How to Compile and Run the Appointment Management System Code

By following these instructions, you can set up, compile, and run the Appointment Management System code. The provided test data will demonstrate the enhancements effectively, showcasing how appointments are managed, validated, and handled within the system.

Prerequisites: Set up

1. Java Development Kit (JDK): Ensure you have the JDK installed.

2. Environment Variables: Set up your environment variables to include `JAVA\_HOME` and add `JAVA\_HOME/bin` to your system's `PATH`.

Step-by-Step Instructions

1. Set Up Directory Structure:

- Open a terminal or command prompt.

- Navigate to the directory where you want to create your project using the `cd` command.

- Create the main directory and navigate into it:

2. Create Java Files and Add Code:

Create and write the following Java files inside the `appointmentservice` directory:

a. Appointment.java:

- Create a new file named `Appointment.java`.

- Open the file in a text editor and add the following code:

package appointmentservice;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.UUID;

public class Appointment {

private String appointmentid;

private String appointmentdate;

private String description;

private static final int ID\_LENGTH = 10;

private static final int DESCRIPTION\_LENGTH = 50;

public String getID() {

return appointmentid;

}

public String getDate() {

return appointmentdate;

}

public String getDescription() {

return description;

}

protected void setDescription(String description) {

if (description == null) {

throw new IllegalArgumentException("Description must not be null. Please provide a valid description.");

} else if (description.length() > DESCRIPTION\_LENGTH) {

throw new IllegalArgumentException("Description must not exceed " +

DESCRIPTION\_LENGTH + " characters. Please shorten the description to fit within the character limit.");

}

this.description = description;

}

protected void setUniqueId() {

String uuid = UUID.randomUUID().toString();

// Use ID\_LENGTH to limit the length of the appointment ID

this.appointmentid = uuid.substring(0, ID\_LENGTH);

}

protected void setDate(String date) {

if (date == null) {

throw new IllegalArgumentException("Date must not be null. Please provide a valid date.");

}

if (!isFutureDate(date)) {

throw new IllegalArgumentException("Appointment date must be in the future. Please select a valid future date.");

}

this.appointmentdate = date;

}

private boolean isFutureDate(String date) {

try {

Date appointmentDate = new SimpleDateFormat("yyyy-MM-dd").parse(date);

return appointmentDate.after(new Date());

} catch (Exception e) {

throw new IllegalArgumentException("Invalid date format. Please use the format yyyy-MM-dd.");

}

}

// Constructor using Date and description

public Appointment(Date date, String description) {

setUniqueId();

setDate(new SimpleDateFormat("yyyy-MM-dd").format(date));

setDescription(description);

}

@Override

public String toString() {

return "Appointment: ID=" + appointmentid + ", date=" + appointmentdate + ", description=" + description;

}

public static void main(String[] args) {

// Test the enhancements

try {

// Valid appointment

Appointment validAppointment = new Appointment(new Date(System.currentTimeMillis() + 86400000), "Valid description");

System.out.println(validAppointment);

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

try {

// Invalid null description

new Appointment(new Date(System.currentTimeMillis() + 86400000), null);

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

try {

// Invalid long description

new Appointment(new Date(System.currentTimeMillis() + 86400000), "This description is way too long and exceeds the fifty character limit.");

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

try {

// Invalid past date

new Appointment(new Date(System.currentTimeMillis() - 86400000), "Past date description");

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

try {

// Invalid date format

new Appointment(new SimpleDateFormat("yyyy-MM-dd").parse("invalid-date"), "Invalid date format");

} catch (Exception e) {

System.out.println("Unparseable date: \"invalid-date\"");

}

}

}

3. Compile the Code:

- Ensure you are in the `appointmentservice` directory by running:

- Compile all Java files

- If the compilation is successful, it will return to the command prompt without errors.

4. Run the Application:

- Run the `Main` method to see the output

- The output will demonstrate the functionalities and handle various test cases, showing both valid and invalid scenarios.

Explanation of Output:

1. Valid Appointment:

- Output: Appointment: ID=69f8fa38-d, date=2024-07-28, description=Valid description

- Explanation: A valid appointment is created with an ID, future date, and valid description.

2. Null Description:

- Output: Description must not be null. Please provide a valid description

- Explanation: The system throws an error indicating that the description must not be null.

3. Long Description:

- Output: Description must not exceed 50 characters. Please shorten the description to fit within the character limit.

- Explanation: The system throws an error indicating that the description exceeds the 50-character limit.

4. Past Date:

- Output: Appointment date must be in the future. Please select a valid future date.

- Explanation: The system throws an error indicating that the appointment date must be in the future.

5. Invalid Date Format:

- Output: Unparseable date: "invalid-date"

- Explanation: The system catches the exception for an invalid date format and prints a user-friendly error message.

Output below:

A screenshot of a computer

Description automatically generated