Programming 2 – Final Exam

13/06/2022

Lasse Loepfe

You have 2 hours to complete the assignment.

● If the code does not compile, the exercise won't be accepted for submission.
● Tests are provided to check if your code is correct. You cannot modify the tests!
● Tests are no warranty for full points, code also has to follow instructions and is expected to be readable, clean, and optimal.
● A skeleton of the exercise is provided. Feel free to add more files and/or include more libraries of you need them.
● To check if your code is correct, you can write the code you need in the main() of the file main.cpp, but leave it unchanged before submitting the exam.

● When you finish, ZIP the whole folder with a filename called "lastname_name.zip" and upload it to the "Final Exam" folder.

● You can use notes, presentations and google stuff. But if you copy from your classmates you will be suspended directly.

Exercise 1 (3 points)

In Ex1.h, implement the classes used to handle weapons in the game Call of Duty. The class hierarchy is the following:

- The abstract base class **Weapon** has these two attributes:
  - const char * name and an integer damage.

  It also has the following public methods:
  - A constructor with two parameters to initialize the name and the damage.
  - A method getName() that returns the name of the weapon.
  - A virtual method getDamage() that returns the damage caused by the weapon
  - A pure virtual method called canUse(), with return type bool.

- A class **Knife** derived from Weapon with the attribute
  - speed (of type integer)

  and the following public methods:
  - A constructor with three parameters to initialize the name, the damage, and the speed.
  - A method getSpeed() that returns the speed of the weapon.
  - Override the method canUse(), returning true;
  - Ovverthe method use(), doing nothing;

- A class **Gun** derived from Weapon with the attributes
  - int lastUse
  - int cadence

  and the following public methods:
  - A constructor with four parameters: the name, the damage, the distance, and the cadence.
  - A method getCadence() that returns the cadence attribute.
  - Overrride the method use, setting lastUse = time(NULL)
  - Override the method canUse(), returning true if the current time is lesser than the last use plus cadence;

Exercise 2 (2 points)

Implement the **Selection Sort algorithm** so that it sorts Circles in order of increasing radius.

Complement the class **Circle** with the following public methods:

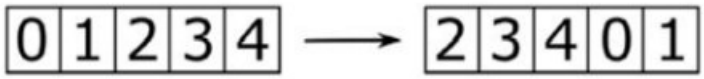
- A method getRadius that returns the radius of the circle
- The overloaded operators “>” that returns true if the radius of the other circle is larger.
- The overloaded operators “<” that returns true if the radius of the other circle is smaller.

The selection sort algorithm that sorts an array of circles in increasing order of radius. This algorithm has to use the overloaded “>” or “<” operators to compare circles.

Exercise 3 (2 points)

Implement the function **circularShift**() that takes an array of integers and a positive integer number. This function has to modify the passed array by shifting the numbers leftwards in a circular fashion, as many positions as indicated by the second parameter.

For instance, if we invoke **circularShift(array,5, 2)**, we need to shift the numbers in array two positions to the left, as depicted by the following image:



Exercise 4 (2 points)

Planet earth will be destructed by 10am today, unless you find the key to deactivate the countdown. The key is stored in a labyrinth, where at every step you have to choose to go left or right. If you always choose wisely, you can find the correct answer.
In a full (complete) binary tree one of of the leave nodes is marked as isExit. That node stores the value of the key to save the world.
Implement the recursive method `GetAccessKey(Waypoint* root, int& val)` so that it sets val to the value stored in the exit node.