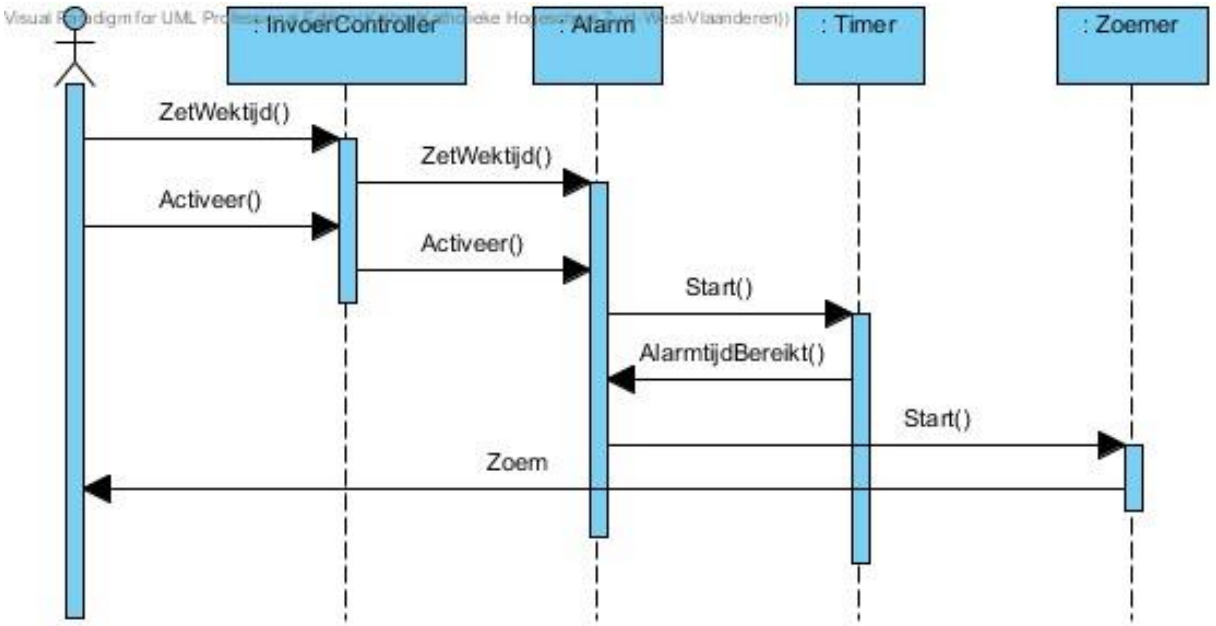
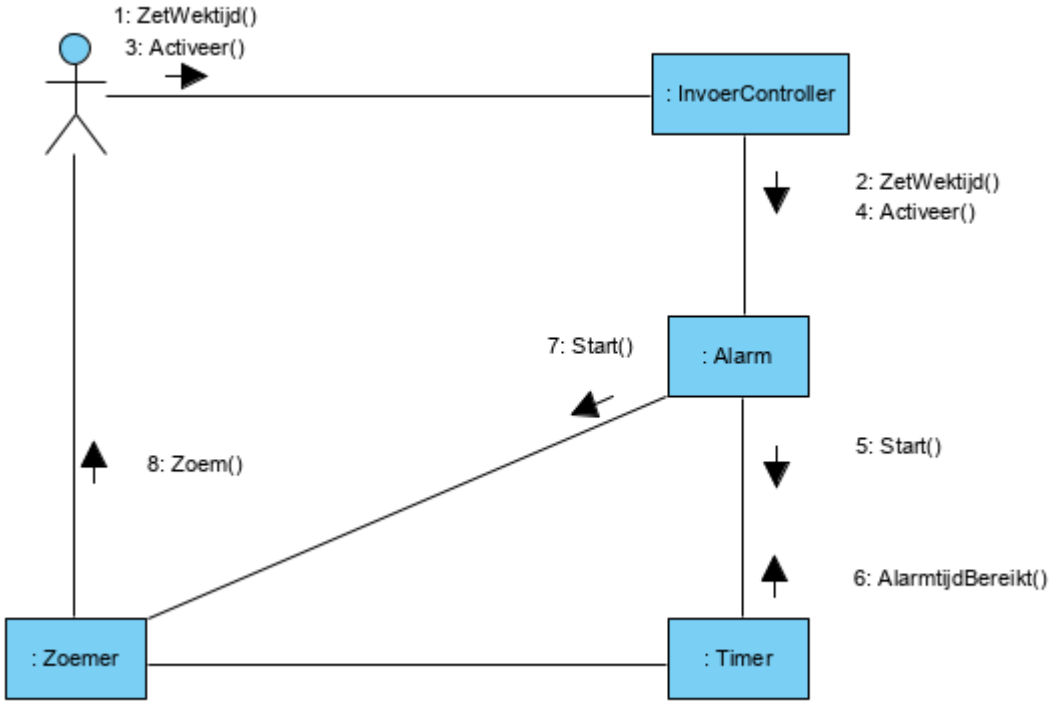
Objectgeoriënteerde analyse en ontwerp

Interactiediagrammen: oefeningen

1. Zet het sequentiediagram om in het equivalente communicatiediagram

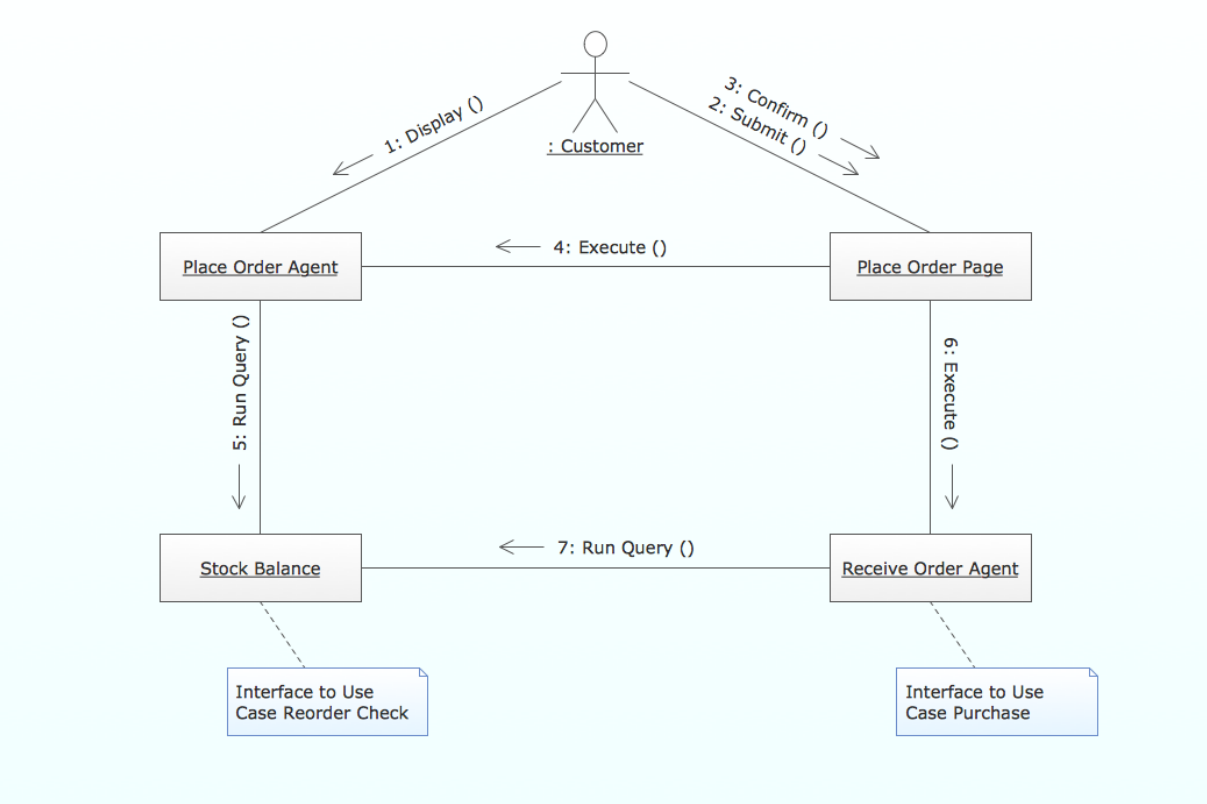


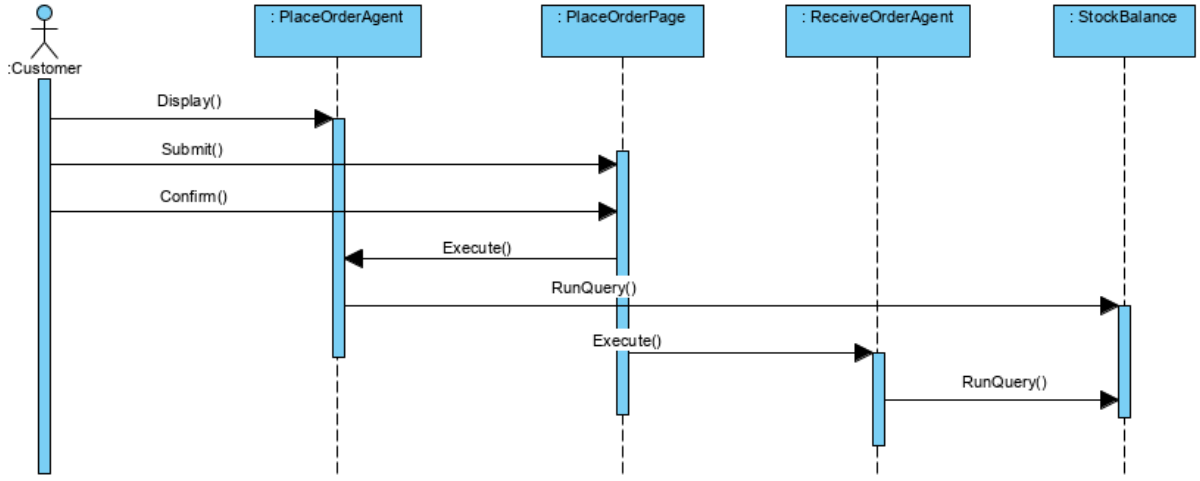


De nummering van de messages in het communicatiediagram is noodzakelijk!

Een doorlopende nummering lijkt me hier overzichtelijker dan een nummering met verschillende niveaus. Maar je bent niet verplicht een doorlopende nummering te gebruiken.

2. Zet het communicatiediagram om in het equivalente sequentiediagram





Aandachtspunten:

- De levenslijnen stellen objecten voor, geen klassen. Daarom staat er een dubbelepunt voor de naam van de klasse in de rechthoeken.
- Nummering van de messages mag, maar moet niet.

3. Zet de volgende pseudocode om in een sequentiediagram

Mainfunctie:

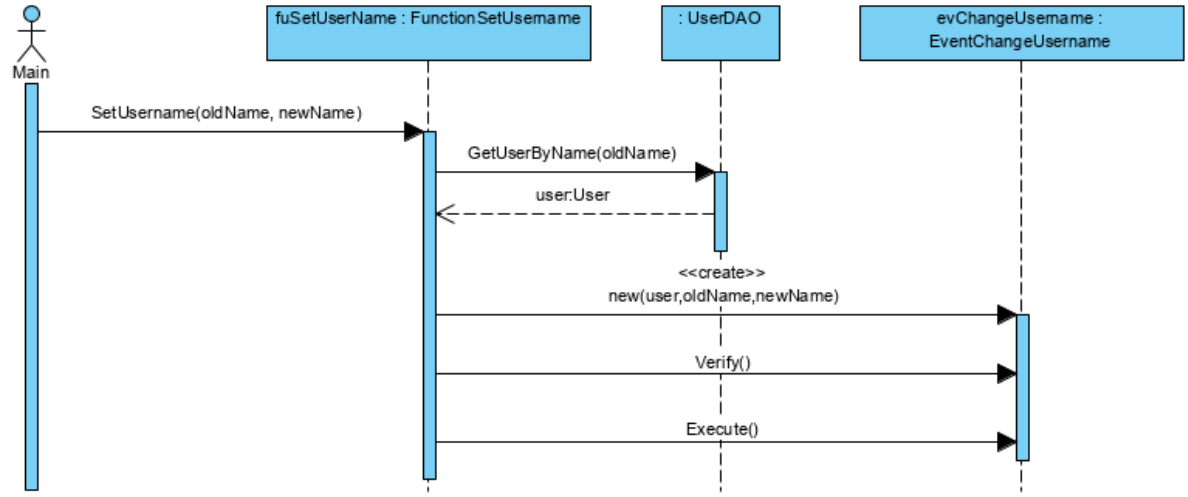
```
FunctionSetUsername fuSetUsername;
string oldName, newName;
fuSetUsername.SetUsername(oldName, newName);
```

FunctionSetUsername.java:

```
public void SetUsername(string oldName, string newName)
{
    User user = UserDAO.GetUserByName(oldName);

    EventChangeUsername evChangeUsername =
                new EventChangeUsername(user, oldName, newName);

    if (evChangeUsername.Verify())
       evChangeUsername.Execute();
};
```



- Je bent niet verplicht om in een rechthoek boven een levenslijn ook de naam van het object te vermelden. Hier doe ik dat voor die objecten waarvan in de code de naam gekend is.
- Er is geen levenslijn nodig voor user:User. Dat object is enkel bij de sequentie betrokken als returnwaarde van een message en als argument voor een message, maar stuurt zelf geen messages uit en ontvangt er ook geen.
- Voor de creatie van evChangeUsername zou je ook de andere creatienotering kunnen gebruiken (zoals in de figuur in de cursus op blz 9). Kies wat je voorkeur heeft.

- Je bent niet verplicht om de argumenten bij de messages te schrijven. Als je ze belangrijk vindt, vermeld ze dan. Als je ze niet zo belangrijk vindt, of als je diagram overladen dreigt te worden, laat ze dan weg.
- De message Execute() staat in een if-blok. Je zou ze in het diagram dus in een opt-frame kunnen zetten. Dat lijkt me hier niet zoveel meerwaarde te geven, dus heb ik het weggelaten.

4. Zet de interactie uit oefening 1 om in pseudocode

```
// externe actor roept methodes van InvoerController aan
InvoerController invoerController;

invoerController.ZetWektijd();
invoerController.Activeer();


class InvoerController {
    private Alarm alarm;

    public void ZetWektijd() {
        alarm.ZetWektijd();
    }

    public void Activeer() {
        alarm.Activeer();
    }
}


class Alarm {
    private Timer timer;
    private Zoemer zoemer;

    public void ZetWektijd() {
    }

    public void Activeer() {
        timer.Start(this);
    }

    public void AlarmtijdBereikt() {
        zoemer.Start();
    }
}


class Timer {
    public void Start(Alarm caller) {
        // wachten op alarmtijd
        caller.AlarmtijdBereikt();
    }
}
```

```
class Zoemer {
    private User user;

    public void Start() {
        user.Zoem();
    }
}
```

5. Zet de interactie uit oefening 2 om in pseudocode

```
// Customer roept methodes van PlaceOrderAgent en PlaceOrderPage aan
PlaceOrderAgent placeOrderAgent;
PlaceOrderPage placeOrderPage;

placeOrderAgent.Display();
placeOrderPage.Submit();
placeOrderPage.Confirm();

// PlaceOrderPage roept methodes in PlaceOrderAgent en ReceiveOrderAgent
aan, wanneer Confirm() uitgevoerd wordt.
class PlaceOrderPage {
    private PlaceOrderAgent placeOrderAgent;
    private ReceiveOrderAgent receiveOrderAgent;

    public void Submit() {
        // implementatie Submit()
    }

    public void Confirm() {
        placeOrderAgent.Execute();
        receiveOrderAgent.Execute();
    }
}

// PlaceOrderAgent en ReceiveOrderAgent roepen methodes in StockBalance
aan.
class PlaceOrderAgent {
    private StockBalance stockBalance;

    public void Execute() {
        stockBalance.RunQuery();
    }
}

class ReceiveOrderAgent {
    private StockBalance stockBalance;

    public void Execute() {
        stockBalance.RunQuery();
    }
```

```
}

class StockBalance {
    public void RunQuery() {
        // implementatie
    }
}
```