

Opleiding Toegepaste Informatica	
Module Object Oriented Architectures & Secure Development	
Lectoren M. Blomme, M. De Wael, F. Vlummens	Nagelezen door T. Clauwaert
Academiejaar 2021 – 2022	Semester 3
Datum en tijd 17/01/2022, 13u30 – 17u30	Zittijd 1
RESULTAAT /20	

BRONBESTANDEN

Alle benodigde bronbestanden (deze opgave, bepaalde Java-klassen, basis-FXML, SQL) kunnen gevonden worden op Leho onder de kop **“EXAM JANUARY”**.

BELANGRIJK

Elke vorm van (on-line) communicatie tussen studenten of andere partijen is strict verboden.

Dit is een individuele opdracht. Elke vaststelling van onregelmatigheid (o.a. telefoon, spieken, kopiëren, hacking, gebruik van social media clients, ...) zal conform het Onderwijs- en Examenreglement (OER) leiden tot verwittiging van zowel de betrokken student(en) als de voorzitter van de examencommissie.

Na de voltooiing van het examen, is het verboden de opgave of oplossing op enigerlei wijze te publiceren, behalve op de daartoe voorziene repository op de Gitlab-server.

Lees de opgave aandachtig en volledig door alvorens aan het examen te beginnen.

TOEGELATEN BRONNEN

Je mag gebruik maken van je eigen klasnotities, slides, boeken, syllabi en andere materialen, inclusief gebruik van het internet. Communicatie met medestudenten of derde partijen is strikt verboden (zie hierboven). Zorg ervoor dat alle applicaties die mogelijks zouden openspringen/opstarten of geïnterpreteerd zouden kunnen worden als een poging tot fraude (Outlook, Messenger, Facebook, ...) afgesloten zijn!

INDIENEN

Indienen van je volledige project gebeurt op de Gitlab-server van Toegepaste Informatica. Je hebt een individuele repository gekregen bereikbaar via het adres <https://git.ti.howest.be/TI/2021-2022/s3/object-oriented-architectures-and-secure-development/exam/01-january/firstname.lastname> (eigen voor- en familienaam, zonder accenten, spaties vervangen door punten, vb. Frédéric Vlummens → frederic.vlummens).

Zorg dat je commit op remote main, aangezien dit de branch is die door de lectoren zal geëvalueerd worden.

Je bent verplicht te committen en pushen op regelmatige basis:

- Minstens één commit per 20 minuten
- Elke commit begint met je initialen (vb. Frédéric Vlummens → FV)
- Minstens één push per uur
- Een finale commit om het indienen te voltooien (zie hieronder)

Voeg een finale commit toe (en bijhorende push) om officieel in te dienen. Hierbij voeg je in de root van je projectdirectory een README-file toe in markdown- of TXT-formaat met daarin een olijsting van de functionaliteiten die werken in je oplossing.

De finale commit message bevat bovendien volgende boodschap:

Ik, *familienaam voornaam*, verklaar hierbij dat dit mijn finale versie van het examen is en dat ik deze niet langer kan wijzigen. Ik heb in eer en geweten dit examen afgelegd zoals bedoeld en zonder te frauderen.
(vervang hierbij *familienaam voornaam* door je eigen familienaam en voornaam).

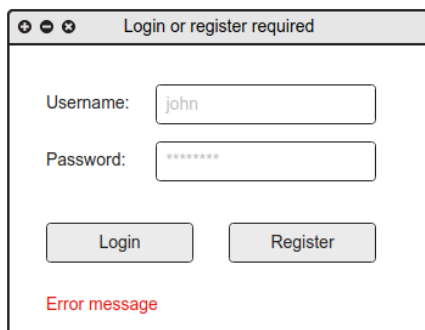
EXAMENOPGAVE

1. Algemeen

Schrijf een meerlagige *JavaFX*-applicatie met *gradle* als build-tool. De applicatie bestaat uit een inlogscherf en een scherm waarin gebruikers films kunnen opzoeken en raadplegen door middel van client-server technologie en reviews schrijven van de opgezochte films. Deze reviews worden in een lokale MySQL-database weggeschreven.

2. Beschrijving van de flow

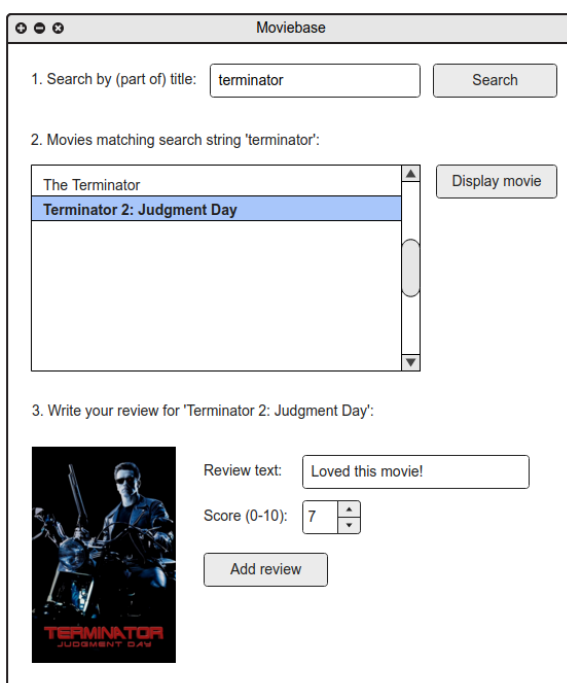
Bij het opstarten van de applicatie moeten de gebruikers registreren/aanmelden via de daartoe voorziene velden en knoppen in het aanmeldscherm:



A JavaFX window titled "Login or register required". It contains two text input fields: "Username:" with the value "john" and "Password:" with masked characters "*****". Below the fields are two buttons: "Login" and "Register". At the bottom left, there is a red text label "Error message".

Beide acties gebeuren op basis van een *username* en een *paswoord* en vervolgens een druk op één van de twee knoppen. Als er iets fout loopt bij het inloggen (vb. ongeldige username en/of paswoord) of registreren (vb. username al in gebruik) toon je een foutboodschap op het scherm zelf.

Na het succesvol aanmelden of registreren ga je als gebruiker naar het volgende scherm, waar je kan zoeken naar films op basis van (een deel van) de titel door deze in te typen en op **Search** te klikken.



A JavaFX window titled "Moviebase". It contains three sections:

- Search by (part of) title:** A text input field with the value "terminator" and a "Search" button.
- Movies matching search string 'terminator':** A list box containing two items: "The Terminator" and "Terminator 2: Judgment Day". The second item is selected and highlighted in blue. To the right of the list box is a "Display movie" button.
- Write your review for 'Terminator 2: Judgment Day':** This section includes a movie poster for "Terminator 2: Judgment Day" on the left. To the right of the poster are two input fields: "Review text:" with the value "Loved this movie!" and "Score (0-10):" with the value "7" and up/down arrow buttons. Below these fields is an "Add review" button.

De overeenkomstige films (treffers) worden getoond in de lijst. Kiest de gebruiker een film uit de lijst, dan wordt na een druk op **Display movie** onderaan de cover afbeelding van de film getoond.

Vervolgens dient de gebruiker een kort tekstje en een score van 0 tot 10 op te geven. Klikte de gebruiker op de knop **Add review**, wordt de review toegevoegd aan de lokale database (zie later).

Loopt er tijdens het toevoegen van de review iets mis, dan toon je een foutboodschap in een dialoogvenster. Ook na het succesvol toevoegen van de review toon je een bevestiging in een dialoogvenster.

The screenshot shows a Java Swing window titled "Moviebase". It contains three main sections:

- Search section:** A text field with "terminator" and a "Search" button.
- Results section:** A list box titled "2. Movies matching search string 'terminator':" containing "The Terminator" and "Terminator 2: Judgment Day". A "Display movie" button is to the right.
- Review section:** A "Thank you" dialog box with "Review has been added" and a "Close" button. Below it, a prompt "3. Write your review for 'Terminator 2: Judgment Day':" is followed by a movie poster, a "Review text:" field with "Loved this movie!", a "Score (0-10):" spinner set to 7, and an "Add review" button.

3. Samenvatting van de databronnen

- De gegevens van de **gebruikers** (voor het inloggen/registreren) hou je zelf bij in de MySQL-database (zie §5 m.b.t. de database).
- De algemene informatie over de **films** en het opzoeken ervan gebeurt via het netwerk (zie §4 m.b.t. communicatie tussen client en server).
- De **reviews** die de gebruiker ingeeft, worden dan weer in de MySQL-database bijgehouden (zie §5 m.b.t. de database).

4. Communicatie tussen client en server

De server is reeds ontwikkeld door ons. Jouw meerlagige Java-applicatie fungeert als client en communiceert met de server via de geziene technologie. Communicatie gebeurt als volgt:

1. De client verbindt met poort **32768** op het adres **172.21.24.8**.
2. De client verzendt een instantie van **MovieSearchMessage**, met de **query** de zoekterm (vb. **terminator**) over het netwerk.
3. De server antwoordt met één van volgende:
 - a. Een instantie van **MovieResultMessage**. Diens property **results** levert een ArrayList op van **Movie**-objecten die overeenkomen met de query.
 - b. Een instantie van **ErrorMessage** als er iets misgegaan is. De property **message** geeft meer uitleg over wat er precies fout liep.

Om zeker te zijn dat client en server dezelfde taal spreken, moeten zij dezelfde versie van de messageklassen **MovieSearchMessage**, **MovieResultMessage** en **ErrorMessage** hebben (die elk erven van **Message**), alsook van de domeinklasse **Movie**. Deze vijf klassen krijg je dan ook al van ons (zie Leho) in de ZIP-file **classes.zip**.

Om ervoor te zorgen dat de messages en movies over het netwerk verzonden kunnen worden, **zul je echter nog iets moeten toevoegen op de juiste plaats(en) in de code**.

Plaats de verschillende gegeven klassen ook in de juiste/relevante packages.

5. Database

- De database bevat twee tabellen: eentje met de gebruikers (username en password) en eentje met de reviews (review id, username, movie id, review text en score). Zorg er wel voor dat je Java-applicatie volgens de OO-principes werkt.

Maak gebruik van het SQL-script **moviebase.sql** om de database op te bouwen. Daarnaast krijg je van ons ook de SQL-statements om gebruikers en reviews toe te voegen in het bestand **statements.txt**. Alles is terug te vinden op Leho.

- Voorzie toegang naar de databank op een zo juist/veilig mogelijke manier. We willen immers vermijden dat iemand zaken op de database-server uitvoert die niet nodig zijn voor de applicatie. Voorzie ook een screenshot waarmee je aantoont hoe je dit gedaan hebt. Plaats deze screenshot met de naam **securedb.png** in de root van je repository.

Het moet mogelijk zijn om te verbinden met een andere MySQL-server of databank of met andere credentials, zonder de Java-broncode zelf aan te passen. Zorg er ook voor dat deze verbindingsgegevens niet zomaar in verkeerde handen kunnen vallen.

Hanteer als gebruikersnaam voor de database **moviebase-user** en als wachtwoord **moviebase-pwd**.

6. Niet-functionele vereisten

- Zorg doorheen de volledige oplossing voor nette en leesbare code (en configuratie), volgens de regels van de kunst.
- Pas de secure coding guidelines van Oracle toe waar nodig.
- Maak gebruik van logging.
- Maak doorheen je code steeds gebruik van een zelfgemaakte exceptie die je **MovieException** noemt. Geef een correcte boodschap mee.
- Voorzie één relevante JUnit5-test, opgesteld volgens de bestudeerde structuur.