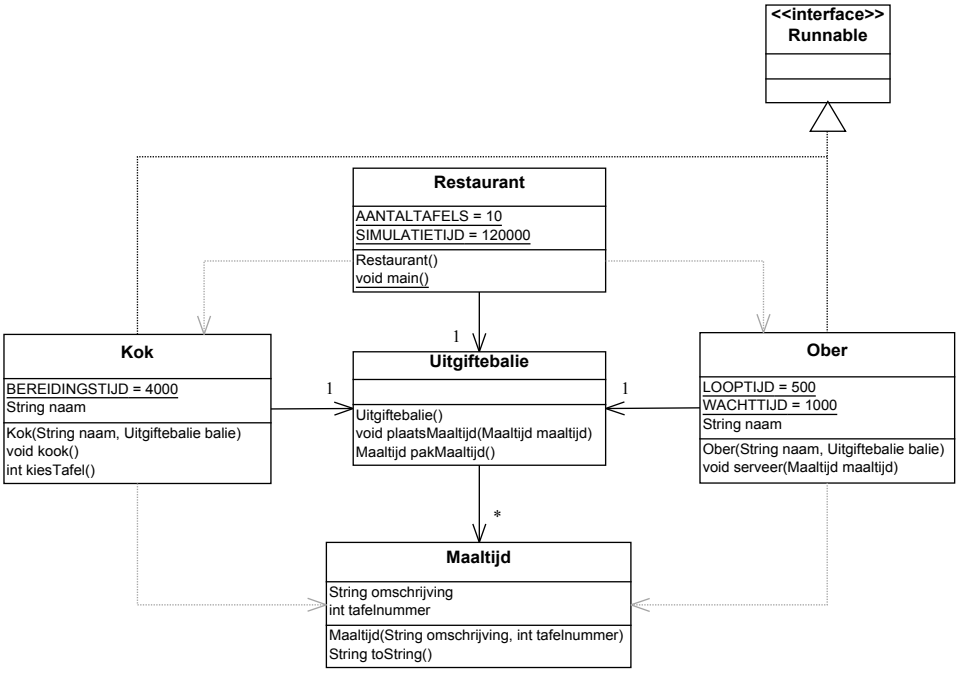
GOP - Opdracht 3

Simulatie van een restaurant

Een restaurant gebruikt een simulatieprogramma om de bediening te optimaliseren. Deze simulatie moet het volgende scenario nabootsen. Eén of meer koks bereiden maaltijden. Wanneer een kok een maaltijd heeft bereid, dan plaatst de kok deze maaltijd op de uitgiftebalie van het restaurant. Een beschikbare ober pakt de maaltijd op van de balie, kijkt voor welke tafel deze bestemd is, loopt met de maaltijd naar de tafel en serveert daar de maaltijd uit. Wanneer de ober daarmee klaar is, loopt hij terug naar de balie, en wacht tot er een nieuwe maaltijd beschikbaar is. Het opnemen van bestellingen laten we buiten beschouwing.
Er werken verschillende koks en obers tegelijkertijd in het restaurant. De opdracht draait dus om draden en om synchronizatie.

1 Het ontwerp

De te ontwikkelen applicatie bestaat uit een aantal klassen (zie figuur 1). In dit ontwerpklassendiagram ziet u naast "gewone associaties" die getekend worden als een doorgetrokken pijl met een multipliciteit ook een aantal gestippelde pijlen. Een gestippelde pijl betekent dat er geen associatie via een attribuut is maar dat de ene klasse de andere wel gebruikt (bijvoorbeeld als lokale variabele of als parameter van een methode). Dus klasse Ober maakt soms gebruik van een maaltijdobject (binnen de serveermethode) maar heeft geen attribuut van het type Maaltijd. Het restaurant kent obers en koks alleen als lokale variabelen in de main methode.



FIGUUR 1 Het ontwerpklassendiagram van het restaurant.

We geven nu kort per klasse de verantwoordelijkheden:

– De klasse Restaurant representeert het gehele restaurant. Het bevat de uitgiftebalie. Verder bevat het een publieke constante die aangeeft hoeveel tafels het restaurant heeft.

– De klasse Uitgiftebalie bevat de uitgiftebalie van het restaurant: een first-in-first-out queue van maaltijden. Een kok plaatst een maaltijd op deze balie en een ober pakt steeds de maaltijd die er het langst staat.

– De klasse Maaltijd representeert een maaltijd. Deze klasse bevat een omschrijving/naam van de maaltijd en het tafelnummer van de tafel waar de maaltijd geserveerd moet worden.

– De klasse Kok representeert een kok die achter elkaar maaltijden bereidt. Het attribuut naam bevat de naam van de kok. De constante BEREIDINGSTIJD geeft aan hoelang het duurt voorhij een maaltijd bereidt heeft. Bereiden van een maaltijd gebeurt in de methode kook().

– De klasse Ober representeert een ober die de gereedstaande maaltijden uitserveert. Het attribuut naam bevat de naam van de ober. Wanneer er een maaltijd is, dan pakt hij deze op, kijkt voor welke tafel deze is, en loopt naar de tafel. De tijd die het lopen duurt is gelijk aan de constante LOOPTIJD vermenigvuldigd met het tafelnummer (tafelnummer 1 ligt dichterbij de balie dan tafelnummer 10). Dan plaatst hij de maaltijd op de tafel. Vervolgens loopt hij weer terug naar de balie wat even lang duurt als de heenweg.
Wanneer de ober weer bij de balie is, kijkt hij of er nog een maaltijd uitgeserveerd moet worden. Is dat het geval, dan pakt hij deze maaltijd van de balie en serveert deze uit. Is dat niet het geval, dan wacht hij even (gelijk aan constante WACHTTIJD), en kijkt daarna opnieuw of er een maaltijd wordt aangeboden.

2 De opdracht

Implementeer het beschreven programma. We stellen het volgende stappenplan voor:

a Implementeer de klasse Maaltijd.
Aanwijzingen:
– Geef deze klasse een methode toString die een duidelijke stringrepresentatie voor een maaltijdobject geeft. Deze kan later gebruikt worden bij het schrijven van de acties van koks en obers.

b Implementeer de klasse Uitgiftebalie.
Aanwijzingen:
– Gebruik een ArrayList voor implementatie van de balie. Methoden pakMaaltijd en plaatsMaaltijd implementeren de gewenste first-in-first-out functionaliteit. Methode pakMaaltijd geeft het eerste maaltijdobject uit de lijst en verwijdert deze uit de lijst. Methode plaatsMaaltijd zet het gegeven maaltijdobject achter in de lijst.

c Implementeer de klasse Restaurant.

Aanwijzingen:
– De constante AANTALTAFELS mag public zijn.

d Implementeer de klasse Kok.
Aanwijzingen:
– De private methode kiesTafel() kiest een willekeurig tafelnummer. Dat wordt het tafelnummer waarvoor de maaltijd bereid gaat worden.
– De private methode kook() maakt een maaltijd voor een willekeurige tafel in het restaurant. Dit duurt BEREIDINGSTIJD lang. Wanneer de maaltijd klaar is, dan plaatst de kok deze op de balie. Verzin zelf iets handigs om iedere maaltijd een enigszins zinvolle omschrijving te geven.
– Het moet mogelijk zijn een kok in een draad te laten beginnen te werken en te stoppen met werken. Tijdens het werk, maakt hij achter elkaar maaltijden.
– Laat de kok al zijn acties (bereiden, plaatsen op balie, starten, stoppen) naar standaard uitvoer schrijven, zodat u kunt volgen wat er gebeurt.

e Implementeer de klasse Ober.
Aanwijzingen:
– Met de private methode serveer(Maaltijd) serveert de ober een maaltijd die hij van de balie heeft gepakt uit. Dat betekent lopen naar de juiste tafel, uitserveren, en teruglopen naar de balie.
– Het moet mogelijk zijn een ober in een draad te laten beginnen te werken en te stoppen met werken. Tijdens het werk, serveert hij maaltijden die op de balie staan. Is de balie leeg, dan wacht de ober even (WACHTTIJD) voordat hij weer kijkt of er een maaltijd op de balie staat om uit te serveren (met de methode serveer()).
– Laat de ober al zijn acties (pakken van balie, lopen, serveren, teruglopen, wachten, starten, stoppen) naar standaard uitvoer schrijven, zodat u kunt volgen wat er gebeurt.

f Geef een implementatie van de methode main in Restaurant. Hierin moet een restaurant worden gecreëerd. In dit restaurant moeten drie koks en twee obers aan het werk gaan. Nadat SIMULATIETIJD verstreken is moeten de kok en de obers stoppen met hun werk.
Doe tenminste tien runs en bekijk zorgvuldig de uitvoer.

Algemene aanwijzingen
– Maak attributen en methoden zoveel mogelijk private.
– Voorzie alle klassen van javadoc commentaar.
– Bedenk goed waar synchronizatieproblemen kunnen optreden, en los deze problemen op.
– U moet uw klassen natuurlijk ook testen, maar hoeft de testklassen niet mee te leveren.

3 Inleveren

Lever het volgende in:
– Het gezipte Eclipse-project met de uitwerking.
– Een toelichting waarin u uitlegt waar synchronizatieproblemen kunnen optreden en welke methoden u daartoe synchronized heeft gemaakt.