Les 1: Optional - Oefeningen

Gebruik IntelliJ als Editor, JDK 11 en JUnit5. Schrijf Unit Tests om je code te testen. Voor JUnit moet je onderstaande dependency toevoegen in jouw pom.xml.

```
<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.9.0</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Oefening 1: Testen op NullPointerException

1. Creëer een nieuw maven project in IntelliJ
2. Maak een klasse Motor aan met 2 properties:
    a. Naam
    b. Aantal pk
    c. Voorzie in de klasse Motor een methode “printInformatie” die de naam en het aantal pk naar console schrijft.
3. Maak een test die een nullPointerException gooit wanneer we de property “naam” wensen aan te passen. Dus de test moet een exception verwachten.
4. Maak een test die een nullPointerException gooit wanneer we de property “naam” wensen af te printen in de console. Dus de test moet een exception verwachten
5. Maak een test die een nullPointerException gooit wanneer we de functie “printInformatie” uitvoeren. Dus de test moet een exception verwachten

Wat is het resultaat van de gegeven voorbeelden? Formuleer eerst een antwoord vooraleer het zelf uit te proberen.

Oefening 2: Is er hier risico voor een NullPointerException

```
public static void main(String[] args) {
  System.out.println("Hello".equals(null));
}
```

Wat is het resultaat van de gegeven code? Formuleer eerst een antwoord vooraleer het zelf uit te proberen. Maak hier ook een test voor.

Oefening 3: Telefoon (Creatie van Optionals)

Ontwerp een Telefoonboek klasse waarmee je een telefoonnummer kan koppelen aan een persoon (een simpele String ”Voornaam Achternaam” is voldoende). Deze klasse zou de volgende zaken moeten kunnen:

1. Namen en telefoonnummers toevoegen;
2. Een Telefoonnummer opzoeken op basis van een persoon (telefoonboek);
3. Een persoon opzoeken op basis van een gegeven telefoonnummer (caller id).

Je mag er vanuit gaan dat één naam gekoppeld kan worden aan één telefoonnummer.

Zorg ervoor dat je klasse veilig is tegen het gebruik van null-objecten, en maak gebruik van de Optional API om aan te tonen dat een opzoeking geen resultaat heeft.

Schrijf testen voor je geschreven code en voer deze uit.

Oefening 4: Strikt Telefoonboek (orElseThrow)

We gaan onze Telefoonboek klasse strenger maken. Ontwerp een StriktTelefoonboek klasse die gebruik maakt van de Telefoonboek klasse met de volgende functionaliteiten:

1. Indien men geen telefoonnummer kan vinden voor de gevraagde persoon, gooi dan een NoSuchElementException;
2. Indien men geen persoon kan vinden voor het gevraagde telefoonnummer, gooi dan een eigen CallerNotKnownException.

Deze aanpassing zorgt ervoor dat er altijd een resultaat is ongeacht de opzoeking. Maak gebruik van de Optional API om de Exceptions te gooien. Schrijf testen voor je geschreven code en voer deze uit.

Oefening 5: Company Directory (orElse, map)

We gaan onze Telefoonboek klasse bruikbaar maken voor de bedrijfswereld. Ontwerp een CompanyDirectory klasse (die gebruik maakt van de Telefoonboek klasse) met de volgende functionaliteiten:

1. Zorg ervoor dat men bij het aanmaken van een CompanyDirectory de naam van de CEO het bedrijf, en het telefoonnummer van de maatschappelijke zetel kan meegeven.
2. Indien men geen telefoonnummer kan vinden voor de gevraagde persoon, geef dan het telefoonnummer van de maatschappelijke zetel terug;
3. Indien men geen persoon kan vinden voor het gevraagde telefoonnummer, geef dan de naam van de CEO het bedrijf terug.
4. Geef in plaats van een naam, een Persoon object terug met voornaam en achternaam eigenschappen. Je mag er van uitgaan dat de voornaam het eerste woord is van de naam die je kan terugvinden via je Telefoonboek klasse.

Maak gebruik van de Optional API om de extra functionaliteiten te implementeren. Schrijf testen voor je geschreven code en voer deze uit.

Oefening 6: Funky Telefoonboek (filter, ifPresent)

We gaan onze Telefoonboek klasse bruikbaar maken voor een persoon die een aantal zeer specifieke eisen heeft. Ontwerp een FunkyTelefoonboek klasse (die gebruik maakt van de Telefoonboek klasse) met de volgende functionaliteiten:

1. Geef enkel telefoonnummers terug indien deze eindigen op een even getal. Als een telefoonnummer gevonden is, maar niet eindigt met een even getal, dan moet je doen alsof het niet bestaat.
2. Indien het Funky Telefoonboek een telefoonnummer gevonden heeft, moet er in de console ”Eureka!” verschijnen.

Maak gebruik van de Optional API om de extra functionaliteiten te implementeren. Schrijf testen voor je geschreven code en voer deze uit.