**Opgave 6: Sequence Diagrams and Java Code - Advanced**

**Opgave 6.1: Java-code vertalen naar een Sequence diagram**

Bestudeer de onderstaande code (van een deel van een applicatie voor wielrenners) en maak aan de hand ervan een Sequence Diagram.

* Begin met de aanroep van Main.main() en laat zien welke objecten vervolgens wat moeten doen.
* Pas de vertaalregels toe die in de les behandeld zijn.

public class **Main** {

public static void main(String [] args) {

Wielrenner piet = new Wielrenner("Piet", 'M', 57, 1.80);

Fiets fiets = new Fiets(1000, 2.00, 1.2, 0.5, "normaal");

piet.voegToeFiets(fiets);

}

}

public class **Wielrenner** {

private String naam;

private char geslacht;

private Fiets fiets;

public Wielrenner(String naam, char geslacht) {

this.naam = naam;

this.geslacht = geslacht;

}

public void voegToeFiets(Fiets fiets) {

this.fiets = fiets;

}

…

}

public class **Fiets** {

private int serienummer;

private double lengte;

private double hoogte;

private HashSet<Wiel> wielen;

public Fiets(int serienummer, double lengte, double hoogte,   
 double wielRadius, String bandtype) {

this.serienummer = serienummer;

this.lengte = lengte;

this.hoogte = hoogte;

maakWielen(wielRadius, bandtype);

}

private void maakWielen(double radius, String bandtype) {

for (int 1 = 0; 1 < 2 ; i++) {

Wiel wiel = new Wiel(radius, bandtype);

wielen.add(wiel);

}

}

Public double geefHoogte() {

Return hoogte;

}

}

public class **Wiel** {

private double radius;

private String bandtype;

…

}

**Opgave 6.2 Opstellen van een sequence diagram voor een use case**

Stel een sequence diagram op voor de use case Persoon Toevoegen.  
Deze use case maakt onderdeel uit van een klein maaltijdsysteem, waarbij huisgenoten en gasten geregistreerd kunnen worden, met hun voorkeuren. Daarna kunnen ze voor maaltijden inschrijven. Op basis van deze administratie kunnen er plannen gemaakt worden en automatisch bestellingen worden gedaan.

* Modelleer hoe deze use case afgehandeld kan worden (welke klassen/objecten wat in welke volgorde moeten doen) op basis van de onderstaande specificaties. Je mag user interface objecten weglaten.
* Volg de werkwijze uit de vorige les en pas de ontwerpprincipes toe: Controller, Lagen, Expert, Creator.
* Geef, daar waar een nieuw object gecreëerd wordt, aan welke parameters meegeven worden.

**Schermontwerp**



**Specificatie van de use case Persoon Toevoegen**

|  |  |  |  |
| --- | --- | --- | --- |
| Naam | Persoon Toevoegen | | |
| Aannamen | 1) Huisgenoot is ingelogd en heeft aangegeven een persoon te willen toevoegen.  2) Als het systeem de eerste keer gaat werken, dan zijn er nog geen maaltijden, personen en voorkeuren.  3) Er is één kookboek dat alle recepten bevat. | | |
| Beschrijving | **Nr** | **Actie Gebruiker** | **Reactie Systeem** |
| 1. | Start PersoonToevoegen | * Haal en toon alle receptennamen bij de  voorkeur-velden van ontbijt, lunch of diner. Toon bij ieder veld alle recepten. * Creëer een persoon-object. * Toon scherm. |
| 2. | Voert naam in | Zet naam bij persoon en controleer of naam uniek is. |
| 3. | Selecteert type | Zet type en controleer of het de waarde “gast” of “huisgenoot” heeft. |
| 4. | Selecteert een recept als voorkeur voor ontbijt, lunch of diner. | Verwerk de selectie (bestaande uit het maaltijdtype en de receptnaam) en zet de voorkeur voor ontbijt, lunch of diner. |
| 5. | Clickt op OK | Bewaar persoon met voorkeuren. Zorg er voor dat de nieuwe Persoon wordt toegevoegd aan verzameling van alle Personen |
| Uitzonderingen | Geen | | |
| Resultaat | Een nieuwe gast is toegevoegd. | | |

**Klassediagram:** zie de volgende pagina

**Klassediagram**

