CS2 Style Guide (Java)

Dr. Ali
Spring 2025

General Naming Conventions

1. **Class Names**: Always capitalize the first letter of all class names. For example, `StudentRecords` or `CourseManager`.
2. **Method Names**: Use camelCase for method names: the first word should be lowercase, and subsequent words should have their first letter capitalized. For example, `processInput`, `calculateSum`.
3. **Variable Names**: Use meaningful names that describe their purpose, except for short-lived variables like `i`, `j`, or `k` in loops. Avoid cryptic names like `x1`, `tempVar2`.

Code Formatting

1. **Curly Braces**:
   - Start a new line for every curly brace `{`.
   - The closing brace `}` should align with the start of the block that it closes.

     ```
     if (condition)
     {
         // Do something
     }
     ```

2. **Indentation**:
   - Indent all code within a new block **one level deeper** than the parent block.
   - Be consistent with indentation across the file. Use either **spaces** or **tabs**, but do not mix them.
   - If using spaces, use **at least 2 spaces per level of indentation**. Avoid using only 1 space.
3. **Line Length**:
   - Do not write lines longer than **100 characters**. Break longer lines into multiple lines for readability.
4. **Blank Lines**:
   - Avoid excessive blank lines. Use one blank line to separate logical blocks of code for readability, but never more than two consecutive blank lines.
5. **Binary Operators**:
   - Always leave a space on both sides of binary operators like `+`, `-`, `*`, `/`, and `=`.
     Example: `int result = (a + b) - c;`
6. **Method Parentheses**:
   - **Do not leave a space** before the opening parenthesis when defining or calling a method.
     Example:

     ```
     System.out.println("Hello!"); // Correct
     ```

```
System.out.println ("Hello!"); // Incorrect
```

Commenting

1. **Inline Comments**:
   - Always use `//` for comments.
   - Leave a space after `//` before writing the comment:

```
// Correct usage
int total = 0; // Initialize the total
```

2. **Block Comments**:
   - Avoid using block comments (`/* */`). Stick to inline comments for simplicity and consistency.
3. **Header Comments**:
   - Place header comments (e.g., your name, course number, semester, NID) **above the import statements** at the top of the file.

     Example:

```
// Author: John Doe
// Course: CS2
// Semester: Spring 2025

import java.util.*;
```

4. **Code Block Comments**:
   - For comments describing a block of code, place them **above the block** and align them with the indentation of the block.
   - Keep comments concise but meaningful.
   - **Use proper capitalization**: Start comments with an uppercase letter.
     Example:

```
// Sort the array in ascending order
Arrays.sort(numbers);
```

5. **End-of-Line Comments**:
   - Use sparingly for brief explanations, preferably no longer than three words. For longer explanations, place comments above the line.

Best Practices

1. **Variable Declaration**:
   - Do not use `var` to declare variables, even when allowed by newer versions of Java. Explicitly declare the type for clarity.

     Example:

```
int count = 0; // Correct
var count = 0; // Avoid
```

2. **Meaningful Names**:
    - Choose names that clearly describe the purpose of variables, methods, and classes. For example, `calculateSum` is better than `doWork`.
3. **Error Handling**:
    - Include error handling (e.g., try-catch blocks) where appropriate to avoid program crashes.
4. **Method Length**:
    - Keep methods short and focused on a single task. If a method grows too long, consider splitting it into smaller helper methods.
5. **Consistent Styling**:
    - Stick to consistent naming, spacing, and formatting styles throughout your codebase.