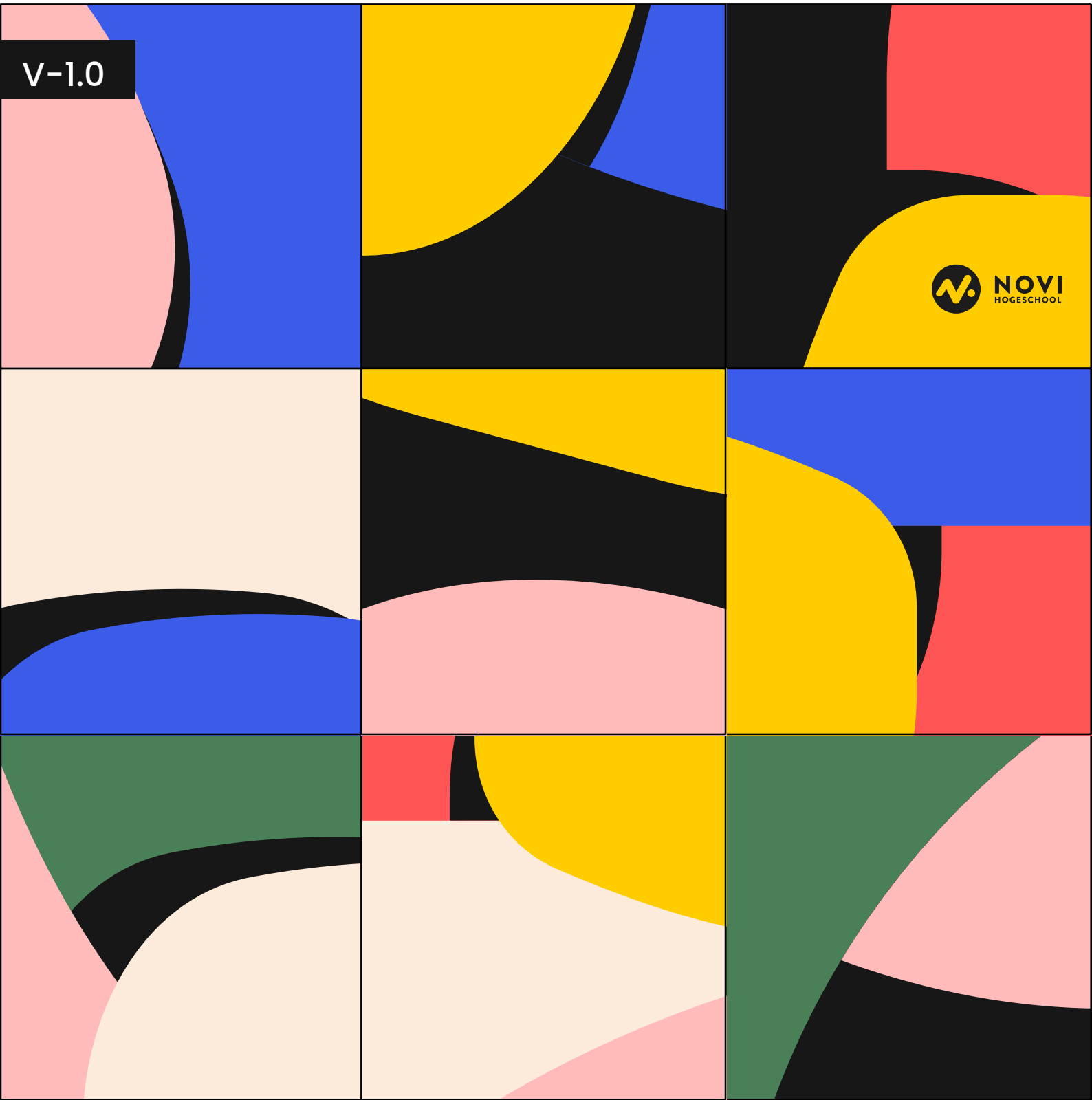
OPDRACHT - GENERIC RESULT



Inhoud

Lesopdracht: generieke klassen in java

Doel van de opdracht

Het doel van deze opdracht is om je kennis te laten maken met het gebruik van generieke klassen in Java. Je zult leren hoe je een generieke klasse kunnen definiëren en hoe je verschillende statussen (succes en falen) kunnen beheren met behulp van een Result-klasse. Daarnaast zul je leren hoe je unit tests hiervoor 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 generieke klassen in Java.
- Het schrijven van eenvoudige methoden.
- Het opzetten en uitvoeren van unit tests met JUnit.
- Het gebruik van Maven voor projectbeheer.

Opdrachtomschrijving

Je gaat een generieke klasse genaamd `Result` opstellen en onderzoeken. Deze klasse zal zowel succes- als foutresultaten kunnen bevatten. De implementatie moet methoden bevatten om succesvolle en mislukte resultaten te creëren, en moet de juiste informatie kunnen teruggeven.

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

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

2. Generieke Klasse Implementatie

Je moet een generieke klasse genaamd `Result<T>` creëren. Deze klasse moet de volgende eigenschappen en methoden hebben:

- `value`: Het waarde-object (type `T`), alleen aanwezig bij succes.
- `errorMessage`: De foutboodschap (type `String`), alleen aanwezig bij falen.
- `success`: Een boolean die aangeeft of het resultaat succesvol is.

Binnen deze klasse definieer je drie methoden:

- `success(T value)`: Een statische methode om een succesvol resultaat te creëren.
- `failure(String errorMessage)`: Een statische methode om een mislukt resultaat te creëren.
- `toString()`: Een methode die een stringrepresentatie van het resultaat geeft, afhankelijk van het succes.

3. Unit Tests Schrijven

Schrijf unit tests om de functionaliteit van de `Result` klasse 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 class Result<T> {
    private final T value;
    private final String errorMessage;
    private final boolean success;

    private Result(T value, String errorMessage, boolean success) {
        this.value = value;
        this.errorMessage = errorMessage;
        this.success = success;
    }

    public static <T> Result<T> success(T value) {
        // <details>
        return new Result<>(value, null, true);
        // </details>
    }

    public static <T> Result<T> failure(String errorMessage) {
        // <details>
        return new Result<>(null, errorMessage, false);
        // </details>
    }

    public T getValue() {
        // <details>
        return value;
        // </details>
    }

    public String getErrorMessage() {
        // <details>
        return errorMessage;
        // </details>
    }

    public boolean isSuccess() {
        // <details>
        return success;
        // </details>
    }
```

```
    @Override
    public String toString() {
        // <details>
        if (success) {
            return "Result{success, value='" + value + "'}";
        } else {
            return "Result{failure, errorMessage='" + errorMessage + "'}";
        }
        // </details>
    }
}
```

Tests

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

class ResultTests {

    @Test
    void success() {
        // Arrange
        var data = "Succes data";
        // Act
        var result = Result.success(data);
        // Assert
        assertEquals(true, result.isSuccess());
        assertEquals(data, result.getValue());
        assertEquals("Result{success, value='" + data + "'}",
result.toString());
    }

    @Test
    void failure() {
        // Arrange
        var errorText = "error 1: faulty fault";
        // Act
        var result = Result.failure(errorText);
        // Assert
        assertEquals(false, result.isSuccess());
        assertEquals(errorText, result.getErrorMessage());
        assertEquals("Result{failure, errorMessage='" + errorText + "'}",
result.toString());
    }
}
```