PXLJavaAdvanced 2018-2019

Week 2: Opdrachten Java Generics

Github repository: https://github.com/PXLJavaAdvanced1819/week2-generics

Opdracht 1:

a. Pas de generieke klasse WorkingPlace aan zodat je WorkingPlace-objecten kan maken om Motorized objecten te herstellen (dus wel WorkingPlace<Motorized>, WorkingPlace<Car>, WorkingPlace<CircularSaw> maar niet WorkingPlace<Bike> of WorkingPlace<Vehicle>)
b. Pas de generieke klasse WorkingPlace aan zodat je WorkingPlace-objecten kan maken om Vehicle-objecten te herstellen (dus wel WorkingPlace<Car> en WorkingPlace<Vehicle>).
c. Pas de generieke klasse WorkingPlace aan zodat je WorkingPlace-objecten kan maken om Motorized-Vehicle-objecten te herstellen.

We passen nu de grenzen van de parameter in de methode getScore in de WorkingPlaceUtility klasse aan. Test dit telkens uit!

d. Zorg ervoor dat je de methode getScore uit WorkingPlaceUtility enkel kan oproepen voor WorkingPlace<Bike>-objecten.
e. Zorg ervoor de je de methode getScore uit WorkingPlaceUtility enkel kan oproepen voor WorkingPlace objecten die Vehicle objecten herstellen.
f. Zorg ervoor de je de methode getScore uit WorkingPlaceUtility enkel kan oproepen voor WorkingPlace objecten die Motorized objecten herstellen.
g. Zorg ervoor de je de methode getScore uit WorkingPlaceUtility enkel kan oproepen voor WorkingPlace objecten die Motorized Vehicle objecten herstellen.

Opdracht 2:

Gegeven het hoofdprogramma MoveableApp.

Implementeer een generieke interface Moveable met de methoden move() en getCurrentLocation(). Maak vervolgens een klasse Elephant en een klasse ChessPiece die deze interface implementeren. Zorg ervoor dat het hoofdprogramma de volgende output geeft:

```
Elephant: IN_THE_VILLAGE
ChessPiece: c5
```

Opdracht 3:

Maak een abstract klasse **Player** met een member variabele *name*. Voorzie een constructor met *name* als parameter. Maak van deze klasse 3 afgeleide klassen: BaseballPlayer, VolleybalPlayer en SoccerPlayer.

Maak nu een klasse **Team**. Voorzie de volgende eigenschappen: name, played (=aantal wedstrijden gespeeld), won (= aantal wedstrijden gewonnen), lost (= aantal wedstrijden verloren), tied (=aantal wedstrijden gelijkgespeeld) en members (lijst van spelers, gebruik hiervoor een ArrayList). Voorzie enkel getters.

In de **constructor** geef je enkel een naam mee voor het team.

Voorzie de methode **addPlayer** om een speler aan het team toe voegen en een methode **numberOfPlayers** om te vragen hoeveel spelers er in het team zitten.

Kan je spelers met een verschillend type (bv. BaseballPlayer en SoccerPlayer) in één team toevoegen? Test uit! Zorg ervoor dat dit niet (meer) mogelijk is.

Voorzie de methode **matchResult**(Team opponent, int ourScore, int theirScore). Deze methode zorgt ervoor dat voor het Team waarvoor de methode wordt aangeroepen en de opponent het aantal gespeelde, gewonnen, verloren en gelijkspel wedstrijden wordt opgehoogd afhankelijk van de waarden van ourScore en theirScore.

Kan je bovenstaande methode aanroepen voor een team van volleybalspelers tegen een team van baseballspelers? Los dit eventueel op, zodat dat niet langer mogelijk is.

Voeg tenslotte een methode **ranking**() toe. Deze geeft een geheel getal terug waarbij het team 3 punten krijgt voor elke gewonnen wedstrijd en 1 punt voor geen wedstrijd met gelijkspel.