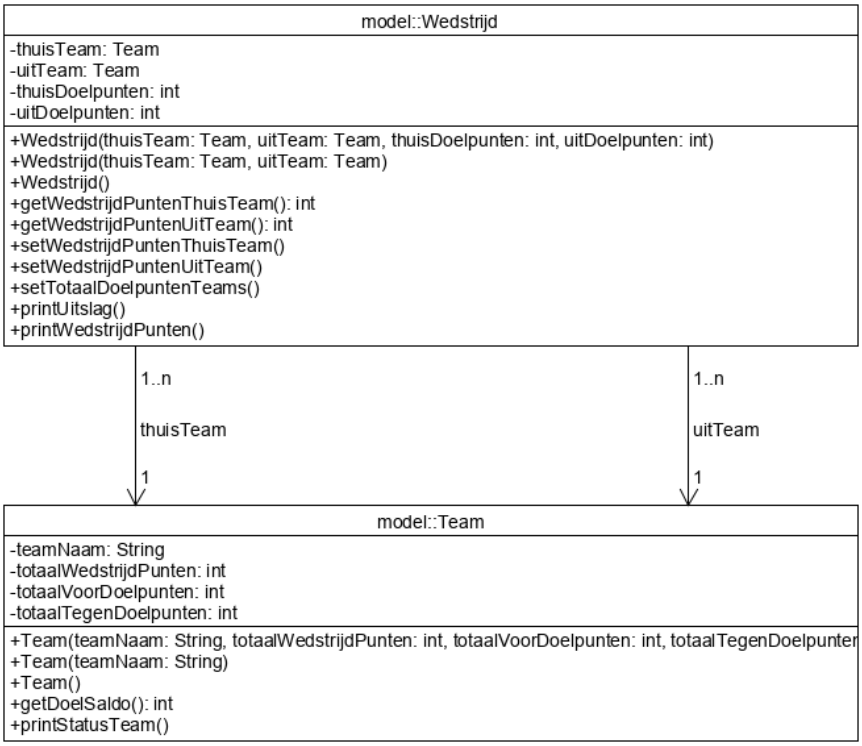
MAKE IT WORK

Opdracht Klassen als attribuut-3 Voetbalscores deel 2

Inleiding

Het doel van deze opdracht is om een applicatie te schrijven die de resultaten van meerdere teams en meerdere wedstrijden vastlegt. Vervolgens kan dan per team het aantal behaalde wedstrijdpunten en het totale doelsaldo worden bijgehouden. De resultaten betreffen een aantal onderlinge wedstrijden van een vast aantal teams. De teams zullen worden bijgehouden in een array. Daarna worden ook de wedstrijden vastgelegd in een array. Tenslotte worden de wedstrijden uit de array doorgerekend en worden telkens de resultaten van de teams bijgewerkt.
Hieronder zie je twee klassen staan die handig zijn voor deze applicatie. In deze klassen zijn alle attributen private gemaakt. Je moet daarom voor alle attributen get- en set-methodes maken. Deze zijn niet opgenomen in het diagram. Verder toelichting volgt in het stappenplan.



Stappenplan

1. Maak een project aan genaamd ‘Voetbalscores2’.
    a. Maak twee packages aan genaamd ‘controller’ en ‘model’.
    b. Maak in de package ‘controller’ een klasse aan genaamd ‘VoetbalScoresLauncher’ en zorg dat deze klasse een main methode bevat.
    c. Maak in de package ‘model’ twee klassen aan genaamd ‘Wedstrijd’ en ‘Team’.

2. De klasse Team moet voldoen aan het klassendiagram.
    a. Declareer de vier attributen, geef ze het juiste datatype en maak ze private.
    b. Maak een all-args constructor voor een Team.
    c. Maak een constructor *Team(String teamNaam)*, waarbij je een teamNaam kunt meegeven, en de overige attributen de default waarde 0 geeft. Zorg voor constructor chaining en roep dus op de juiste manier de all-args constructor aan in deze constructor.
    d. Maak een default constructor die de teamNaam op de lege string (“”) zet en de overige attributen op 0 zet. Maak weer gebruik van constructor chaining.
    e. Maak voor alle attributen een getter en een setter aan.
    f. Maak een methode *getDoelSaldo()* die op basis van totaalVoorDoelpunten en totaalTegenDoelpunten het juiste doelsaldo van het team uitrekent.
    g. Maak een methode *printStatusTeam*() die voor een team een zin toont als: “Het team NAC heeft 7 wedstrijdpunt(en) en een doelsaldo van 13”. Het aantal wedstrijdpunten wordt bijgehouden in het attribuut ‘totaalWedstrijdPunten’. Voor het doelsaldo maakt deze methode gebruik van de methode getDoelSaldo().

3. Test de werking van de klasse Team door in de main methode een object van het type Team aan te maken met behulp van de all-args constructor (bv Team(“Twente”, 15, 28, 12)) en vervolgens met de methode *printStatusTeam()* te laten zien dat de juiste informatie terugkomt.

4. De klasse Wedstrijd moet voldoen aan bovenstaande klassendiagram.
    a. Declareer alle variabelen en maak ze private.
    b. Maak een all-args constructor *Wedstrijd(Team thuisTeam, Team uitTeam, int thuisDoelpunten, int uitDoelpunten)* die de meegegeven teams en getallen kopieert in de bijbehorende attributen. Maak ook de twee andere constructors (zie het klassendiagram) aan en zorg voor constructor chaining.
    c. Maak voor alle attributen een get- en een set-methode aan.
    d. Maak twee methoden *getWedstrijdPuntenThuisTeam()* en *getWedstrijdPuntenUitTeam()* die op basis van de doelpunten teruggeeft of er 3, 1 of 0 punten is verdiend door het thuisTeam, dan wel het uitTeam.
    e. Maak een methode *setWedstrijdPuntenThuisTeam()* en *setWedstrijdPuntenUitTeam()* die voor het thuisTeam en het uitTeam het attribuut totaalWedstrijdPunten bijwerkt. Deze methodes halen de huidige waarde van totaalWedstrijdPunten op en tellen daar de wedstrijdPunten van de wedstrijd bij op. Ze maken daarbij gebruik van de toepasselijke methode *getWedstrijdPuntenThuisTeam()* of *getWedstrijdPuntenUitTeam()*, die in stap 4d. zijn gemaakt.
    f. Maak een methode *setTotaalDoelpuntenTeams()* die voor beide teams op basis van de uitslag de doelpunten voor en de doelpunten tegen aanpast. Let wel: het gaat om 4 aanpassingen!
    g. Maak een methode *printUitslag()* die op basis van de waarde van de attributen de uitslag print: `Uitslag van de wedstrijd Ajax-PEC: 5-0`
    h. Maak een methode *printWedstrijdPunten()* die uitprint: `Dit levert 3 wedstrijdpunten voor Ajax en 0 wedstrijdpunten voor PEC.`

5. Test de werking van je klasse in de main methode van de klasse VoetbalScoresLauncher.
    a. Maak twee objecten van het type Team, waarbij je alleen de naam van het team meegeeft aan en maak een wedstrijd aan tussen deze twee teams waarbij je ook de doelpunten meegeeft.
    b. Roep de printUitslag() en de printWedstrijdPunten() methodes aan op dit object.

6. Bouw nu de uiteindelijke applicatie (basis)
    a. Maak een array ‘teams‘ aan voor 3 Teams.
    b. Bedenk 3 teamnamen en maak voor alle 3 de elementen van de array een object van het type Team aan met steeds één van de 3 teamnamen.
    c. Maak een array ‘wedstrijden’ aan voor 6 Wedstrijden.
    d. Maak voor alle 6 de elementen van de array ‘wedstrijden’ een object van het type Wedstrijd aan tussen de 3 teams in de array ‘teams’. Let op dat je alle 6 mogelijke wedstrijden aanmaakt. Je geeft nog niet de doelpunten mee!
    e. Vraag nu in een for-loop voor alle 6 wedstrijden de voor- en tegendoelpunten van de gebruiker en zorg dat deze aan de juiste attributen van de betreffende wedstrijd worden toegekend. Tip: je kunt ook een variant maken waarin je niet om informatie van de gebruiker vraagt – dit is omslachtig bij het testen – maar waarbij je de voor- en tegendoelpunten uit twee arrays haalt. Maak dan twee int[] arrays aan genaamd *voordoelpunten* en *tegendoelpunten* met daarin steeds zes gehele getallen, die gezamenlijk de uitslagen weerspiegelen.
    f. Toon de gebruiker de uitslag van alle wedstrijden.
    g. Laat van alle wedstrijden de uitslag doorrekenen, dat wil zeggen dat van de teams steeds het totaal aantal wedstrijdpunten, het totaal aantal voor- en tegendoelpunten moet worden bijgewerkt.
    h. Toon de gebruiker van alle teams het totaal aantal wedstrijdpunten en het doelsaldo na doorrekenen van alle wedstrijden.

7. Bouw de applicatie (gevorderden).
    a. Vraag de gebruiker voor hoeveel teams uitslagen moeten worden bijgehouden.
    b. Maak een array aan voor het aantal gevraagde teams.
    c. Vraag de gebruiker om de namen van de teams.
    d. Maak voor alle elementen van de team array een object aan van het type team met één van de teamnamen.
    e. Teams spelen een volledige competitie. Maak een array aan voor alle wedstrijden van de competitie. Bij 3 teams zijn dit 6 wedstrijden. Bij 4 teams zijn dit 12 wedstrijden. De mooiste oplossing is dat je een methode maakt die bij een gegeven aantal teams het juist aantal wedstrijden uitrekent.
    f. Maak nu alle mogelijke wedstrijden aan in de array zonder uitslag. Je kunt het programma dit geheel zelf laten doen door middel van slimme geneste for-loops. Ga bij het testen steeds uit van 3 teams en 6 wedstrijden. Je programma moet echter ook meer teams en wedstrijden aan kunnen.
    g. Vraag de gebruiker in een for-loop achtereenvolgens om de doelpunten voor en de doelpunten tegen van alle wedstrijden. Tip: je kunt ook een variant maken waarin je niet om informatie van de gebruiker vraagt – dit is omslachtig bij het testen – maar waarbij je de voor- en tegendoelpunten uit twee arrays haalt. Maak dan twee int[] arrays aan genaamd *voordoelpunten* en *tegendoelpunten* met daarin steeds net zoveel gehele getallen als er wedstrijden zijn. De twee arrays vormen dan samen de uitslagen. Of nog mooier, genereer de uitslagen random.
    h. Toon de gebruiker van alle wedstrijden de uitslag.
    i. Laat van alle wedstrijden de uitslag doorrekenen, dat wil zeggen dat van de teams steeds het totaal aantal wedstrijdpunten, het totaal aantal voor- en tegendoelpunten moet worden bijgewerkt.
    j. Toon de gebruiker van alle teams het totaal aantal wedstrijdpunten en het doelsaldo na doorrekenen van alle wedstrijden.

Output voorbeeld basis

Voor de duidelijkheid is input die de gebruiker invoert *schuin en onderstreept*.

```
Geef van de wedstrijd VVV-NAC de thuisdoelpunten: 3
Geef van de wedstrijd VVV-NAC de uitdoelpunten: 0
Geef van de wedstrijd VVV-PEC de thuisdoelpunten: 1
Geef van de wedstrijd VVV-PEC de uitdoelpunten: 1
Geef van de wedstrijd NAC-PEC de thuisdoelpunten: 2
Geef van de wedstrijd NAC-PEC de uitdoelpunten: 1
Geef van de wedstrijd NAC-VVV de thuisdoelpunten: 0
Geef van de wedstrijd NAC-VVV de uitdoelpunten: 0
Geef van de wedstrijd PEC-VVV de thuisdoelpunten: 2
Geef van de wedstrijd PEC-VVV de uitdoelpunten: 0
Geef van de wedstrijd PEC-NAC de thuisdoelpunten: 1
Geef van de wedstrijd PEC-NAC de uitdoelpunten: 2
De uitslag van de wedstrijd VVV-NAC: 3-0
De uitslag van de wedstrijd VVV-PEC: 1-1
De uitslag van de wedstrijd NAC-PEC: 2-1
De uitslag van de wedstrijd NAC-VVV: 0-0
De uitslag van de wedstrijd PEC-VVV: 2-0
De uitslag van de wedstrijd PEC-NAC: 1-2
Het team VVV heeft 5 wedstrijdpunten en een doelsaldo van 1
Het team NAC heeft 7 wedstrijdpunten en een doelsaldo van -1
Het team PEC heeft 4 wedstrijdpunten en een doelsaldo van 0
```

Richtlijnen bij coderen (zie ook HBO-ICT code conventions [ICC])

- Zorg dat je naam en het doel van het programma bovenin staan (ICC #1).
- Gebruik de juiste inspringing (indentation) bij de lay-out (ICC #2).
- Let op juist gebruik hoofdletters en kleine letters (ICC #3).
- Gebruik goede namen (ICC #4).
- Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).