UNIVERSITEIT VAN AMSTERDAM 2018-2019

minor programmeren

Oefententamen Programmeren 2

herfst 2018

Vul hier je naam en studentnummer in vóór je begint:	/ 24 p

1. Je mag de vragen in Engels of Nederlands beantwoorden.
2. Dit is een "gesloten boek"-tentamen. Je mag voor het invullen je pen of potlood gebruiken, maar verder niets. Schrijf duidelijk en niet te groot.
3. Leg je studentenkaart (of ander ID met foto) klaar op je tafel. We komen langs om te kijken of je hierboven je naam hebt ingevuld en of deze klopt met je ID.
4. Laat het weten als je kladpapier nodig hebt.
5. Als je vragen hebt over hoe we iets bedoelen, dan kunnen we dat waarschijnlijk niet beantwoorden zonder een deel van het antwoord weg te geven (maar voel je vrij om het te proberen!).
6. Je hoeft geen comments in je code te schrijven.

Functionality

1p **Question 1.**

Write down what is printed when the following program is run:

```
int dim(z, x)
    return z * 2
int gus(z, y)
    return dim(y, z) + 2
int yap(x, z)
    return 6 - gus(z, x)
x = 4
z = 2
print(yap(x, z))
```

1p **Question 2.**

Write down what is printed when the following program is run:

```
int hit(x, z, y)
    return 4 - sal(y, x)
int sal(x, z)
    return 4 * uru(x, z)
int uru(x, z)
    return 2 / x
y = 4
z = 2
x = 4
print(hit(x, y, z))
```

1p **Question 3.**

Write down what is printed when the following program is run:

```
int bad(x, z)
    return mux(x, z) + 5
int dip(x, y)
    return y / 6
int mux(x, y)
    return dip(y, x) - 4
x = 3
z = 3
print(bad(x, z))
```

Football manager

Consider the following class. It is a simple *data class*, used to represent the results of a single football match.

```
class MatchResult():
    def __init__(self, club1, club2, goals1, goals2):
        # TODO
    def won(self):
        """decide which club has won the match"""
        # TODO
    def __str__(self):
        # TODO
```

Additionally, we will define a class that's able to calculate a goal difference for a series of football matches. The goal difference for one club is the number of goals scored in all matches, minus the number of goals conceded in all matches.

```
class GoalDifferenceCalculator:
    def __init__(self):
        self.results = []
    def add(self, match_result):
        """add given MatchResult to list"""
        # TODO
    def get_difference(self, club_name):
        """calculate goal balance for specific club"""
        # TODO
```

And here is some testing code, which should tell you how the class is supposed to be used:

```
if __name__ == "__main__":
    calc = GoalDifferenceCalculator()
    calc.add(MatchResult("ADO", "FEY", 3, 2))   # ADO won
    calc.add(MatchResult("HEE", "NEC", 1, 4))   # NEC won
    calc.add(MatchResult("AJA", "VIT", 0, 2))
    calc.add(MatchResult("HER", "AJA", 0, 2))
    calc.add(MatchResult("TWE", "AJA", 1, 1))
    calc.add(MatchResult("AJA", "AZ",  3, 2))
    print(calc.get_difference("AJA"))           # should print: 1
```

3p **Question 4.**

Implement the `__init__` method for `MatchResult`, saving all incoming data as attributes.

```
def __init__(self, club1, club2, goals1, goals2):
```

3p **Question 5.**

Implement the `won` method for `MatchResult`. Simply return the name of the winning team.

```
def won(self):
```

3p **Question 6.**

Implement the `__str__` method for `MatchResult`. The string should look like: `ADO 3 - 2 FEY`

```
def __str__(self):
```

3p **Question 7.**

Implement the `add` method for `GoalDifferenceCalculator`, saving the object to the `results` list.

```
def add(self, match_result):
```

3p **Question 8.**

Implement the `get_difference` method for `GoalDifferenceCalculator`.

```
def get_difference(self, club_name):
```

Design challenge

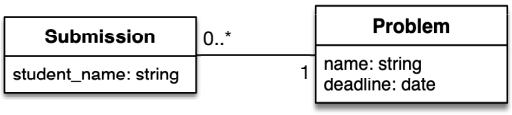
6 p **Question 9.**

In the previous section, you have contributed to a football management system by providing core functionality for the `GoalDifferenceCalculator`. This is only one of the many components that a football management system might comprise.

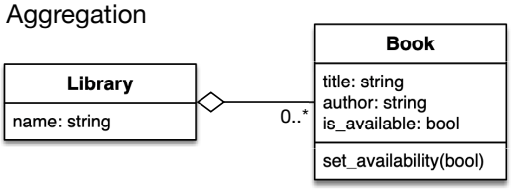
Draw a class diagram for (part of) such a system. Clearly specify associations between related classes. Try to find aggregation and composition relations and indicate those.

Conceptually, not much can be wrong about your class diagram. Simply keep your classes in the domain of football management systems and however you imagine those. The diagram will be graded on clarity (clearly specified classes and relations) and thoroughness (amount, but most of all, quality of classes).

Association



Aggregation



Composition