

Draft Coding Standards for the Project (Prepared by Sadman Sakib Sarkar)

This draft compiles coding guidelines from multiple sources to provide a comprehensive standard for Java development within the project. The goal is to ensure consistency, readability, maintainability, and quality across the codebase.

1. Naming Conventions

Classes and Interfaces: Use nouns with the first letter of each word capitalized (PascalCase).

Example: `CustomerManager`, `InvoiceProcessor`.

Methods: Use verbs, in camelCase, with the first word lowercase and subsequent words capitalized.

Example: `calculateTotalAmount()`, `fetchCustomerDetails()`.

Variables: Use meaningful names in camelCase, avoiding one-letter or ambiguous names.

Example: `employeeName`, `totalAmount`.

Constants: Use all uppercase letters with words separated by underscores.

Example: `MAX_RETRY_ATTEMPTS`, `DEFAULT_BUFFER_SIZE`.

2. Curly Braces

Always start curly braces on a new line for classes, methods, and control structures.

Example:

```
public class OrderManager
{
    public void processOrder()
    {
        if (order.isValid())
        {
            // process
        }
    }
}
```

Avoid extra lines after opening or before closing a curly brace.

3. Indentation

Use **4 spaces** per indentation level. Tabs should not be used to maintain consistency across environments.

Apply consistent indentation for blocks, control structures, and method declarations.

Example:

```
public void processOrder()
```

```

{
    for (int i = 0; i < orders.size(); i++)
    {
        Order order = orders.get(i);
        // process order
    }
}

```

4. White Spaces

Surround operators, keywords, and commas with spaces for better readability.

Example:

```

int total = (a + b) * c;
for (int i = 0; i < 10; i++) { ... }
add(a, b, c);

```

Keep spaces between keywords like `if`, `while`, `for`, and parentheses.

Example:

```

if (isValid) { ... }

```

5. Comments

Single-line comments: Use for brief explanations, starting with `//`.

Example:

```

// Increment order count
orderCount++;

```

Block comments: Use for larger explanations, or to describe complex code blocks.

Example:

```

/*
 * This method processes an order and updates the database.
 * Make sure to validate the order before calling this
method.
 */
public void processOrder(Order order) { ... }

```

Javadoc comments: Use for documenting classes, methods, and fields.

Example:

```

/**
 * Calculates the total amount of an order.
 *
 * @param items List of items in the order.

```

```

        * @return The total order amount.
        */
        public double calculateTotalAmount(List<Item> items) { ...
    }

```

6. Method Length and Complexity

Methods should be short and perform a single task. Break down complex logic into smaller methods.

Example:

```

public void processOrder(Order order)
{
    validateOrder(order);
    applyDiscounts(order);
    updateDatabase(order);
}

```

7. Variable Scope

Declare variables in the narrowest possible scope. Avoid global variables unless necessary.

Example:

```

public void processOrder(Order order)
{
    double totalAmount = 0.0; // Declare inside method
    for (Item item : order.getItems()) {
        totalAmount += item.getPrice();
    }
}

```

8. Lambda Expressions

Use lambda expressions to simplify code and improve readability, but avoid over-complicating expressions.

Example:

```

List<Integer> filteredList = list.stream()
                                .filter(num -> num > 50)

                                .collect(Collectors.toList());

```

9. Boxing and Unboxing

Be cautious with autoboxing and avoid unnecessary boxing operations, as they can lead to `NullPointerException` or performance issues.

Example:

```
Integer count = null;  
    // Avoid passing boxed types that might be null
```

10. Exception Handling

Always handle exceptions using `try-catch` blocks with specific exceptions. Avoid catching generic `Exception` unless necessary.

Example:

```
try  
{  
    Connection con =  
DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD);  
    // ...  
}  
catch (SQLException e)  
{  
    e.printStackTrace();  
}
```

11. String Handling

Prefer `StringBuilder` or `StringBuffer` over `String` for operations that modify strings to avoid creating unnecessary objects.

Example:

```
StringBuilder sb = new StringBuilder();  
    sb.append("Order: ").append(orderId).append(" processed.");
```

12. Testing

Follow Test-Driven Development (TDD) principles. Use JUnit for writing unit tests, keeping tests simple, short, and focused on one functionality.

Example:

```
@Test  
public void testCalculateTotalAmount()  
{  
    Order order = new Order(...);  
    double total = order.calculateTotalAmount();  
    assertEquals(100.0, total, 0.01);  
}
```

13. Interfaces and Abstraction

Use interfaces where possible to define contracts, and prefer dependency injection over tight coupling.

Example:

```
public interface PaymentService {
    void processPayment(double amount);
}

public class CreditCardPayment implements PaymentService {
    @Override
    public void processPayment(double amount) {
        // Process credit card payment
    }
}
```

14. Immutability

Make objects immutable whenever possible. Use the `final` keyword for fields and avoid setter methods.

Example:

```
public class Order {
    private final String orderId;
    private final List<Item> items;

    public Order(String orderId, List<Item> items) {
        this.orderId = orderId;
        this.items = Collections.unmodifiableList(items);
    }
}
```

15. Libraries and APIs

Use standard Java libraries instead of reinventing the wheel, ensuring code consistency and reliability.

Example:

```
import java.util.List;
import java.util.stream.Collectors;
```

References:

1. Se-Education: <https://se-education.org/guides/conventions/java/intermediate.html>
2. Developer.com: <https://www.developer.com/design/top-10-java-coding-guidelines/>
3. GeeksforGeeks: <https://www.geeksforgeeks.org/coding-guidelines-in-java/>
4. Javatpoint: <https://www.javatpoint.com/coding-guidelines-in-java>
5. Oracle: <https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html>

