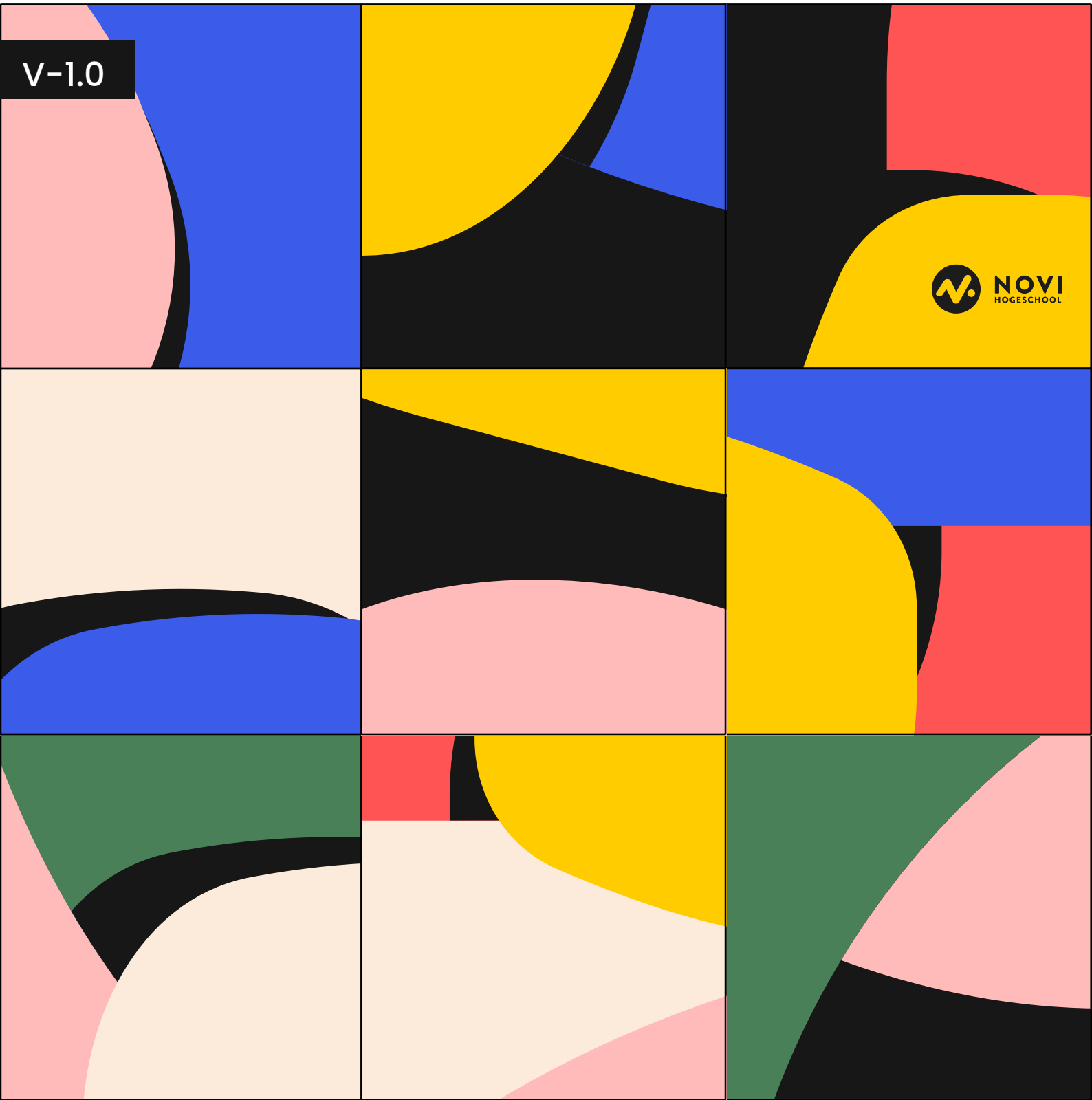
ENUMS



Inhoud

Lesopdracht: enums in java

Doel van de opdracht

Het doel van deze opdracht is om je kennis te laten maken met het gebruik van enums in Java en je te leren hoe je methoden binnen een enum kunt definiëren. Daarnaast zullen je leren hoe je unit tests kunt schrijven om de functionaliteit van hun code te verifiëren met behulp van JUnit.

Benodigde kennis

Voor deze opdracht moeten je 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 `Status` op te stellen en te onderzoeken. Deze enum zal de verschillende statussen van een taak in een project beheren. De implementatie moet methoden bevatten die de status van de taak wijzigen en enkele eenvoudige business rules afhandelen.

1. 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

```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 `Status` creëren met de volgende waarden:

- `NEW`
- `IN_PROGRESS`
- `COMPLETED`
- `ON_HOLD`
- `CANCELED`

Binnen deze enum definieer je drie methoden:

- `next()`: Deze methode geeft de volgende status terug op basis van de huidige status.
- `tryPutOnHold()`: Deze methode probeert de status op `ON_HOLD` te zetten als de huidige status `IN_PROGRESS` is.
- `tryReActivate()`: Deze methode probeert de status weer op `IN_PROGRESS` te zetten als de huidige status `ON_HOLD` is.

3. Unit Tests Schrijven

Schrijf unit tests om de functionaliteit van de `Status` 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 Status {
    NEW,
    IN_PROGRESS,
    COMPLETED,
    ON_HOLD,
    CANCELED;

    public Status next() {

        switch (this) {
            case NEW:
                return IN_PROGRESS;
            case IN_PROGRESS:
            case COMPLETED:
                return COMPLETED;
            case ON_HOLD:
            case CANCELED:
                return CANCELED;
            default:
                throw new IllegalStateException("Unexpected value: " + this);
        }

    }

    public Status tryPutOnHold() {
            if (this.equals(Status.IN_PROGRESS)) {
            return ON_HOLD;
        }
        return this;

    }

    public Status tryReActivate() {

        if (this.equals(Status.ON_HOLD)) {
            return IN_PROGRESS;
        }
        return this;

    }
}
```

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

class StatusTest {

    @Test
    public void canSetOnHold() {
        // Arrange
        // Act
        // Assert
        assertEquals(Status.ON_HOLD, Status.IN_PROGRESS.tryPutOnHold());
        assertEquals(Status.COMPLETED, Status.COMPLETED.tryPutOnHold());
    }

    @Test
    public void canReActivate() {
        // Arrange
        // Act
        // Assert
        assertEquals(Status.IN_PROGRESS, Status.ON_HOLD.tryReActivate());
        assertEquals(Status.COMPLETED, Status.COMPLETED.tryReActivate());
    }

    @Test
    public void getMoveNext() {
        // Arrange
        // Act
        // Assert
        assertEquals(Status.IN_PROGRESS, Status.NEW.next());
        assertEquals(Status.COMPLETED, Status.IN_PROGRESS.next());
        assertEquals(Status.COMPLETED, Status.COMPLETED.next());
        assertEquals(Status.CANCELED, Status.ON_HOLD.next());
        assertEquals(Status.CANCELED, Status.CANCELED.next());
        assertEquals(Status.IN_PROGRESS, Status.NEW.next());
    }
}
```