Piscine C++ - d14a

Kluedoala

Alexandre “Bibi” BOURLON bourlo_a@epitech.eu
Pierre-Yves “Belga” LEFEUVRE lefeuv_p@epitech.eu
Adrien AUBEL aubel_a@epitech.eu

Abstract: This document is the subject of d14a - Kluedoala

Contents

Chapter I

GENERAL RULES

- READ THE GENERAL RULES CAREFULLY !!

  ◦ You will have no possible excuse if you end up with a 0 because you didn't follow one of the general rules

- GENERAL RULES :

  ◦ If you do half the exercises because you have comprehension problems, it's okay, it happens. But it you do half the exercises because you're lazy, and leave at 2PM, you WILL have problems. Do NOT play with fire.

  ◦ Every function implemented in a header, or unprotected header, means 0 to the exercise.

  ◦ Every class MUST have a constructor and a destructor.

  ◦ Every output goes to the standard output, and will be ended by a newline, unless specified otherwise.

  ◦ The imposed filenames must be PRECISELY respected, as well as class, function and method names.

  ◦ Remember : You're coding in C++ now, and not in C. Therefore, the following functions are FORBIDDEN, and their use will be punished by a -42, no questions asked :

    ∗ `*alloc`

    ∗ `*printf`

    ∗ `free`

    ∗ `open, fopen, etc ...`

- Files associated with a class will be CLASS_NAME.hh and CLASS_NAME.cpp (If applicable), unless specified otherwise.
- Turn-in directories are ex00, ex01, ..., exN
- Any use of `friend` will be punished by a -42, no questions asked.
- Read the examples CAREFULLY. They might require things the subject doesn't say ...
- These exercises require that you create lots of classes, but most of them are VERY short. So, don't be lazy !
- Read ENTIRELY the subject of an exercise before you start it !
- THINK. Please.
- THINK. By Odin !
- T.H.I.N.K ! For Pony !

- COMPILATION OF THE EXERCISES :

  - The Koalinette compiles your code with the following flags : `-W -Wall -Werror`
  - To avoid compilation problems with the Koalinette, include every required headers in your headers.
  - Note that none of your files must contain a `main` function. We will use our own to compile and test your code.
  - This subject may be modified up to 4 hours before turn-in time. Refresh it regularly !
  - The turn-in dirs are `(SVN REPOSITORY - piscine_cpp_d13-promotion-login_x/exN)` , N being the exercise number

Chapter II

Exercise 0

Exercise : 00	points : 5
Exercise 0	
Turn-in directory: (SVN REPOSITORY - piscine_cpp_d14a-promo-login_x)/ex00	
Compiler: g++	Compilation flags: -W -Wall -Werror
Makefile: No	Rules: n/a
Files to turn in : ex00.cpp, ex00.hh	
Remarks : The file "ex00-partial.hh" is provided; you must fill it and turn it in as "ex00.hh"	
Forbidden functions : malloc - free	

You must fill the Weapon, Suspect, and Room classes declarations. Then you will implement the Card, Weapon, Suspect, and Room classes.

For the record, the Weapon, Suspect, and Room are subclasses of Card . So they are cards.

Cards involved in the crime are stored in the

```
static const Name PartOfTheCrime
```

member of each class (one by class, a total of three: a weapon, a suspect and a room).

These statics will be set by us, for the correction tests.

Now you must, according to the declared class methods, add one or multiple attributes to each class.

The Weapon class has the "bool BearsFingerprints() const" method, the Suspect class has the "bool IsLying() const" method, and the Room class has the "bool Has SecretPassage() const" method. This should help you to choose which attributes you have to add to your classes.

In order to set the "isPartOfTheCrime" property, you just have to check if the "static const name PartOfTheCrime" matches with the one you are initializing.

The BearsFingerprints, IsLying and HasSecretPassage attributes don't refer to the PartOfTheCrime concept. They are specific to the class and so are not part of any base class.

Implement the `Game` class, or more precisely the static member function

```
bool CaseSolved(Weapon* w[], Suspect* s[], Room* r[])
```

This member function returns `true` if the arrays given as parameters contain the weapon, the suspect and the room involved in the crime.

Arrays are terminated by a NULL pointer.

The three elements must be all present in order to solve the case.

You will create and implement the following classes

- `WeaponException`
- `SuspectException`
- `RoomException`

out of any namespace.

These will be thrown if (and only if) the array corresponding to the class doesn't contain the crime element while the two other arrays contain the crime element. Example:

- `Weapon *w` → contains the `PartOfTheCrime` weapon.
- `Suspect *s` → contains the `PartOfTheCrime` suspect.
- `Room *r` → does not contain `PartOfTheCrime` room.

A `RoomException` exception is thrown.

All of these three classes will have as private attributes:

- `std::string _message;` which will be set to: "[CLASS] ne contient pas la bonne carte!"
- a copy of the array which was responsible for the thrown exception.

"[CLASS]" must of course be replaced by Weapon, Suspect or Room according to the class name.

The array will be given as a parameter to the class constructor.
All of the three classes will also have member functions allowing to read their attributes:

- `std::string const & getMessage() const;`
- `TYPE **getCards() const;` (Replace "TYPE" with `Weapon`, `Suspect` or `Room` according to the class name).

For now you don't have to worry about how the try/catch will be performed... But you must know that it takes a pointer on the exceptions.

Chapter III

Exercise 1

Exercise : 01	points : 5
Exercise 1	
Turn-in directory: (SVN REPOSITORY - piscine_cpp_d14a-promo-login_x)/ex01	
Compiler: g++	Compilation flags: -Wextra -Werror -Wall
Makefile: No	Rules: n/a
Files to turn in : ex01.cpp, ex01.hh	
Remarks : The file "ex01-partial.hh" is provided; you must fill it and turn it in as "ex01.hh"	
Forbidden functions : malloc, free	

The `Card` now has a new property `CardType.` This property must be correctly set at a `Weapon,` `Suspect,` or `Room` initialization.

Implement also the overload

```
bool CaseSolved(Card* c[])
```

in the `Game` class, which will return `true` only if the array given as a parameter contains the three `PartOfTheCrime.`

You will now create the `CardException` class.
This class inherits from `std::exception` .
You will replace the member function `getMessage` by the implementation of `what()` .
The output will be the following:

```
Aucune des cartes ne correspond!
```

This exception will be thrown if the array given as a parameter to `CaseSolved` does not contain any card involved in the crime. This array will be stored in the `CardException` instance, given to the constructor as a parameter.
Again, you will have to `throw` a pointer on the exception. No other exception should

be thrown in this exercise.

You will also implement the member function `Card** getCards() const` , that will return the array stored in the instance.

Chapter IV

Exercise 2

Exercise : 02	points : 5
Exercise 2	
Turn-in directory: `(SVN REPOSITORY - piscine_cpp_d14a-promo-login_x)/ex02`	
Compiler: `g++`	Compilation flags: `-Wextra -Werror -Wall`
Makefile: `No`	Rules: `n/a`
Files to turn in : `ex02.cpp, ex02.hh`	
Remarks : `The file "ex02-partial.hh" is provided; you must fill it and turn it in as "ex02.hh"`	
Forbidden functions : `malloc - free`	

In the Kluedoala, weapons are represented by a card and a figure. Suspects are represented by a card and a piece (a pawn) representing the player.

So make sure that `Weapon` and `Suspect` classes inherit from classes corresponding to their representations, respectively `Figurine` and `Piece` .

- The `Piece` has a position on the game board.
  - This position will have a value between 1 and 100.
  - It will be accessible through a getter `int Position() const`
  - and a setter `void Position(int position).`
- It will also be configured during the instantiation via its constructor.
- The starting position is 1 and must be defined directly by the inheriting class.

The initialization list is your friend :p

As we occasionally lose figures in a board game like Cluedo, the `Figurine` class will implement the `bool IsLost() const` method, allowing to know if the figure has been lost over the course of the years by your big brother :)

`bool IsLost()` is a Getter.

This property is configured by the constructor and must be set externally by the inheriting class' constructor.

You will now create the `GamException` class. This class will inherit from the `std::exception` class.

You will then create the classes:

- `FigurineException`
- `PieceException`

which will both implement the `GameException` class.

The `what` implementations will return:

- "Lost Figurine" // for the `FigurineException` class
- "Piece is not on the board" // for the `PieceException` class

`FigurineException` will be thrown if the figure is the crime weapon, and is lost as we try to use it in `CaseSolved(Card **)` . It will contain a `Figurine` pointer and the member function `Figurine *getFigurine() const;`

`PieceException` will be thrown if we try to provide a `Piece` having a position set to a value out of the 1-100 range. It will contain a `Piece` pointer and the member function `Piece *getPiece() const;`

All thrown exceptions will be compatible with the following `try/catch` block:

```
try
{
        ....
}
catch (std::exception* e)
{
        std::cout << e->what() << std::endl;
}
```

To sum it all up:

- Complete the class declaration and implementation for `Piece` and `Figurine`.
- Implement the `int Position() const` method in `Piece` .
- Implement the `void Position(int position)` method in `Piece`.
- Complete the implementation of `Weapon` and `Suspect`.
- Modify `Suspect` 's constructor to set the position to the default value.
- Modify `Weapon` 's constructor to set the `IsLost` property to the user-defined value.

Chapter V

Exercise 3

Exercise : 03	points : 5
Exercise 03	
Turn-in directory: `(SVN REPOSITORY - piscine_cpp_d14a-promo-login_x)/ex03`	
Compiler: `g++`	Compilation flags: `-Wextra -Werror -Wall`
Makefile: `No`	Rules: `n/a`
Files to turn in : `ex03.hh, ex03.cpp`	
Remarks : `The file "ex03-partial.hh" is provided; you must fill it and turn it in as "ex03.hh"`	
Forbidden functions : `malloc - free`	

Rewrite the static method `bool CaseSolved(Card *c[])` from the exercise 2, but this time without using the `CardType`. Moreover this method must not throw exceptions anymore.

⚠ `This doesn't mean that you just have to remove your throw (exception) from the functions body...`

Summary:

- Re-implement the static `bool CaseSolved(Card* c[])` method from the Cluedo class.

Chapter VI

Exercise 4

Exercise : 04	points : 5
Exercise 4	
Turn-in directory: (SVN REPOSITORY - piscine_cpp_d14a-promo-login_x)/ex04	
Compiler: g++	Compilation flags: -Wextra -Werror -Wall
Makefile: No	Rules: n/a
Files to turn in : ex04.cpp, ex04.hh	
Remarks : The file "ex04-partial.hh" is provided; you must fill it and turn it in as "ex04.hh"	
Forbidden functions : None	

Fill the Weapon, Suspect, and Room declaration and implementation according to the new method in Card .

```
virtual bool SpecialAbility() const
```

This method behaves the same way than the BearsFingerprints , IsLying , and HasSecretPassage methods from classes defined before.

If it behaves the same way, you should not have so much to code.

You will also edit the WeaponException , SuspectException , and RoomException classes in order to make them inherit directly from std::exception.

Your what implementations will return the following information:

```
This Weapon bears finger prints. But it's not a part of the crime!
This Suspect is lying. But he's not part of the crime!
```

```
This Room has a secret passage. But it's not part of the crime!
```

These exceptions will be respectively thrown by the `BearsFingerPrints` , `IsLying` , and `HasSecretPassage` methods. And only by these member functions.
Of course, the thrown exceptions will be of `std::exception*` type.

The BearsFingerPrints, IsLying, and HasSecretPassage are const and must stay as such. Think carefully about how you will build these exceptions.

Summary:

- Fill the `Weapon`, `Suspect`, and `Room` classes declaration and implementation.
- Edit the `WeaponException` , `SuspectException` , and `RoomException` classes.

WINNING