suggest ChatGPT to other Java programmers, especially for those looking to improve their code quality with refactoring. ChatGPT can serve as an on-demand coding assistant, offering suggestions that can help programmers write cleaner, more efficient code. It can also be a valuable learning tool, helping to reinforce best practices and introduce new coding techniques.

Group time: Interview Question (Part 4)
983.94