Objektorientierte Sprachen

Sommersemester 2021 - Zweite Prüfung

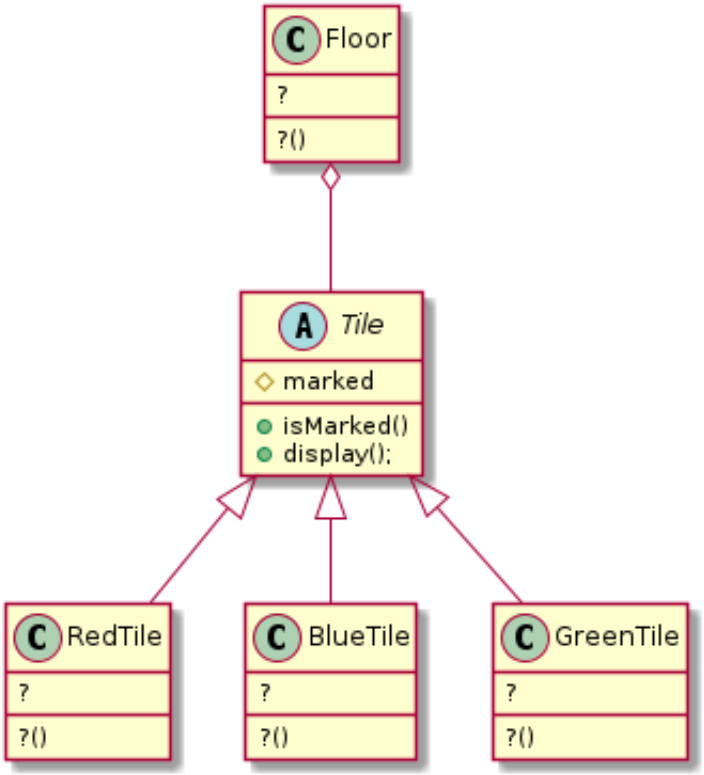
Tasks and submission

Implement a program that evaluates a floor (class Floor) of colored tiles (class Tile) by marking and replaceing tiles till only tiles of a certain color remain.
You are given a basic code structure which contains a main-function as well as the header- and CPP-files for the abstract class Tile. Expand this structure by implementing additional features for main-function and the abstract class and creating three new classes, which inherit from the abstract class.

Put your code files into a ZIP-archive and upload that archive on Moodle. You cannot hand in any submissions afterwards, even if your submission is faulty or incomplete.
The upload is possible until 19:20.

Comments in the code explain the basic code structure, on the following pages you find a class diagram as well as the program's description.

Class diagram



Summary

- In the beginning a new Floor-object is created. During the creation a two-dimensional pointer-array for Tile-objects is initialized. **Implement the constructor for the Floor-class.**
- The Floor-class constructor fills the floor randomly with BlueTile- and GreenTile-objects. Both classes as well as the RedTile class inherit from the abstract class Tile. **Define the pure virtual functions getColor and mark for the class Tile. Implement an inheritance for the three child classes and implement their constructors and getColor and mark functions.**
- Afterwards random tiles will be marked and replaced using the mark- and getColor-function of Tile-objects via the floors's mark- and replace-function until the floor consists only of RedTile-objects. **Implement the mark and replace-function of the Floor-class and the isRed-function to check if the whole floor is red.**
- **Implement the print-function** to display the floor. Adjust the display-function of the Tile-class.
- While running the mark- and replace-function exceptions can occur. **Implement a correct error handling in the main-function.**
- Finally the memory has to be managed correctly. **Implement the destructor for the Floor-class.**

Tasks

main-function

- Implement error handling via exceptions for the mark- and replace-function's call. Catch invalid_argument-exceptions as well as other objects.

Abstract class Tile

Define the following **purely virtual** function for the abstract class:

- **getColor():** The function returns a single character.
- **mark():** The function sets the marked-Value.

Adjust the display-function so that the corresponding color is printed instead of X.

Class RedTile

Define the class RedTile which inherits from Tile in a header- and CPP-file. Implement the following functions for the class:

- **Constructor:** The marked value of the object should be set to false.
- **mark():** The function sets marked to false.
- **getColor():** The function should return the character 'R'.

Class BlueTile

Define the class BlueTile which inherits from Tile in a header- and CPP-file. Implement the following functions for the class:

- **Constructor:** The marked value of the object should be set to false or true with a 50:50 probability.
- **mark():** The function sets marked to true.
- **getColor():** The function should return the character 'B'.

Class GreenTile

Define the class GreenTile which inherits from Tile in a header- and CPP-file. Implement the following functions for the class:

- **Constructor:** The marked value of the object should be set to true.
- **mark():** The function sets marked to true.
- **getColor():** The function should return the character 'G'.

Class Floor

Define the class Floor in a header- and CPP-file. The class contains a two-dimensional array consisting of Tile-pointers. These pointers are used to access the RedTile-, BlueTile- and GreenTile-objects via polymorphism. Implement the following class functions:

- **Constructor:** The constructor creates a 3x3 floor placing blue and green tiles randomly in it with a 50:50 probability.
- **Destructor:** The destructor releases the memory, which was allocated by the constructor.
- **print():** Displays the floor in a 3x3 grid by calling the display-function for each Tile-pointer.
- **mark(int x, int y):** If one of the parameters is invalid (too large or too small) an invalid_argument exception is thrown.
  The mark function of the tile at the coordinates x/y of the floor is called.
- **replace(int x, int y):** If one of the parameters is invalid (too large or too small) an invalid_argument exception is thrown.
  If there is a green tile at the x/y coordinates of the floor, it will be replaced by a new blue tile.
  Otherwise, if there is a blue tile at the coordinates x/y of the floor and it is marked, it will be replaced by a new red tile.
  Nutzen Sie die Funktionen getColor und isMarked der von Tile vererbten Klassen.
  Use the getColor and isMarked functions of the classes inherited from Tile.
- **isRed():** The function should check whether the floor only consists of RedTile objects and accordingly return true if this is the case, otherwise false.

Grading aspects

Aspect	Grading
main - Error handling (Exceptions)	10%
Tile - getColor and mark	7%
RedTile - Header-/CPP-files	1%
RedTile - Inheritance+constructor	4%
RedTile - getColor und mark	3%
BlueTile - Header-/CPP-files	1%
BlueTile - Inheritance+constructor	4%
BlueTile - getColor und mark	3%
GreenTile - Header-/CPP-files	1%
GreenTile - Inheritance+constructor	4%
GreenTile - getColor und mark	3%
Floor - Header-/CPP-files	3%
Floor - Constructor	10%
Floor - Destructor	8.5%
Floor - print	7.5%
Floor - mark	2%
Floor - mark-Exception	3.5%
Floor - replace	6%
Floor - replace-Exception	3.5%
Floor - replace-Memory management	7.5%
Floor - isRed	7.5%
Total	100%

Compiler warnings, memory errors and bad coding style can result in a deduction of points.

Grading

$\geq 88\%$	Excellent
$\geq 75\%$ and $< 88\%$	Good
$\geq 63\%$ and $< 75\%$	Satisfactory
$\geq 50\%$ and $< 63\%$	Sufficient
$< 50\%$	Unsatisfactory