University of Washington Bothell

CSS 340: Applied Algorithmics

Program 2: Class Design / Operator Overloads

Purpose

This programming assignment will provide exercises in designing classes with proper abstraction and interfaces. Encapsulation and abstraction are key components of Python as well as OOP in general. The programming assignment will also require understanding of operator overloading.

Problem 1: The Vending Bank

Design a class which models the coin-operated "bank" part of a Vending machine which sells snacks. You do not need to implement this class. You only need express the design using a simple UML diagram. Include the diagram in a file (.doc, .docx, or .pdf) in your .zip submission that you turn into Canvas. Here is a start of VendingBank UML diagram with one function already defined.

VendingBank
__id: int
Fill in other data fields required
VendingBank(id: int)
getVendingBankId(): int
Fill in other methods required
…

Problem 2: TimeSpan

Design and implement a **TimeSpan** class which represents a duration of time in seconds, minutes and hours. The order seconds, minutes, and hours should be respected in the constructor.

As an example

```
    duration = TimeSpan(3, 2, 1);
```

is a duration of time of 1 hour, 2 minutes and 3 seconds.

You should store the values as integers in a normalized way but they may be passed in as floats. The stored number of seconds should be between -60 and 60; the stored number of minutes should be between -60 and 60. However, durations can be created with input arguments outside these ranges and you should normalize these. You do not need to worry about integer overflow for very big TimeSpans.

As another example

```
    duration = TimeSpan(90, 2, 1);
```

is stored as a duration of time of 1 hour, 3 minutes and 30 seconds.

Accessor functions required

The **TimeSpan** class should implement the following getters/setters:

def getHours():	return the number of hours as an int
def getMinutes():	return the number of minutes as an int
def getSeconds():	return the number of Seconds as an int

def setTime(seconds, minutes, hours): set the number of hours, minutes, seconds

Constructor

The class should define the constructor so that it can receive both floats and ints. However, the class stores the data as integers so rounding is required.

TimeSpan(-10, 4, 1.5) represents 1 hour, 33 minutes, 50 seconds.

If only one parameter is passed during initialization assume it is a second. If there are two assume seconds and minutes (in that order).

TimeSpan(3, 67) represents 1 hour, 7 minutes, 3 seconds.

Operators

The class must overload and implement the following math operators: **addition, subtraction, and Unary Negation.** The class must make sure that **+=** and **-=** assignment statements as well.

The class must overload and implement the full set of equivalence and comparator operations. For instance, **==**, **<**, **<=**, etc.

I/O

The class must print out a useful representation of itself when passed to the print function.

Output

For formatting use the following:

```
duration = TimeSpan(1,2,3)
print(duration)
```

Should output:

```
Hours: 3, Minutes: 2, Seconds: 1
```

Please use this EXACT format.

Turn In

A **.zip file** which the module named:

- A module names time340.py which has the TimeSpan class
- A doc, docx or .pdf with the UML diagram for VendingBank class

The following is some example code which you may use to make sure that your class is working. Note that the tests below are not comprehensive and you should fully test your TimeSpan class.

```python
from time340 import TimeSpan

dur1 = TimeSpan(3.1, 7)
dur2 = TimeSpan(74, -5, 8)
print(dur1)
print(dur2)
print(dur1 + dur2)
dur2 += dur1
print(dur2)
dur3 = -dur2
print(dur3)

dur4 = TimeSpan(6,7,8)
dur5 = TimeSpan()
dur5.setTime(6, 5, 8)
print("dur4: " + str(dur4))
print("dur5: " + str(dur5))
if dur4 >= dur5:
    print("dur4 is >= than dur5")
else:
    print("dur4 is not >= than dur5")

dur6 = TimeSpan(9, 78, -7)
dur6 = -dur6
print(dur6)
```