Hotelapplicatie - implementatiestappen

Auteur: Ellen Leutbecher

Eerste 4 Domeinklassen implementeren en testen

1. Klassen Klant, KamerType, Kamer, Boeking implementeren met Attributen, Constructor en toString() kun je met eclipse genereren
2. Klasse Test4Klassen implementeren Hier worden alle 4 klassen en hun methoden getest
3. met Main-klasse runnen

Hotelklasse – basis implementeren

- Hotelklasse implementeren met Attributen (ook alle associaties!), Constructor en toString() versie 1 toString() NIET genereren
- Klasse TestHotel implementeren en aan Main toevoegen. Alles testen wat er al is
- Methode voegBoekingToe() uit UML zo eenvoudig mogelijk. Methoden kop en return 1
- Methode getKamerTypen() correct implementeren
- Testdata in de constructor van Hotel aanmaken (KamerTypen en Kamers)
- Klasse TestHotel uitbreiden. Alles testen wat er al is, ook getKamerTypen()

Hotel toString() uitbreiden

1. Code van toString() uitbreiden
   - check ArrayLists == null
   - Boekingen tonen
   - aantal kamers en aantal kamertypen tonen
2. Klasse TestHotel uitbreiden. Alles testen wat er al is
3. Testdata voor boekingen in de constructor van Hotel aanmaken en na de test **weghalen**!

Hotel voegBoekingToe() uitbreiden

1. Klasse TestHotelBoeking implementeren en aan Main toevoegen. Testcode uit werkboek (pagina 34) toevoegen en runnen (het loopt, inhoud fout)
2. In Klasse Hotel de methode voegBoekingToe() uitbreiden
   - tekst uit de opdracht als commentaar boven de methode
   - volgende slide volgen:

Hotelapplicatie implementeren

- Implementeer de klasse Hotel – deel 2
  `voegBoekingToe()`
  - Tijdens het ontwerp hebben we gezien dat deze methode de methode **`zoekVrijeKamer(kamerType) : Kamer`** in de Klasse Hotel aanroept.
    Tip: De eerste versie van deze methode is zo eenvoudig mogelijk. Ze geeft slechts een (nieuwe) Kamer met een kamerNummer en de meegegeven kamerType of de eerste kamer in de lijst terug.
  - Implementeer dan de **`voegBoekingToe()`** methode in kleine stappen en test ieder stap totdat deze af is
  - **Sla deze versie van je code veilig op!!**
  - Breidt nu de methode **`zoekVrijeKamer()`** in kleine stappen uit

Implementeer dan de voegBoekingToe() methode volgens deze slide:

Hotelapplicatie ontwerpen – stap 5 - methoden

Wat moet de methode **`Hotel.voegBoekingToe()`** doen?

- Een object van de klasse Klant aanmaken en naam en adres opslaan
- Een vrije kamer met de gewenste kamerType zoeken
- Een object van de klasse Boeking aanmaken en volgende objecten opslaan via de constructor of later met setters
  - Klant
  - Kamer
  - Boekdatum
  - Aankomstdatum
  - Vertrekdatum
- Dit object aan de lijst met alleBoekingen toevoegen
- Kamernummer teruggeven

3. Klasse TestHotelBoeking runnen, misschien uitbreiden, runnen.
4. Hotel methode zoekVrijeKamer() uitbreiden. Zoek in alleKamers de eerste kamer met de juiste kamertype en geef deze terug! Check of de returnwaarde null is.
5. Klasse TestHotelBoeking runnen, wijzig kamertype, runnen,
6. Klasse TestHotelBoeking voeg 2^{e} boeking met andere kamertype toe, runnen.

Tussenresultaat

We hebben een Hotelapplicatie waarmee we kamers kunnen boeken en in het geheugen van de PC opslaan.

1. De kamers zijn in het hotel
2. De kamers hebben de juiste kamertype
3. We houden geen rekening ermee of de kamer in de gewenste tijdsperiode vrij is

Dit is een voldoende basis voor les 8 en Practicum 3.

Ik verwacht dat iedereen tot hier komt met zijn/haar implementatie!

Hotel zoekVrijeKamer() uitbreiden

In het werkboek vind je de volgende tekst:

Let op: een kamer is al bezet als de nieuwe boeking **overlap** heeft met een bestaande boeking voor die kamer. Van overlap tussen boeking A en boeking B is sprake **ALS** de aankomst van boeking A voor het vertrek van boeking B valt **EN** de aankomst van boeking B voor het vertrek van boeking A valt!

Eerste inzicht: We moeten meer parameter meesturen (van, tot)

Maak een uitgebreide analyse

Maak goede testcases

Begin dan met de implementatie.