MAKE IT WORK

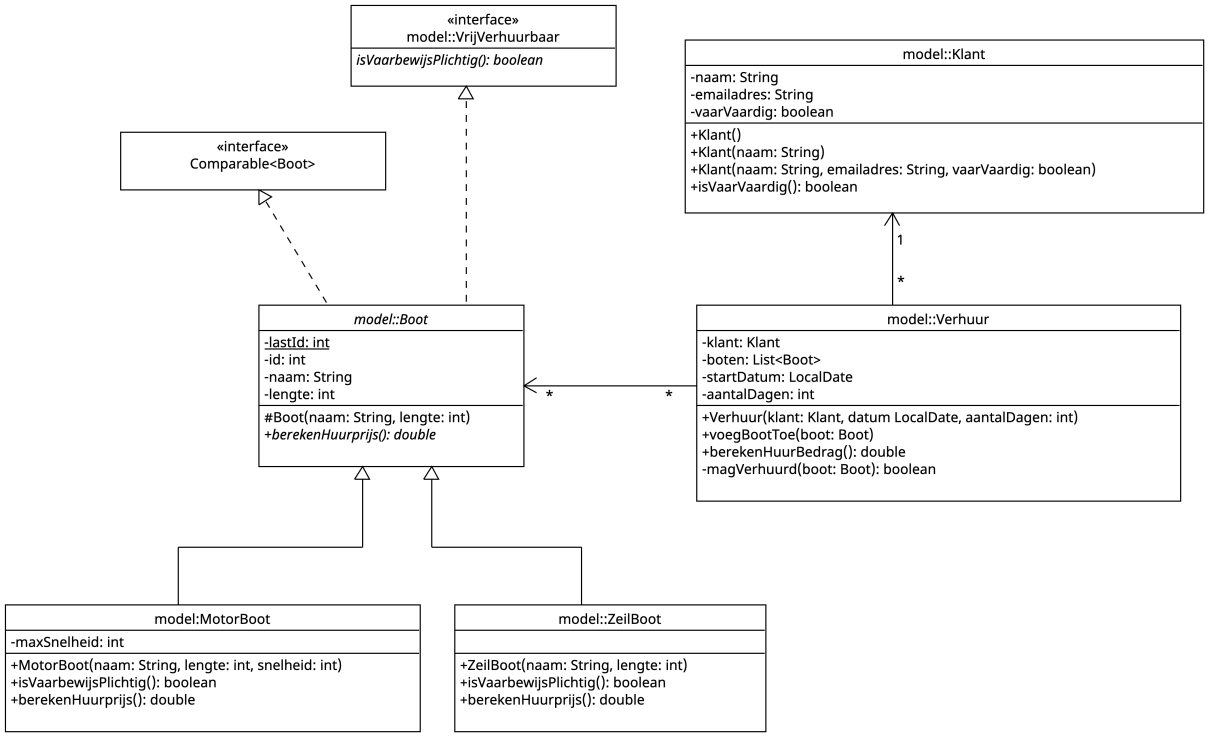
Opdracht Botenverhuur
Bouw klassenstructuur

Inleiding

Een bedrijf verhuurt boten. Er zijn twee typen boten beschikbaar, motorboten en zeilboten. Een klant kan een of meerdere boten voor meerdere dagen huren. Daarbij moet gecontroleerd worden of de klant een boot wel mag huren. Voor sommige boten is een vaarbewijs noodzakelijk, dan moet de klant dit bewijs dus in bezit hebben. Voor de berekening van het verschuldigde huurbedrag wordt gekeken naar de grootte en het type boot. Het doel van deze opdracht is om de klassen te bouwen, die nodig zijn voor de applicatie.

Beschrijving

Het onderstaande klassendiagram moet worden gebouwd. Uiteraard moet dubbele code voorkomen worden.



MAKE IT WORK

Algemene aanwijzingen

- Maak in de package `controller` een Launcher klasse met een eigen `main` methode aan.
- *De Launcher kun je gebruiken om je eigen code te testen. Zie het eind van dit document voor de code die je **moet** laten zien in de Launcher. De overige code hoef je niet te laten zien, maar laat goed werkende code in de Launcher wel staan.*
- In het klassendiagram zijn geen getters en setters getoond. Voeg zelf de getter(s) en setter(s) toe die je nodig denkt te hebben. Je mag niet standaard alle getters en setters opnemen, maar alleen degenen die je nodig hebt.
- Zorg dat je `toString()` methodes dezelfde output genereren als in de outputvoorbeelden.
- Merk op dat datums als datatype de `LocalDate` klasse gebruiken. Je kunt dan handig gebruik maken van de methodes van een `LocalDate` object zoals `now()` voor de huidige datum.
- De interface `Comparable` bestaat natuurlijk al in Java, die hoef je dus niet zelf te maken.
- Als je zelf nog zaken wil toevoegen (omdat je denkt dat ze nodig zijn) die niet in het klassendiagram staan, geef dat dan duidelijk met commentaar aan in de code.

Klasse Klant

1. De default waarde voor naam en e-mailadres is “onbekend” en de default waarde voor vaarVaardig is false.
2. Zorg voor de juiste constructor chaining.
3. Zorg dat de `toString()` methode bijvoorbeeld de volgende uitvoer geeft:

```
Klant Jan de Boer
```

Abstacte klasse Boot en de interface VrijVerhuurbaar

1. De klasse Boot zorgt ook voor het geven van een unieke id. De klasse heeft een klassenvariabele `lastId`, die het laatste uitgegeven id bijhoudt en dus ook hoeveel boten er zijn aangemaakt. De variabele `id` van een boot houdt dan de uiteindelijke id van een boot bij, en die waarde mag niet meer gewijzigd worden.
2. De lengte van de boot moet positief zijn. Geef een foutmelding als de lengte van een ingevoerde boot niet positief is en zet de lengte van deze boot op de standaard lengte van 5 meter.
3. Zorg dat de methode `compareTo()` de boten op naam sorteert. Maak hierbij gebruik van de `compareTo()` methode van de `String` klasse.
4. Van alle boten moet met een toString() methode gemakkelijk de basis informatie getoond kunnen worden zoals hieronder te zien is.

```
Boot 1 met naam De Engel
```

5. Van alle boten is te bepalen of zij te huren zijn zonder een benodigd vaarbewijs. De subklassen van `Boot` bepalen welke voorwaarden daarbij nodig zijn. Daarom is in het diagram de interface `VrijVerhuurbaar` opgenomen. Deze interface schrijft voor dat van motorboten en zeilboten te bepalen moet zijn of een vaarbewijs nodig is om ze te kunnen huren.
   Zorg dat de interface `VrijVerhuurbaar` en de relatie met `Boot` in je project verwerkt is.

MAKE IT WORK

Subklasse MotorBoot

1. Motorboten die langer zijn dan 15 meter of sneller kunnen dan 20 km/u mogen alleen verhuurd worden aan klanten met een vaarbewijs. Implementeer de methode `isVaarbewijsPlichtig()` op basis van deze informatie.
2. Voor motorboten onder de 10 meter wordt een huurprijs van 60 euro per dag gerekend, langere boten kosten 90 euro per dag. Implementeer de methode `berekenHuurprijs()` op basis van deze informatie.
3. Zorg dat de `toString()` methode bijvoorbeeld de volgende uitvoer geeft:

```
MotorBoot 1 met naam De Engel
    Vaarbewijs vereist: Nee
    Huurprijs: 60,00 euro
```

Subklasse ZeilBoot

1. Voor zeilboten langer dan 15 meter is een vaarbewijs noodzakelijk.
2. Zeilboten kosten 40 euro per dag als ze korter zijn dan 7 meter en anders kosten ze 70 euro.
3. Zorg dat de `toString()` methode bijvoorbeeld de volgende uitvoer geeft:

```
ZeilBoot 2 met naam Valk 1
    Vaarbewijs vereist: Ja
    Huurprijs: 70,00 euro
```

MAKE IT WORK

Klasse Verhuur

1. De klasse `Verhuur` heeft alleen een constructor nodig voor het meegeven van de Klant, de datum (startDatum van de verhuurperiode) en het aantal dagen.
2. Er is een methode `magVerhuurd(Boot boot)` die bepaalt of de boot wel verhuurd mag worden aan de klant. Als voor de boot een vaarbewijs verplicht is en de klant heeft geen vaarbewijs dan moet deze methode false terug geven, in alle andere gevallen uiteraard true.
3. Boten worden aan de lijst `boten` toegevoegd met behulp van de methode `voegBootToe(Boot boot)`. Deze methode controleert eerst of de klant de boot wel mag huren en gebruikt daarbij de methode `magVerhuurd()`.
4. Als de boot niet mag worden verhuurd, dan toont de methode de volgende boodschap .

```
MotorBoot 3 met naam SuzyQ
    Vaarbewijs vereist: Ja
    Huurprijs: 90,00 euro
Mag niet verhuurd worden! Toevoegen mislukt.
```

5. De methode `berekenHuurBedrag()` berekent op basis van het aantal dagen verhuur en alle prijzen van de boten het totaal bedrag voor de gehele verhuur.
6. De methode `toString()` in de klasse `Verhuur` moet het volgende doen:
   a. Allereerst moet de methode controleren of er wel boten zijn toegevoegd. Als de lijst leeg is dan moet het volgende getoond worden:

```
De verhuur aan Klant Jan de Boer op 2020-12-03 is mislukt, geen boten
toegevoegd.
```

   b. Als de lijst niet leeg is, dan geeft de methode als eerste de huurinformatie terug, dat wil zeggen, de naam van de klant, de startdatum en de einddatum van de verhuur alsmede de kosten. Zie hierna voor de output. Tip: de startdatum is een LocalDate. Je kunt gebruik maken van de methode `plusDays()` van de LocalDate klasse, waarbij een aantal dagen bij een datum optelt om een nieuwe datum te krijgen. Let wel: in het voorbeeld is sprake van een verhuur van 5 dagen!

```
De verhuur aan Klant Jan de Boer van 2020-12-03 tot 2020-12-07 kost 300,00
euro:
```

   c. Vervolgens toont de methode alle verhuurde boten op volgorde van de naam van de boot.

```
De verhuur aan Klant Van der Valk van 2020-12-10 tot 2020-12-16 kost 1540,00
euro:
MotorBoot 1 met naam De Engel
    Vaarbewijs vereist: Nee
    Huurprijs: 60,00 euro
MotorBoot 3 met naam SuzyQ
    Vaarbewijs vereist: Ja
    Huurprijs: 90,00 euro
ZeilBoot 2 met naam Valk 1
    Vaarbewijs vereist: Ja
    Huurprijs: 70,00 euro
```

MAKE IT WORK

Launcher

Zorg dat deze code in de Launcher aanwezig is:

1. Geef een welkomstboodschap met je naam en studentnummer
2. Laat zien dat een boot een foutmelding geeft als de lengte niet positief is.
3. Laat zien dat een boot niet toegevoegd wordt aan een verhuur als de boot niet verhuurd mag worden.
4. Laat zien dat je meerdere boten kunt huren, dat die gesorteerd getoond worden en dat de totale huurprijs klopt.
5. Laat zien dat alle `toString()` methodes goed geïmplementeerd zijn.

Let op de code conventions

1. Zorg dat je naam en het doel bij elke klasse bovenin staan (ICC #1).
2. Gebruik de juiste inspringing (indentation) bij de lay-out (ICC #2).
3. Let op juist gebruik hoofdletters en kleine letters (ICC #3).
4. Gebruik goede namen (ICC #4).
5. Vermijd magic numbers (ICC#5).
6. Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).
7. Vermijd dode code (ICC #8).
8. Denk aan encapsulation (ICC #9).