Javadoc:  
For core java we will create java doc for understanding about our classes and methods and functionality or business achievements.

How to generate Javadoc for java classes :  
by using Javadoc cmd we can generate Javadoc for or java classes   
  
D:\lab\202402\_JavaBatch\src>javadoc -d docs Notificatio.java

Loading source file Notificatio.java...

Constructing Javadoc information...

Building index for all the packages and classes...

Standard Doclet version 22.0.2+9-70

Building tree for all the packages and classes...

Generating docs\Notificatio.html...

Notificatio.java:6: warning: no comment

public int age;

^

Notificatio.java:7: warning: no comment

public String name;

^

Notificatio.java:5: warning: use of default constructor, which does not provide a comment

public class Notificatio{

^

Generating docs\package-summary.html...

Generating docs\package-tree.html...

Generating docs\overview-tree.html...

Generating docs\allclasses-index.html...

Building index for all classes...

Generating docs\allpackages-index.html...

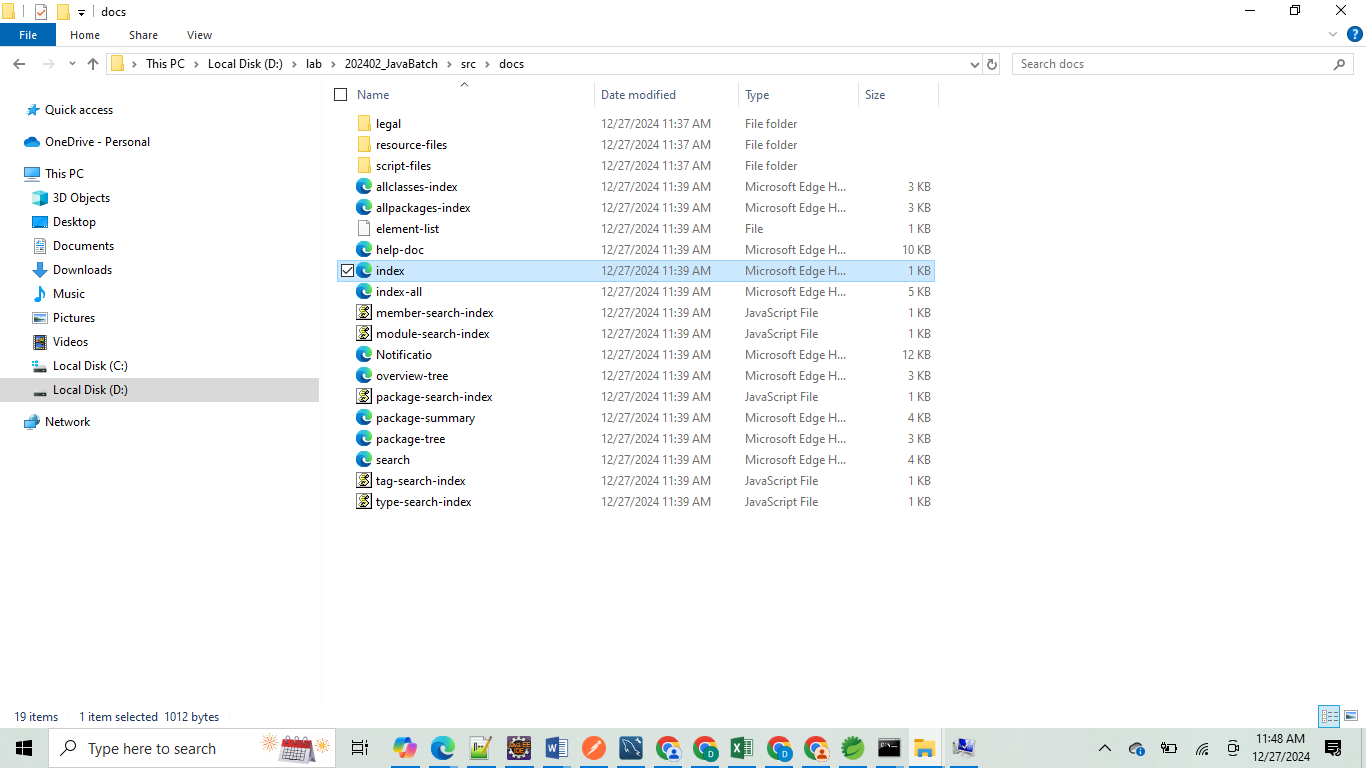
Generating docs\index-all.html...

Generating docs\search.html...

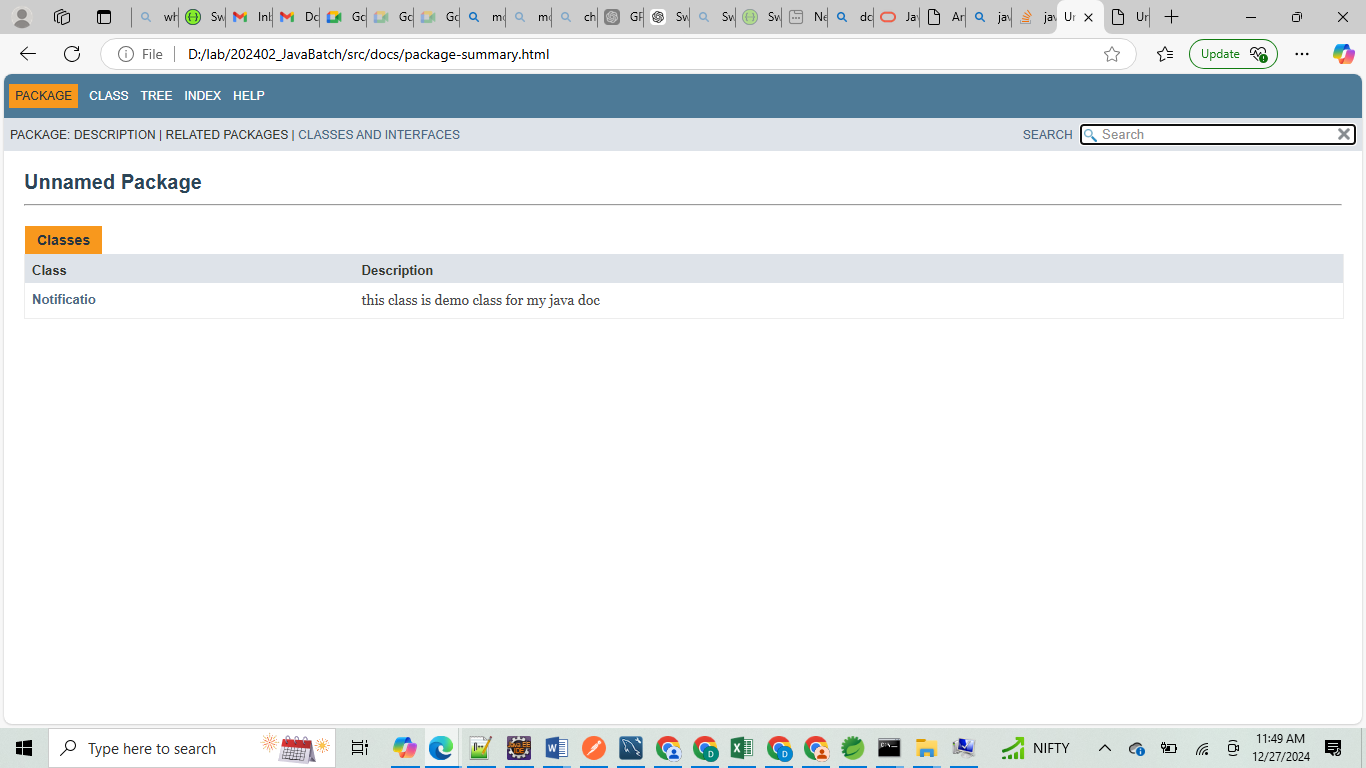
Generating docs\index.html...

Generating docs\help-doc.html...

3 warnings



Then click on index.html



How to add javac comments :  
====================

By using

/\*\*  
 here we will Javadoc comments and descriptions and Javadoc annotations which are required like

Author , since, version, param, return..

\*/

Swagger UI for spring boot:  
=========================

In springboot we can create swagger ui in two ways

* Basic swagger using @Tag and @Operation tags are controller class level with Auto configuration
* Swagger UI Using openapi.yml or swagger.yml file configurations

Basic swagger using @Tag and @Operation tags are controller class level  
================================================================

1. Add this dependency in pom.xml file

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

<version>2.7.0</version>

</dependency>

1. Add these two properties in application.properties file

springdoc.api-docs.path=/v3/api-docs

springdoc.swagger-ui.path=/swagger-ui.html

3. Add two annotations in controller classes one is for class level and one is for method level

**@Tag Annotation**

* **Purpose**:  
  The @Tag annotation is used to group related endpoints (API operations) under a single category in the Swagger UI. This makes your API documentation more organized and easier to navigate.
* **Location**:  
  Typically applied at the class level of a Spring Boot controller.
* **Attributes**:
  + name (required): The name of the tag (category) displayed in the Swagger UI.
  + description (optional): A description providing additional information about the tag.
* **Example**:

java

Copy code

import io.swagger.v3.oas.annotations.tags.Tag;

@RestController

@RequestMapping("/api/v1/products")

@Tag(name = "Product Management", description = "APIs for managing products")

public class ProductController {

// Endpoints here

}

* + **Effect in Swagger UI**: All operations (endpoints) in this controller will appear under the "Product Management" section.

### **@Operation **Annotation****

* **Purpose**:  
  The @Operation annotation is used to describe individual API operations (endpoints). It allows you to provide metadata such as a summary, description, response codes, etc.
* **Location**:  
  Applied at the method level of a controller.
* **Attributes**:
  + summary (optional): A brief overview of what the operation does.
  + description (optional): A detailed description of the operation.
  + tags (optional): An array of tag names to categorize this operation. Useful if you want to group an operation under multiple tags.
  + responses (optional): Specifies the possible responses for the operation (e.g., HTTP status codes and descriptions).
* **Example**:

java

Copy code

import io.swagger.v3.oas.annotations.Operation;

import io.swagger.v3.oas.annotations.responses.ApiResponse;

import io.swagger.v3.oas.annotations.responses.ApiResponses;

@GetMapping("/products")

@Operation(

summary = "Get All Products",

description = "Retrieve a list of all available products"

)

@ApiResponses({

@ApiResponse(responseCode = "200", description = "Successful retrieval of products"),

@ApiResponse(responseCode = "404", description = "No products found")

})

public List<Product> getAllProducts() {

return productService.getAllProducts();

}

* + **Effect in Swagger UI**:
    - The endpoint will display the provided summary and description.
    - The responses (200, 404) will also be shown in the API details.

Final example:

@RestController

@RequestMapping("/api/v1/orders")

@Tag(name = "Order Management", description = "APIs for managing orders")

**public** **class** OrderController {

@GetMapping("/{id}")

@Operation(

summary = "Get Order by ID",

description = "Fetch the details of an order by its ID"

)

**public** Order getOrderById(@PathVariable Long id) {

**return** orderService.getOrderById(id);

}

@PostMapping

@Operation(

summary = "Create a new order",

description = "Add a new order to the system",

tags = {"Order Management", "Create Operations"} // Categorize under multiple tags

)

**public** Order createOrder(@RequestBody Order order) {

**return** orderService.createOrder(order);

}

}

### Swagger UI Using openapi.yml or swagger.yml file configurations =========================================================

### ****Step 1: Add Swagger-UI Dependency to**** pom.xml

Include the SpringDoc OpenAPI dependency in your pom.xml:

<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui -->

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

<version>2.7.0</version>

</dependency>

### ****Step 2: Configure Spring Boot to Use the YAML File****

Update your application.properties (or application.yml) to specify the path to your OpenAPI YAML file:

#common swagger property for basic and yml

#

# these for normal basic swagger ui

#springdoc.api-docs.path=/v3/api-docs

#springdoc.swagger-ui.path=/swagger-ui.html

#these keys for yml swagger ui

springdoc.swagger-ui.path=/swagger-ui.html

springdoc.swagger-ui.url=/openapi.yml

springdoc.api-docs.enabled=true

#logging.level.org.springdoc=DEBUG

### ****Step 3:** Create openapi.yml or swagger.yml file in src/main/resources/static directory**

**openapi.yml  
============================**openapi: 3.1.0

info:

title: Notification API

description: API for sending notifications based on the type (email, SMS, push)

version: 1.0.0

servers:

- url: http://localhost:8080

description: Local server

tags:

- name: Notification Controller

description: API for notifications

paths:

/api/v1/sendNotification/{type}:

post:

tags:

- Notification Controller

summary: Send notifications

description: Sends notifications based on the type (email, SMS, push)

operationId: sendNotification

parameters:

- name: type

in: path

required: true

description: Type of notification to send (email, SMS, push)

schema:

type: string

enum:

- email

- sms

- push

responses:

'200':

description: Notification sent successfully

content:

application/json:

schema:

type: string

'400':

description: Invalid notification type

content:

application/json:

schema:

type: string  
  
Explanation of Key Sections:

1. **info**: Contains metadata about the API such as its title, description, and version.
2. **servers**: Specifies the base URL of the API.
3. **tags**: Groups operations logically; in this case, all endpoints belong to the "Notification Controller."
4. **paths**: Defines the available endpoints. Here, the /api/v1/sendNotification/{type} endpoint is documented with a POST method.
5. **parameters**: Describes the type path parameter, specifying the valid options (email, SMS, push).
6. **responses**: Lists possible HTTP responses (e.g., 200 for success and 400 for invalid input).

### ****Step 4: Access Swagger UI****

1. Start your Spring Boot application.
2. Open your browser and go to:  
   <http://localhost:8080/swagger-ui.html>

### Swagger UI will load your openapi.yml file and display the API documentation. ****Advantages of Using a YAML File****

1. **Version Control**: YAML files are easy to track in version control systems like Git.
2. **Reusability**: The YAML file can be reused across different environments and tools.
3. **Manual Edits**: You can manually edit and share the YAML file without modifying the codebase.

This approach works well for projects where the API specification needs to be managed outside the application code or shared with external teams.