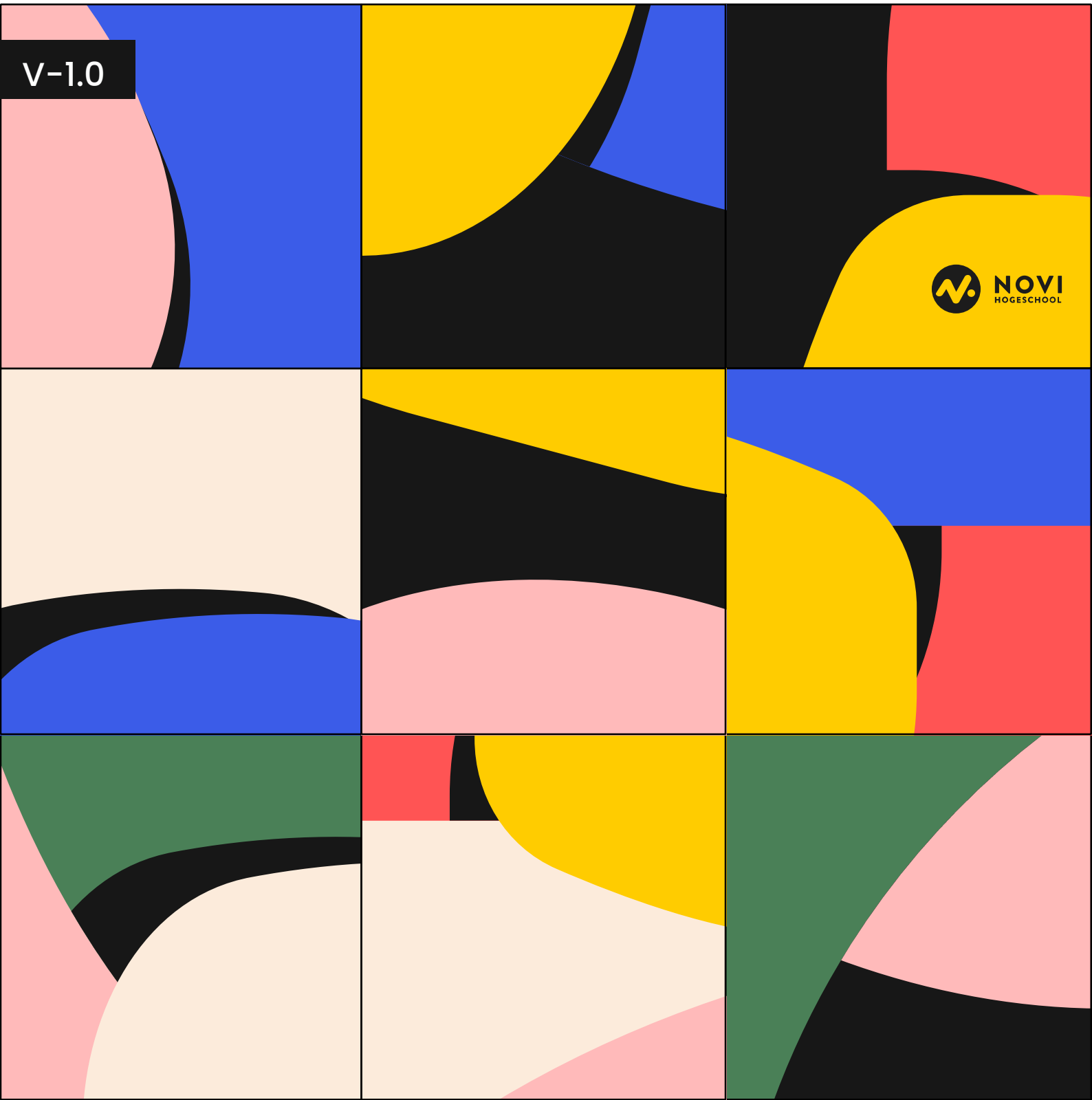
# OPDRACHT - WEEKDAGEN ENUM



# Inhoud

# Lesopdracht: Onderzoek naar Enums in Java met weekdagen

## Doel van de opdracht

Het doel van deze opdracht is om je  kennis te laten maken met het gebruik van enums in Java en hen te leren hoe je methoden binnen een enum kunnen definiëren.

## Benodigde kennis

Voor deze opdracht moeten studenten bekend zijn met:

- De basisprincipes van Java-programmeren.
- Het concept van enums in Java.
- Het schrijven van eenvoudige methoden.
- Het opzetten en uitvoeren van unit tests met JUnit.
- Het gebruik van Maven voor projectbeheer.

## Opdrachtomschrijving

Het is tijd om een enum genaamd `Weekday` op te stellen en te onderzoeken. Deze enum zal de verschillende dagen van de week beheren. De implementatie moet methoden bevatten die de dag wijzigen naar de volgende dag en de volgende werkdag, rekening houdend met het weekend.

### I. JUnit Toevoegen aan je Maven Project

Voordat je begint met het schrijven van tests, voeg je de JUnit dependency toe aan je Maven project. Dit stelt je in staat om unit tests uit te voeren met JUnit 5. Voeg de volgende dependency toe aan je `pom.xml`:

xml

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.7.0</version>
    <scope>test</scope>
</dependency>
```

## 2. Enum Implementatie

Je moet een enum genaamd `Weekday` creëren met de volgende waarden:

- `MONDAY`
- `TUESDAY`
- `WEDNESDAY`
- `THURSDAY`
- `FRIDAY`
- `SATURDAY`
- `SUNDAY`

Binnen deze enum definieer je twee methoden:

- `nextDay()`: Deze methode geeft de volgende dag van de week terug.
- `nextWorkDay()`: Deze methode geeft de volgende werkdag terug, waarbij zaterdag en zondag worden overgeslagen.

## 3. Unit Tests Schrijven

Schrijf unit tests om de functionaliteit van de `Weekday` enum te verifiëren. Gebruik JUnit 5 voor het testen. Hieronder vind je de structuur van de tests die je moet schrijven. De details zijn verborgen om de uitdaging te behouden:

## Uitwerking

```
public enum Weekday {
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY;

    public Weekday nextDay() {
        // <details>
        switch (this) {
            case MONDAY:
                return TUESDAY;
            case TUESDAY:
                return WEDNESDAY;
            case WEDNESDAY:
                return THURSDAY;
            case THURSDAY:
                return FRIDAY;
            case FRIDAY:
                return SATURDAY;
            case SATURDAY:
                return SUNDAY;
            case SUNDAY:
                return MONDAY;
            default:
                throw new IllegalStateException("Unexpected value: " + this);
        }
        // </details>
    }

    public Weekday nextWorkDay() {
        // <details>
        switch (this) {
            case FRIDAY:
                return MONDAY;
            case SATURDAY:
                return MONDAY;
            case SUNDAY:
                return MONDAY;
            default:
```

```
                return nextDay();
        }
        // </details>
    }
}
```

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class WeekdayTest {

    @Test
    public void testNextDay() {
        // Arrange
        // Act
        // Assert
        assertEquals(Weekday.TUESDAY, Weekday.MONDAY.nextDay());
        assertEquals(Weekday.WEDNESDAY, Weekday.TUESDAY.nextDay());
        assertEquals(Weekday.THURSDAY, Weekday.WEDNESDAY.nextDay());
        assertEquals(Weekday.FRIDAY, Weekday.THURSDAY.nextDay());
        assertEquals(Weekday.SATURDAY, Weekday.FRIDAY.nextDay());
        assertEquals(Weekday.SUNDAY, Weekday.SATURDAY.nextDay());
        assertEquals(Weekday.MONDAY, Weekday.SUNDAY.nextDay());
    }

    @Test
    public void testNextWorkDay() {
        // Arrange
        // Act
        // Assert
        assertEquals(Weekday.TUESDAY, Weekday.MONDAY.nextWorkDay());
        assertEquals(Weekday.WEDNESDAY, Weekday.TUESDAY.nextWorkDay());
        assertEquals(Weekday.THURSDAY, Weekday.WEDNESDAY.nextWorkDay());
        assertEquals(Weekday.FRIDAY, Weekday.THURSDAY.nextWorkDay());
        assertEquals(Weekday.MONDAY, Weekday.FRIDAY.nextWorkDay());
        assertEquals(Weekday.MONDAY, Weekday.SATURDAY.nextWorkDay());
        assertEquals(Weekday.MONDAY, Weekday.SUNDAY.nextWorkDay());
    }
}
```