GROEIPROJECT – ALGEMENE INFO

De bedoeling van dit "groeiproject" is om een uitgebreid project in IntelliJ te maken dat elke week "meegroeit". Het komt er dus op neer dat je de leerstof van elke week toepast in een nieuw toegevoegde module. Je krijgt hiervoor wekelijks specifieke instructies. Je werkt verplicht met Git en Gradle; daarover krijg je uitleg tijdens de eerste les. Door het versiebeheer in Git kunnen we jouw evolutie opvolgen.
Uiteindelijk moet je het groeiproject voor het examen opleveren. Het wordt dan gebruikt bij je mondeling examen.

Veel succes!

MODULE 1: HERHALING

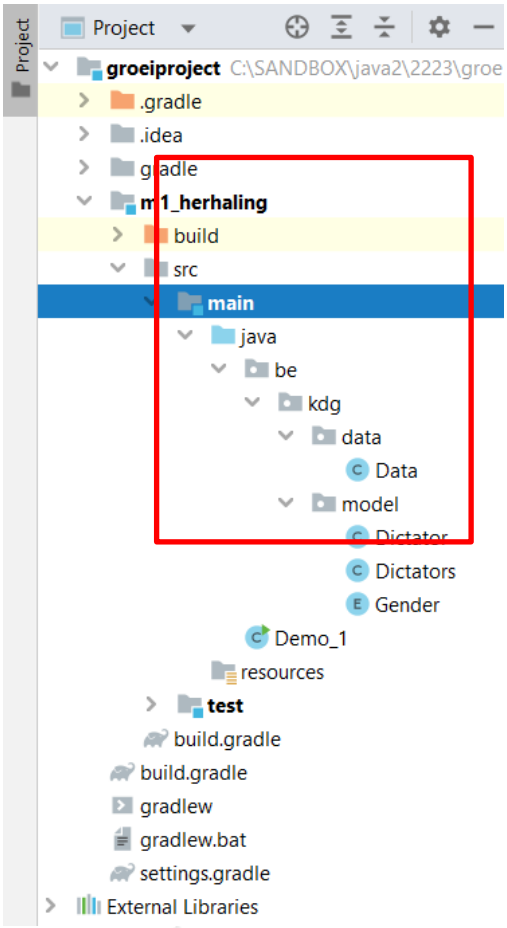
1. VOORBEREIDING

1.1. Kies een **origineel** onderwerp dat je interesseert of dat je goed kent. Bijvoorbeeld: netflixFilms, kamerplanten, trekvogels, racewagens, smartphones, rockbands, planeten, vlinders… De voorwaarde is dat er **niemand anders in jouw klasgroep** hetzelfde onderwerp heeft gekozen.
Als voorbeeld waarnaar we voortdurend zullen verwijzen, kiezen we: **dictators** (biografische data over beruchte en historische dictators zoals bijvoorbeeld Jozef Stalin en Adolf Hitler)
Je moet eerst via het discussiekanaal op MS Teams goedkeuring krijgen voor jouw gekozen onderwerp.
Lees ook de details onder 2.1

1.2. Volg het "Stappenplan_Git_Gradle.pdf" (op Canvas) om je groeiproject helemaal klaar te maken voor Git en Gradle. Uiteindelijk voeg je aan je groeiproject een eerste module toe: m`1_herhaling`.
Let op: de naamgeving van de modules is belangrijk en volgen deze syntax: m `cijfer underscore`

`titel`

1.3. Maak onder de reeds aanwezige map "`src\main\java`" een aparte package: `be.kdg.<projectnaam>`.
Voorzie daaronder een aparte subpackage: `model` met daarin 2 klassen; een **basisklasse** en een **multiklasse**.
De woordkeuze is belangrijk: de multiklasse is het meervoud van de naam van de basisklasse.
In ons voorbeeld heet de basisklasse: **`Dictator`** en bevat de beschrijving van één dictator.
De multiklasse heet: **`Dictators`** en bevat een groep `Dictator`-objecten.



1.4. Als je alle opdrachten van deze week hebt afgerond, zal de uiteindelijke package structuur er zo uitzien:

2. DE BASISKLASSE (IN ONS VOORBEELD: DICTATOR)

2.1. In de basisklasse voorzie je **minstens 6 attributen**. Het is erg belangrijk voor het verdere groeiproject dat daar verschillende datatypes tussen zitten. Zorg voor minstens één attribuut van volgende types:

- `String`
- `double`
- `int`
- een enum-type dat in een **aparte klasse** beschreven wordt (bijvoorbeeld: `Geslacht`)
- een `LocalDate`

Als je beslist hebt, post dan je voorstel op MS-Teams. Doe dat liefst op de volgende manier:
Dictator (name:String, gender:enum, birth:LocalDate, country:String, regime:String, duration:int, cruelty:double, victims:double)

2.2. Voorzie in de basisklasse contructors, getters en setters.

- In de **setters** doe je controle op de nieuwe waarden die via de parameters binnenkomen. Indien verkeerd, doe je een throw van een `IllegalArgumentException` (met gepaste foutmelding).
  In ons voorbeeld kan de naam van een `Dictator` niet leeg zijn, de geboortedatum moet in het verleden liggen, het percentage wreedheid moet in de range 0..1 liggen, enz...
- In de **constructor** komen alle attribuutwaarden via parameters binnen. Roep hier de setters aan voor de controle.
- Voorzie ook een **default-constructor** (=constructor zonder parameters). Daarin roep je de andere constructor op met dummy-waarden (bijvoorbeeld "*Anoniem*" of "*Ongekend*" voor Strings, nul voor getallen, ...)

2.3. Laat het uniek-zijn van een object van de basisklasse afhangen van één of meer attribuut(en); werk dus een gepaste `equals` en `hashCode` methode uit.
→ In ons voorbeeld is elke `Dictator` uniek door zijn naam.

2.4. Maak de basisklasse `Comparable` op hetzelfde attribuut(en) als hierboven (2.3)

2.5. Schrijf in de basisklasse een `toString` methode: gebruik `String.format` om een keurig geformatteerde string aan te maken met de belangrijkste attribuutwaarden. Zorg voor een mooie uitlijning (bekijk daarvoor de voorbeeldafdruk op de volgende pagina onderaan.)
→ Voorbeeld van de klasse `Dictator`:

```
Ayatollah Khomeini   (°1902) Iran      regime: Fundamentalisme     8,5 mln doden
```

3. DE MULTIKLASSE (IN ONS VOORBEELD DICTATORS)

3.1 In de multiklasse maak je gebruik van een ingekapselde **Treeset**. Het uniek-zijn en de volgorde binnen deze set heb je al vastgelegd in de vorige opdrachten (2.3 en 2.4) .

3.2 Voorzie de volgende methoden:

a. **add**: enkel unieke objecten worden effectief toegevoegd. Daar zorgt de set zelf voor (gebruikt achterliggend `equals` en `hashCode` om te controleren)

```
public boolean add(Dictator dicatator)
```

b. **remove**: als parameter komt het attribuut binnen waarvan het uniek-zijn afhangt (zie 2.3)
Opgelet: veilig verwijderen doe je met een Iterator!
In ons voorbeeld is een dictator uniek door zijn naam, dus:
```
public boolean remove(String naam)
```

c. **search**: als parameter komt het attribuut binnen waarvan het uniek-zijn afhangt (zie 2.3).
Returnwaarde: de gevonden `Dictator` of `null`.
```
public Dictator search(String naam)
```

d. **sortedOnXXX**: voorzie minstens 3 methoden die een gesorteerde List teruggeven. Maak daarin gebruik van bijkomende **Comparator** klassen die je als private inner classes implementeert.
In onze voorbeeldklasse `Dictators` maken we 3 extra methoden:
```
public List<Dictator> sortedOnName()
public List<Dictator> sortedOnBirth()
public List<Dictator> sortedOnVictims()
```

e. Voorzie tenslotte ook nog een methode **`getSize`** die het aantal objecten in de set weergeeft.

4. DE DATAKLASSE EN DE MAIN

4.1. Maak een klasse `Data` in een aparte `data`-package. Daarin voorzie je een **static** methode `getData` die een gevulde `List` retourneert. Voorzie **minstens 15** elementen met realistische en uiteenlopende data. In ons voorbeeld dus een `List` met 15 `Dictator`-objecten.

4.2. Maak een main-klasse `be.kdg.Demo_1` waarin je het volgende uittest:
   a. Maak een instantie van de multiklasse.
   b. Vraag een gevulde datalist op (zie 4.1) en voeg al deze objecten toe aan de multiklasse (zie 3.2/a)
   c. Voeg ook eens een dubbel object toe (dat zou niet mogen lukken; zie 2.3)
   d. Test de methoden `search`, `remove` en `getSize` uit (zie 3.2)
   e. Druk de 3 gesorteerde listen af (zie 3.2/d)
   f. Test beide constructors uit en ook de `IllegalArgumentException` (zie 2.2)

5. MOGELIJKE OUTPUT VAN HET DICTATOR-PROJECT (LET OP DE UITLIJNING IN KOLOMMEN):

```
Dictators gesorteerd op naam:
Adolf Hitler           (°1889) Duitsland     regime: Nazisme             66,0 mln doden
Augusto Pinochet       (°1915) Chili         regime: Militaire dictatuur  0,1 mln doden
Ayatollah Khomeini     (°1902) Iran          regime: Fundamentalisme      8,5 mln doden
Benito Mussolini       (°1883) Italië        regime: Fascisme             2,0 mln doden
enz…

Dicators gesorteerd 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
enz…

Dicators gesorteerd op slachtoffers:
Adolf Hitler           (°1889) Duitsland     regime: Nazisme             66,0 mln doden
Jozef Stalin           (°1878) Rusland       regime: Stalinisme          45,0 mln doden
Ayatollah Khomeini     (°1902) Iran          regime: Fundamentalisme      8,5 mln doden
```

```
Mao Tse-tung          (°1893) China          regime: Communisme          8,0 mln doden
enz…
```

6. GIT

- Vraag de **git status** op en bestudeer de rode files.
- Doe een **git add** en vraag opnieuw de status op: groene files.
- Doe een **git commit** met een zinvolle message en vraag opnieuw de status op: "nothing to commit, working tree clean"
- Doe een **git push**
- Check op GitLab:
  - Klik eens door naar "History" en bekijk hoe daar de changes (additions en deletions) worden weergegeven.
  - Bekijk de inhoud van de README.md file:

Vergeet ook niet om je docent (voornaam.achternaam@kdg.be) toe te voegen als **'Reporter'** op je GitLab repository. (via Settings->Members)