Java code refactoring with or without ChatGPT

Survey response 1

Response ID
19

Date submitted
2024-01-19 12:52:51

Last page
13

Start language
en

Seed
1046339373

Date started
2024-01-19 12:27:58

Date last action
2024-01-19 12:52:51

Total time
1499.81

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)?
4

How would you compare your Java expertise to those with over 20 years of practical experience (e.g 1-very inexperienced, 5-very experienced)?
3

How would you rate your Java expertise in comparison to your peers or colleagues (e.g 1-very inexperienced, 5-very experienced)?
4

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

Have you used ChatGPT for code refactoring tasks (e.g 1-low, 5-high)?
4

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)
116.91

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
36.31

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 abstract class Employee {
    public abstract double getPayAmount();
}

public class DeadEmployee extends Employee {
    public double getPayAmount() {
        return deadAmount();
    }
}

public class SeparatedEmployee extends Employee {
    public double getPayAmount() {
        return separatedAmount();
    }
}

public class RetiredEmployee extends Employee {
    public double getPayAmount() {
        return retiredAmount();
    }
}

public class NormalEmployee extends Employee {
    public double getPayAmount() {
        return normalPayAmount();
    }
}
```

Group time: Question 1 for Pretest (Part 2)
170.1

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; } }

```
public class Customer {
  private String name;
  private String address;
  private double balance;
  private static final double INITIAL_BALANCE = 0;

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

  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
106.04

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 (isPaymentValid(amount)) {
      deductAmount(amount);
    } else {
      throw new InsufficientFundsException();
    }
  }

  private boolean isPaymentValid(double amount) {
    return amount
```

Group time: Question 3
66.96

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 interface ShippingCostStrategy {
   double calculate(Order order);
}

public class HighPriceHighWeightStrategy implements ShippingCostStrategy {
   @Override
   public double calculate(Order order) {
      return order.getTotalPrice() * 0.2;
   }
}

public class HighPriceLowWeightStrategy implements ShippingCostStrategy {
   @Override
   public double calculate(Order order) {
      return order.getTotalPrice() * 0.05;
   }
}

public class LowPriceStrategy implements ShippingCostStrategy {
   @Override
   public double calculate(Order order) {
      return 0;
   }
}

public class ShippingService {
   public double calculateShippingCost(Order order) {
      ShippingCostStrategy strategy;
      if (order.getTotalPrice() > 100) {
         if (order.getWeight() > 10) {
            strategy = new HighPriceHighWeightStrategy();
         } else {
            strategy = new HighPriceLowWeightStrategy();
         }
      } else {
         strategy = new LowPriceStrategy();
      }
      return strategy.calculate(order);
   }
}
```

Group time: Question 4
86.04

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; }}

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

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

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

    public String getCustomerName() {
        return customerName;
    }

    public String getProductName() {
        return productName;
    }

    public double getPrice() {
        return price;
    }

    public int getOrderId() {
        return orderId;
    }

    @Override
    public String toString() {
        return "Order{" +
                "customerName='" + customerName + '\'' +
                ", productName='" + productName + '\'' +
                ", price=" + price +
                ", orderId=" + orderId +
                '}';
    }
}
```

Group time: Question 5
94.81

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)
155.45

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 final String name;
   private final 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) {
      if (amount > 0) {
         balance += amount;
      }
   }

   public void withdraw(double amount) {
      if (amount > 0 && balance >= amount) {
         balance -= amount;
      }
   }

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

Group time: Question 2
38.9

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) {
      setName(name);
      setAddress(address);
      setBalance(initialBalance);
   }

   private void setName(String name) {
      if (name == null || name.trim().isEmpty()) {
         throw new IllegalArgumentException("Name cannot be null or empty");
      }
      this.name = name;
   }

   private void setAddress(String address) {
      if (address == null || address.trim().isEmpty()) {
         throw new IllegalArgumentException("Address cannot be null or empty");
      }
      this.address = address;
   }

   private void setBalance(double balance) {
      if (balance < 0) {
         throw new IllegalArgumentException("Initial balance cannot be negative");
      }
      this.balance = balance;
   }

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

   public String getStatement() {
      return "Customer name: " + name + "\nCustomer address: " + address + "\nCustomer balance: " + balance;
   }

   public void printStatement() {
      System.out.println(getStatement());
   }
}
```

Group time: Question 3
88.7

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; } }}

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

        if (totalPrice  10 ? totalPrice * 0.2 : totalPrice * 0.05;
    }
}
```

Group time: Question 4
36.53

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; }}

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

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

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

    public Order(String customerName, String productName, double price) {
        validateInput(customerName, productName, price);

        this.customerName = customerName;
        this.productName = productName;
        this.price = price;
        this.orderId = orderIdGenerator.incrementAndGet();
    }

    private void validateInput(String customerName, String productName, double price) {
        if (customerName == null || customerName.isEmpty()) {
            throw new IllegalArgumentException("Customer name cannot be null or empty.");
        }
        if (productName == null || productName.isEmpty()) {
            throw new IllegalArgumentException("Product name cannot be null or empty.");
        }
        if (price < 0) {
            throw new IllegalArgumentException("Price cannot be negative.");
        }
    }

    @Override
    public String toString() {
        return "Order ID: " + orderId + ", Customer: " + customerName + ", Product: " + productName + ", Price: " + price;
    }
}
```

Group time: Question 5
83.88

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?
Efficiency and Speed: AI can quickly analyze and refactor code, often faster than manual refactoring. This speed can be especially beneficial for large codebases or when dealing with tight deadlines. Consistency: AI maintains a consistent style throughout the refactoring process, which can be crucial for maintaining a coherent codebase, especially in large teams where different coding styles might otherwise clash. Detecting Anti-Patterns: AI can easily identify and correct common coding anti-patterns, such as deeply nested conditionals, duplicated code, or violation of SOLID principles. Educational Tool: For learners and junior developers, AI can serve as a teaching tool, demonstrating best practices in coding and refactoring. It can provide multiple ways to refactor the same piece of code, offering a broader perspective on possible solutions. Error Reduction: AI can reduce the likelihood of introducing new errors during refactoring by adhering to well-tested patterns and practices. Code Analysis: AI can analyze the code beyond mere syntax; it can understand the context to some extent and suggest improvements based on best practices and standard guidelines. However, there are also limitations: Context Understanding: While AI can understand the code to a certain extent, it might not fully grasp the business logic or specific context of the application, which can sometimes lead to less optimal suggestions. Complex Refactoring: For very complex refactoring tasks that require deep understanding of the system architecture or specific domain knowledge, AI might not always provide the best solution. Human Oversight Required: It's important to review AI suggestions, as they might not always fit the specific needs or constraints of your project. In conclusion, using AI like ChatGPT for code refactoring can offer significant advantages in terms of efficiency, consistency, and learning, but it's important to complement it with human expertise and oversight.

In what ways did ChatGPT enhance your productivity and efficiency in completing code refactoring tasks? Please provide specific examples.
Quick Suggestions: Users can present a piece of code, and I can rapidly generate a refactored version. This immediate feedback accelerates the refactoring process. For instance, if a user provides a nested if-else structure, I can quickly transform it into a more readable format using guard clauses or the strategy pattern, depending on the context. Multiple Approaches: I can offer different refactoring strategies for the same piece of code. This variety allows users to compare approaches and choose the one that best fits their needs. For example, when refactoring a class with multiple responsibilities, I can suggest splitting it into several classes, each with a single responsibility, or I might propose using a facade pattern to simplify interactions. Explanation and Education: Along with refactoring suggestions, I provide explanations for why certain changes are beneficial, which can be educational for users. This aspect is particularly useful for less experienced developers or those learning new programming languages or patterns. For example, when suggesting the use of a Map over a series of if-else statements, I explain how this enhances readability and maintainability. Error Identification: I can help identify and correct common programming errors or anti-patterns. For instance, if a user's code has duplicated logic in multiple methods, I can suggest ways to abstract this logic into a single method, reducing redundancy and potential for errors. Code Standardization: I can assist in aligning the code with standard coding conventions and best practices, which is especially helpful in team environments. For example, if a user's code doesn't follow the Java Naming Conventions, I can refactor the variable and method names accordingly. Tailored Refactoring: Based on the user's requirements and the specifics of the codebase, I can offer tailored refactoring advice. For instance, if a user is working with a legacy system where certain refactoring strategies might be risky, I can provide more conservative suggestions that consider these limitations.

Did ChatGPT help you discover new refactoring techniques or approaches that you were previously unaware of? If yes, please elaborate on these insights.
yes In essence, while interacting with ChatGPT, users might gain insights into various aspects of code refactoring, from high-level design patterns and principles to specific language features and best practices. These insights can expand their toolkit for writing cleaner, more maintainable, and efficient code.

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?

Improvements
Simplifying Complex Logic: AI can suggest ways to simplify complex logic, like deep nesting or convoluted conditionals. This simplification often involves using design patterns, breaking down methods into smaller units, or employing more readable constructs, thus enhancing both readability and maintainability.

Identifying Code Smells: AI can help in recognizing code smells – signs that the code might need refactoring. This includes long methods, large classes, duplicate code, and more. By identifying these, ChatGPT can guide users towards specific improvements.

Enforcing Best Practices: AI can recommend best practices in coding, such as naming conventions, consistent formatting, and commenting. This standardization helps in making the code more understandable and easier to maintain.

Encouraging Documentation and Comments: AI can suggest adding or improving comments and documentation, which is crucial for maintainability, especially when the code is revisited after a long time or by a different team.

Refactoring Patterns: Introduction to various refactoring patterns and techniques that a developer might not be familiar with. For example, replacing conditional logic with polymorphism, or using the Builder pattern for complex object creation.

Challenges
Contextual Understanding: AI might lack complete understanding of the specific context or business logic of the code. Its suggestions, while syntactically correct, might not always align with the intended functionality or could overlook complex interdependencies.

Over-Reliance on AI: There's a risk of becoming too reliant on AI for code decisions. Developers should critically assess AI suggestions and ensure they align with the project's specific needs and standards.

Limitations in Advanced Refactoring: For very complex refactoring tasks that require deep system understanding or significant architectural changes, AI might provide suggestions that are too generic or not feasible.

Balancing Readability and Performance: Occasionally, making code more readable can impact performance. Developers need to balance these aspects, and AI might not always provide the most optimal solution in terms of performance.

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

Contextual Limitations: AI might not fully grasp the broader context or specific business logic behind the code. This can lead to suggestions that are syntactically correct but not functionally appropriate.

Overcoming: Users should review AI suggestions within the context of their project's specific requirements and constraints. Critical thinking and understanding of the project's goals are essential.
Complex Refactoring: AI might struggle with highly complex refactoring tasks that require deep understanding of the codebase or intricate system architecture.

Overcoming: For complex refactoring, AI suggestions should be used as a starting point or a source of ideas rather than a definitive solution. Combining AI guidance with expert human judgment is often necessary.

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?
Less Effective or Unsuitable Scenarios Complex Business Logic: AI may not fully understand complex, domain-specific business logic and nuances, leading to suggestions that might not be feasible or optimal. Advanced Architecture Refactoring: For large-scale architectural changes that require deep system understanding, contextual knowledge, and strategic planning, AI's capabilities might be limited. New Technologies or Languages: If the code involves cutting-edge technologies or very recent language updates, AI might not have the requisite training or knowledge to provide accurate suggestions. Sensitive and Security-Critical Code: In cases where code is highly sensitive or critical to security (like cryptographic algorithms), AI-generated refactorings should be approached with caution. Performance-Critical Sections: For performance-sensitive code, AI might not always balance readability with performance optimization effectively. Unique or Innovative Solutions: AI tends to suggest solutions based on existing knowledge and may not come up with creative or innovative solutions for unique problems.

How does ChatGPT's performance vary depending on the complexity of the code?
High Performance: ChatGPT tends to perform well with simple or straightforward code. This includes basic functions, standard algorithms, or code following common patterns. Accurate Suggestions: The AI can easily suggest improvements, identify code smells, and apply standard refactoring techniques. Quick Responses: Given the simplicity, ChatGPT can quickly analyze and provide feedback or refactored code. Educational Value: For beginners or learners, ChatGPT can effectively demonstrate best practices and simple refactorings.

Would you recommend ChatGPT to other Java programmers?
yes

Group time: Interview Question (Part 4)
419.18