Project related :

My project name is Nordea Bank and currently in BFSI domain.

Main purpose of this project is take of customers data and handle different application related to credit card and loans and stock etc.

In project tech stack would be spring and spring boot and hibernate and java and java fx for fronted tech as well as for testing swagger etc.

My role in this project is

* Document existing API with help of OPEN-API Swagger3 in spring boot
* Customize existing UI for an application using Java-FX
* Handle and monitor incidents and CRQ of applications in helix
* Test REST-API endpoints using postman or swagger
* Analyze, Maintain and Support 4 existing software applications.
* Work With OS and MQ team for solving bugs in applications
* Use Debugging skills to improve the quality and performance of code
* Configure and Setup application in Spring-MVC and hibernate
* Prepare program specifications and diagrams, and develops logic flowcharts

SDLc :

* Requirement analysis
* Planning
* Software design such as architectural design
* Software development
* Testing
* Deployment

**Tell me about yourself:**

1 ) Greeting + Name + brief description

- Hi, My name is Ajay Ingle, I have completed my Bachelor of engineering from Vidyalankar Institute of technology and I am currently working TCS as Junior JAVA Software developer with having 1 year 2 month of experience in Java, OOPS, Spring and Spring Boot and many more.

2) Projects, Awards, Research Paper & Internship

- I have deep interested in development and As I am in BFSI domain in Nordea Bank. I have

done work on spring boot like maintain and test application and well document by swagger. I used to solved bugs and support application. I used to handle incidents and CRQ or WO related app and connects with client regarding this and many more.

3) speak what you can give.

- I wish to use my technical acumen to contribute to team and create positive impact on society.

**Why you decided to apply for this job?**

I wanted to apply for this job because I have mainly three reasons that I wanted to tell you

First Reason: I see that this is job role that I am passionate about. I really deep interested in development area. I wanted to increase my knowledge and skills so that I can properly use for your company.

Second Reason: I have done little research about your company like your company got good rating and good employee review as well as have done more projects with clients like payment processing , neo bank , loyalty management and crypto currency platform.

Third reason: I wanted to do this job because I want to spend lot of time in work so that I can put good time in work environment and that will create supportive and creative team.

**What are all strength?**

my strength are align with my job description like I am good problem solver and easily to manage difficult task with break down approach. I used to do task within deadlines as well as respect everyone in team and good communication.

**What are all weakness?**

Maintainable Codebase : Clean code **can help develop software that is easy to change and maintain** over time.

Easier Troubleshooting : Software developed with clean coding principles **is easier to troubleshoot for problems**.

Faster Onboarding : **a quicker onboarding to keep productivity high**, and clean code helps achieve this goal.

 A piece of **code should be written to solve a specific problem**

 The software **design and implementation must be as simple as possible**,

**easy to test the codebase, preferably in an automated manner**.

How to write clean code :

1. **Project structure :**

**it's always useful to follow a consistent pattern to organize our source files, tests, configurations, data, and other code artifacts**.

 Maven, a popular build tool for Java, [prescribes a particular project structure](https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html). While we may not use Maven, it's always nice to stick to a convention.

1. **Naming Convention**

**naming conventions can go a long way in making our code readable and hence, maintainable**

* *Classes*: Class in terms of object-oriented concepts is a blueprint for objects which often represent real-world objects. Hence it's meaningful to use nouns to name classes describing them sufficiently: ex. Customer
* *Variables*: Variables in Java capture the state of the object created from a class. The name of the variable should describe the intent of the variable clearly: ex. customerName
* *Methods*: Methods in Java are always part of classes and hence generally represent an action on the state of the object created from the class. It's hence [**useful to name methods using verbs**](https://www.baeldung.com/java-pojo-class#javabeans):ex. getcustomerName()

### Source File Structure

Let's see how should a typical ordering of elements in a source file look:

* Package statement
* Import statements
  + All static imports
  + All non-static imports
* Exactly one top-level class
  + Class variables
  + Instance variables
  + Constructors
  + Methods

Apart from the above, **methods can be grouped based on their functionality or scope**. There is no one good convention, and the idea should be **decided once and then followed consistently**

1. Whitespaces : We all know that it is easier to read and understand short paragraphs compared to a large block of text

Two blank lines before starting static blocks, fields, constructors and inner classes

One blank line after a method signature that is multiline

A single space separating reserved keywords like if, for, catch from an open parentheses

A single space separating reserved keywords like else, catch from a closing parentheses

### Indentation

**a well-indented code is much easier to read and understand**

### Method Parameters

### **a long list of parameters can make it difficult for someone to read and understand the code**

* Consider [**refactoring**](https://www.baeldung.com/cs/refactoring) the method if it needs more than recommended parameters, typically a long parameter list also indicate that the method may be doing multiple things
* We may consider bundling parameters into custom-types but must be careful not to dump unrelated parameters into a single type

### Hardcoding

### **it can lead to duplication, which makes change more difficult**

* Consider replacing with constants or enums defined within Java
* Or else, replace with constants defined at the class level or in a separate class file
* If possible, replace with values which can be picked from configuration or environment

### Code Comments

### [Code comments](https://www.baeldung.com/cs/clean-code-comments) can be **beneficial while reading code to understand the non-trivial aspects**.

### Java allows two types of comments: Implementation comments and documentation comments.

* Comments should only complement a code, if we are not able to understand the code without comments, perhaps we need to refactor it
* We should use block comments rarely, possibly to describe non-trivial design decisions
* We should use JavaDoc comments for most of our classes, interfaces, public and protected methods
* All comments should be well-formed with a proper indentation for readability

### Logging

### **importance of logs can not be over-emphasized in development in general and maintenance in particular**.

* Be very clear and descriptive with contextual data in the log message
* Use external tools for tracing, aggregation, filtering of log messages for faster analytics

------------------

### DRY stands for “Don's Repeat Yourself”. This principle states that **a piece of code should not be repeated across the software**.

### KISS stands for “Keep It Simple, Stupid”. This principle states that **we should try to keep the code as simple as possible**.

### TDD stands for “Test Driven Development”. This is a programming practice that asks us to [write any code only if an automated test is failing](https://www.baeldung.com/java-test-driven-list). Hence, we've to **start with the design development of automated tests**. In Java, there are several frameworks to write automated unit tests like JUnit and TestNG.

### The benefits of such practice are tremendous. This leads to software that always works as expected.