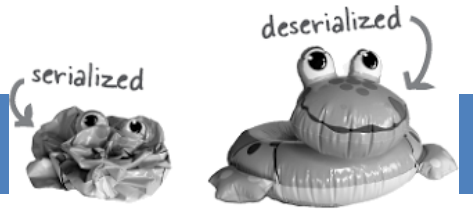
GROEIPROJECT
MODULE 7: PERSISTENTIE



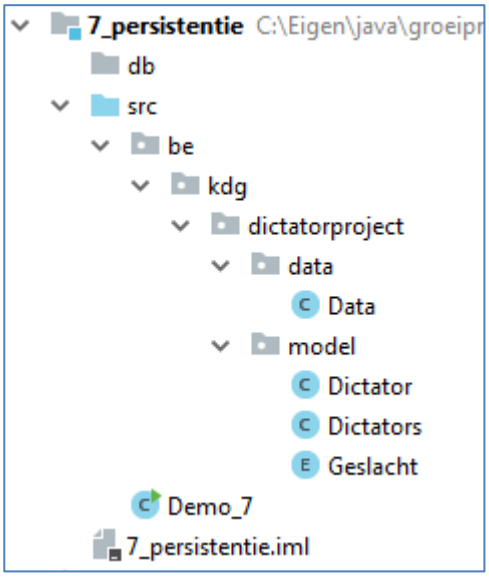
1. Voorbereiding

1.1. Open het groeiproject en maak daarin een zevende module: `7_persistentie`

1.2. Kopieer van module `1_herhaling` de hele src map (met o.a. de oorspronkelijke basis- en multiklasse). Kopieer ook de hele package `data`.

1.3. Zet in de map met de naam `lib` de benodigde jar: `hsqldb 2.3.1` Je vindt die op Blackboard.
Gebruik de menukeuze "*File* > *Project Properties* > *Project Settings* > *Libraries*" om naar deze jar te verwijzen.

1.4. Maak in de module `5_persistentie` een nieuwe directory (dus **NAAST** `src`) met de naam `db`. Hierin komen de databestanden te staan. Hiernaast een screenshot van de gewenste projectstructuur →



2. Object serialization

2.1. Zowel in de basisklasse als in de multiklasse implementeer je de `serializable` – interface.

2.2. Maak een klasse `Demo_7` met een `main`
- a. Maak een nieuwe instantie van de multiklasse aan (in ons geval `Dictators`) en laat ze vullen door de klasse `Data`.
- b. Druk het multiobject ter controle af.
- c. **Serialiseer** het object naar een databestand in de directory `db`. Gebruik een relatief pad (in ons voorbeeld: `"7_persistentie/db/dictators.ser"`
- d. **Deserialiseer** vanuit het databestand in een nieuw object en toon het opnieuw ter controle.
- e. Test of het ingelezen object gelijk is aan het oorspronkelijke. Zorg ervoor dat de multiklasse een `equals` methode bevat om deze test goed te kunnen doen.

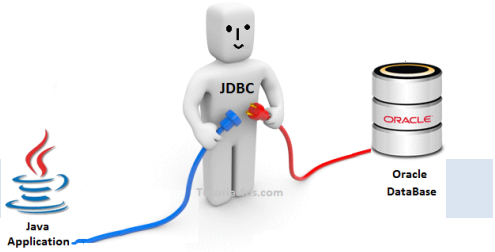
2.3. Mogelijke afdruk:

```
Voor serialize:
[Adolf Hitler(°1889) Duitsland regime: Nazisme, Augusto Pinochet(°1915) Chili regime:
Militaire dictatuur, Ayatollah Khomeini (°1902) Iran regime: Fundamentalisme, Benito
Mussolini (°1883) Italië regime: Fascisme, Ferdinand Marcos (°1917) Filipijnen regime:
enz ...]

Na deserialize:
[Adolf Hitler(°1889) Duitsland regime: Nazisme, Augusto Pinochet(°1915) Chili regime:
Militaire dictatuur, Ayatollah Khomeini (°1902) Iran regime: Fundamentalisme, Benito
Mussolini (°1883) Italië regime: Fascisme, Ferdinand Marcos (°1917) Filipijnen regime:
enz ...]

equals: true
```

3. Database connectie (JDBC)



TIP: Baseer je bij de uitwerking van de volgende opdrachten op de code-voorbeelden die je op BB vindt.

3.1. Omdat we met een relationele databank gaan werken, moeten alle records voorzien zijn van een **primary key** (PK). De databaseomgeving die we hier gebruiken (**hsqldb**) doet de nummering van de PK zelf. Vandaar enkele aanpassingen in de basisklasse (in ons geval `Dictator`):

a. Maak een extra attribuut `id` van het type `int`. Dit wordt de PK.
b. Maak een kopie van de bestaande constructor en voeg een extra `id` parameter toe
c. In de oude constructor, roep je de nieuwe constructor aan met de `id` by default op `-1`
d. Voorzie een getter en setter voor `id`

3.2. Maak een DAO

a. Extraheer de methoden van multiklasse naar een interface persist/<*basis*>Dao (in het voorbeeld `DictatorDao`).
   - In IntelliJ kan je op de multiklasse refactor>extract...>interface gebruiken.
   - Selecteer de voegtoe, verwijder, zoek en gesorteerdOp... methoden.
   - Geef interfacenaam `<basis>Dao` en package `persist`
   - Als IntelliJ je vraagt de interface te gebruiken in bestaande code, mag je de voorgestelde aanpassingen accepteren (bekijk even de aanpassingen).

3.3. Implementeer de interface in een klasse persist/<*basis*>DbDao in het voorbeeld is dit persist.DictatorDbDao:

a. Maak de klasse persist/<*basis*>DbDao als volgt aan (in het voorbeeld is dit persist.DictatorDbDao):
   - Positioneer de cursor in de interface op de naam van de interface, tik <ALT><ENTER> en selecteer implement interface
   - noem de klasse <basis>DbDao en selecteer package persist
   - selecteer alle methoden. klik en OK

3.4. Maak een `Connection`. We gebruiken eenzelfde Connection voor alle databank interactie in de DAO.

a. Voeg aan de DbDao een attribuut Connection toe. Maak een constructor die de databank url als parameter neemt, de connectie met de databank maakt en opslaat in het attribuut.
b. Voeg een close methode aan de DbDAO toe en sluit daar de connectie (als het attribuut niet null is)

3.5. Maak een tabel aan in de DB DAO. Voorzie een methode maakTabel die de database-tabel en alle velden definieert. In ons voorbeeld start dit als volgt:

a. `"CREATE TABLE dictatortable (id INTEGER IDENTITY", naam VARCHAR(30), geslacht VARCHAR(5), ... "`
   Roep maakTabel aan in de constructor van de DAO, nadat de connectie aangemaakt is.
b. Voeg in `maakTabel` voor de create een statement toe dat de tabel dropt als die reeds bestaat. Voor het Dictatorvoobeeld is de SQL instructie: `"DROP TABLE dictatortable IF EXISTS"`

3.6. Maak een klasse `Demo_7` om te testen. In deze klasse voeg je een main methode toe, die volgende stappen uitvoert:

a. Maak de DB DAO aan en geef de URL met het relatief `databasePath` als parameter mee..
b. Zet de code van main in een try/finally blok. In het finally gedeelte roep je de `close` methode van de DAO aan
c. Run en controleer of er effectief databasebestand(en) aangemaakt werd(en) in de map `db`.

3.7. In de klasse `DictatorDao` implementeer je vervolgens de **CRUD**- methoden:

a. **`voegToe (Create)`**: Voeg een databank implementatie toe voor de `voegToe` methode in de DB DAO. Voeg het basisobject toe aan de database (`"INSERT INTO..."`). Maak liefst gebruik van een

`PreparedStatement`, met plaatshouders voor elke attribuutwaarde van de basisklasse.
Opgelet:
- de PK wordt door hsqldb zelf aangemaakt, dus voor dat veld geef je als waarde `NULL` mee.
- een `LocalDate`-waarde moet je omzetten naar een `java.sql.Date` waarde via `Date.valueof(localDate)`.
- een enumtype moet je omzetten naar String via de `name()` methode.
- Zorg ervoor dat elk `(Prepared)Statement` gesloten wordt.

b. **`zoek (Read)`**: Voeg een databank implementatie toe voor deze methode (`"SELECT * FROM… WHERE…"`).
Opgelet:
- als je een `Date`-waarde moet omzetten naar een `localDate`. Dat kan op deze manier:
```
LocalDate myLocalDate = resultSet.getDate("geboorte").toLocalDate();
```
- als je een String moet omzetten naar een enumtype. Dat kan op deze manier:
```
Geslacht geslacht = Geslacht.valueOf(resultSet.getString("geslacht"))
```
- Deze keer lees je ook de PK in en gebruik je de nieuwe constructor die je in 3.1 gemaakt hebt.

c. **`update`**: Als parameter komt het gewijzigde basisobject binnen; als return-value geef je een boolean terug. Zoek het record ahv de PK en wijzig de andere veldwaarden (`"UPDATE… SET… WHERE…"`). Gebruik hier liefst een `PreparedStatement`.

d. **`verwijder (Delete)`**: Voeg een databank implementatie toe voor deze methode.

3.8. In de klasse `Demo_7` testen we alles uit:
a. Instantieer de DAO klasse.
b. `voegToe`: Voeg alle records toe vanuit de klasse `Data`.
c. `zoek`: Zoek een record op naam en druk het af.
d. `update`: Verander de naam van het record, zoek op de nieuwe naam en druk het af.
e. `verwijder`: Verwijder het record met de aangepaste naam doe opnieuw een `zoek` om te controleren of het weg is.

3.9. Implementeer nu ook de sorteermethoden in de interface.
- Volg dezelfde werkwijze als in de vorige methoden. Laat het sorteren over aan de databank (ORDER BY).
- Test in de Demo klasse en toon de records in de juiste volgorde.

3.10. **`filter`**: Voorzie ook een filtermethode die een limietwaarde binnenkrijgt waarop gefilterd zal worden. In ons voorbeeld filteren we alle `Dictator`-objecten eruit waarvan het aantal slachtoffers boven een bepaalde limiet ligt. Don't Repeat Yourself: deze methode voert net als de sorteermethoden een query uit, die een lijst van basisklasse objecten teruggeeft. Stop gemeenschappelijke code in een afzonderlijke methode.
Test in de demo klasse.

3.11. Mogelijke afdruk:

```
Connected to DB
Zoek Mao: Mao Tse-tung           (°1893) China           regime: Communisme
8,0 mln doden
Veranderde naam: Xi Jin-Ping            (°1893) China           regime: Communisme
8,0 mln doden
Xi Jin-Ping verwijderd
Geordend op geboortedatum:
Jozef Stalin          (°1878) Rusland         regime: Stalinisme            45,0 mln doden
Benito Mussolini      (°1883) Italië          regime: Fascisme               2,0 mln doden
Adolf Hitler          (°1889) Duitsland       regime: Nazisme               66,0 mln doden
Francisco Franco      (°1892) Spanje          regime: Militaire dictatuur    5,0 mln doden
```

```
…
Maakten > 4 miljoen slachtoffers:
Ayatollah Khomeini  (°1902) Iran           regime: Fundamentalisme        8,5 mln doden
Jozef Stalin        (°1878) Rusland        regime: Stalinisme            45,0 mln doden
Adolf Hitler        (°1889) Duitsland      regime: Nazisme               66,0 mln doden
Francisco Franco    (°1892) Spanje         regime: Militaire dictatuur    5,0 mln doden
```