Bern University
of Applied Sciences

394661-FS2020-0 - C++ Programming I

EXERCISE-07

TABLE OF CONTENTS

1 Introduction

This exercise of 394661-FS2020-0 will focus on the basic concepts of polymorphism. Polymorphism is the holy grail of object-oriented programming.

You will learn the following topics when completing this exercise:

- Abstract Classes
- Pure Virtual Functions
- Keywords `override` and `final`
- An example of dynamic binding

2 Excercises

Create CMake-Projects with C++ 11 compiler support and Debug/Release build options for the exercise. Add additional files manually to the project to gain full control over the included project files. Separate the implementation from the declaration in a header and source file, respectively.

2.1 Geometrical Objects

In this exercise you will create an abstract base class `Shape` and derived class objects inheriting form `Shape` representing more specific geometrical objects.

1. The base class provides following **pure virtual functions**:
   - `getArea()`, which calculates the area of the respective shape
   - `getCircumference()`, which calculates the area of the shape

   In addition, class `Shape` has a **private** member of type string holding the type of object (`square`, `circle`, `triangle`) and a `report()` function producing an output as shown in the snippet below.
2. Create the derived classes `Triangle` (isosceles), `Square` and `Circle` which implement the methods of the abstract base class.
3. Make sure one can not inherit from `triangle`
4. Make sure the correct destructors are called! How can you achieve this?

Test your classes with the following test program (`ex07.cpp`):

```cpp
#include <iostream>
#include <vector>
#include "triangle.h"
#include "square.h"
#include "circle.h"

int main()
{
    Triangle t1(1,2);
    Triangle t2(3,4);
    Triangle t3(5,6);
    Square s1(1);
    Square s2(2);
    Square s3(3);
    Circle c1(1);
    Circle c2(2);
    Circle c3(3);

    std::vector<Shape*> shapeVec{&t1, &t2, &t3, &s1, &s2, &s3, &c1, &c2, &c3};

    // Range-based for loop (C++11)
    for(const auto& element : shapeVec)
    {
        element->report();
    }
    return 0;
}
```

You should get similar output to this:

```
Triangle has area: 1 and circumference: 5.12311
Triangle has area: 6 and circumference: 11.544
Triangle has area: 15 and circumference: 18
Square has area: 1 and circumference: 4
Square has area: 4 and circumference: 8
Square has area: 9 and circumference: 12
Circle has area: 3.14159 and circumference: 6.28318
Circle has area: 12.5664 and circumference: 12.5664
Circle has area: 28.2743 and circumference: 18.8495
Circle destructor called
Shape destructor called
Circle destructor called
...
..
.
```

3 Submission

Submit your source code (as a zip-file) to Ilias EXERCISE-07 **before the deadline** specified in Ilias.