



C++ Pool - d07a

Koalab koala@epitech.eu

Abstract: This document is the subject for d07a

Contents

I	Basic Rules	2
II	Exercise 0	4
III	Exercise 1	8
IV	Exercise 2	12
V	Exercise 3	17
VI	Exercise 4	22

Chapter I

Basic Rules

- BASIC RULES:

- If you do half the exercises because you have comprehension problems, it's okay, it happens. But if you do half the exercises because you're lazy, and leave at 2PM, you WILL have problems. Do NOT tempt the devil.
- Every function implemented in a header, or unprotected header, means 0 to the exercise.
- Every class must possess 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 followed TO THE LETTER, 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 question asked :

- * `*alloc`

- * `*printf`


- * `free`

- Generally, Files associated to a class will always be `CLASS_NAME.h` and `CLASS_NAME.cpp` (class name can be in lower case if applicable).
- Turn in directories are `ex00`, `ex01`, ..., `exN`
- Any use of `"friend"` will result in the grade -42 no questions asked.

- You must read the examples thoroughly. They can contain requirements that are not obvious in the exercises description.
 - You are asked to turn in an important number of classes. However, these classes are pretty short. Slackers not accepted!
 - Read each exercise FULLY before starting it!
 - USE YOUR BRAIN, Please!
-
- EXERCISES COMPILATION :
 - The Koalinette compiles your code with the following flags : `-W -Wall -Werror`
 - To avoid Koalinette compilation problems, please include necessary files within your include (*.hh).
 - Please note that none of your files must contain the `main` function except if it is explicitly asked. We will use our own `main` function to compile and test your code.
 - Remember, you are writing C++ code now. Thus the compiler is g++!
 - The exercise description can be modified until 4h before the final turn in time! It is advised to refresh it regularly!
 - The turn in directory is: `(piscine_cpp_d07a)/exN` (N being the exercise number).

Chapter II

Exercise 0

	Exercise : 00	points : 3
Meeeeeeedic		
Turn-in directory: (piscine_cpp_d07a)/ex00		
Compiler: g++	Compilation flags: -Wall -Wextra -Werror	
Makefile: No	Rules: n/a	
Files to turn in : Skat.h, Skat.cpp		
Remarks : n/a		
Forbidden functions : malloc, free, *printf, pointeurs		

Soldiers, welcome to the SKAT (Special Kreog and Tactical). If you are here, it is because you want to protect our wonderful country, Australia, against this crawling brood, named Dingos. This infamous rot won't rest until they undermine the bases of our wonderful country.

I am the drill sergeant Hartog and I am in charge of making war dogs out of you.

But before you show them what you're worth, you are now at the level zero of your life on earth.

Let's starts the briefing.

As you will very likely get a bullet in your skin on the battle field, our scientist team designed a special package that will save your butt more than you would think: the stimpak. This stimpak is able to repair your bones, suture clean wounds and so on. As long as you have one left, you have a chance to stay alive and to come back home.

Implement the following class :

```
class Skat
{
public:
    Skat(std::string const& name, int stimPaks);
    ~Skat();

public:
```

```
        [...]      stimPaks();
        const std::string& name();

public:
    void    shareStimPaks(int number, [...] stock);
    void    addStimPaks(unsigned int number);
    void    useStimPaks();

public:
    void status();

private:
    [...]
};
```

Explanations :

- A Skat has a name represented by a string, and a number of stimpaks. By default, your Skat's name is bob, and he possesses 15 stimpaks.
- The member function `stimPaks` returns the number of stimpaks currently in possession of your Skat.
This member function allows you to change your unit's number of stimpaks from outside the class.
- The member function `name` returns your unit's name. This member function doesn't modify the calling instance.
- The member function `shareStimPaks` allows you to provide extra stimpaks to a teammate needing it. This member function increments of `[number]` the `[stock]` of stimpaks.
You must then decrement your stock of `[number]` stimpaks. If the number of stimpaks asked is too big, you have to display :

```
1 Don't be greedy
```

on the standard output and then do nothing. Otherwise, display :

```
1 Keep the change.
```

- The member function `addStimPaks(unsigned int number)` adds `[number]` stimpaks to those that your unit already own. If `[number]` is equal to 0, display

```
1 Hey boya, did you forget something ?
```

on the standard output.

- Your unit can use a stimpaks thanks to a call to the member function `useStimPaks()` . It displays :

```
1 Time to kick some ass and chew bubble gum.
```

on the standard output, if this is possible. Otherwise, display :

```
1 Mediiiiiic
```

- A unit can give you its status at any point through a call to the member function `status` . A unit communicates the following way :

```
1 Soldier [NAME] reporting [NUMBER] stimpaks remaining sir !
```

where `[NAME]` is the name of the unit and `[NUMBER]` its stimpaks number.
This member function can be called on immutable Skat objects.

The following code must compile and display the following output:


```
1 int main()
2 {
3     Skat s('Junior', 5);
4
5     std::cout << "Soldier " << s.name() << std::endl;
6
7     s.status();
8
9     s.useStimPaks();
10
11     return 0;
12 }
```

Output:

```
1 Kinder:ex00$ g++ -W -Wall -Werror *.cpp
2 Kinder:ex00$ ./a.out | cat -e
3 Soldier Junior$
4 Soldier Junior reporting 5 stimpaks remaining sir !$
5 Time to kick some ass and chew bubble gum.$
6 Kinder:ex_0$
```


Chapter III

Exercise 1

	Exercise : 01	points : 2
KoalaBot, Assemble		
Turn-in directory: (piscine_cpp_d07a)/ex01		
Compiler: g++	Compilation flags: -Wall -Wextra -Werror	
Makefile: No	Rules: n/a	
Files to turn in : KoalaBot.h, KoalaBot.cpp, Parts.h, Parts.cpp		
Remarks : n/a		
Forbidden functions : malloc, free, *printf		

Gentlemen, this is the KoalaBot Kreog mk5 (the 4 previous versions tend to burst into flames once in a while). This new prototype will be your back up when the situation gets too tough for you (A recon mission in a mine field for example).

The KoalaBot possesses 3 removable parts : arms, legs as well as a head. (The first to say: Sir, he has 2 arms and legs, I break his knees).

These different parts are grouped together in files `Parts.h,cpp` and they possess the following names : `Arms` , `Legs` , `Head` .

All these parts are built according to the following model :

- Each class possesses a constructor with the following signature:
`Constructor(std::string [...] serial, bool fonctionnal);`
- They must also possess a serial represented by a string, as well as a boolean indicating if it is functional or not.

* The Arms class has a default serial of “A-01”

- * The Legs class has a default serial of “L-01”
- * The Head class has a default serial of “H-01”

These class are all functional by default.

- o They should possess the following public member functions:

- * `bool isFunctionnal()` which indicates if the piece is functional or not.
- * `std::string serial()` which returns the serial of the current piece.
- * `void informations()` which displays :

```
1 [Parts] [PARTSTYPE] [SERIAL] status : {OK | KO}
```

Preceded by one tabulation and followed by a newline.

- [PARTSTYPE] must be replaced by the name of the class.
- [SERIAL] is the serial of the current piece.
- if the piece is working fine (functional), then display “OK” otherwise “KO”



All these member functions don't modify the calling instance.

Now that we have the different pieces, let's put them together.

Create the `KoalaBot` class

- o This class is composed of an instance of each pieces created previously. (`Arms` , `Legs` et `Head`).
- o It possesses a serial that is stored as a string, with the default value of “Bob-01”.
- o It possesses a member function `setParts` that accepts as a parameter a reference on a constant piece.



It can be called with any class included in the file `Parts.h`

- It possesses a member function `swapParts` that accepts as a parameter a reference on a piece. This piece will be swapped with the piece of the `KoalaBot`



It can be called with any class included in the file `Parts.h`

- It possesses a member function void `informations()` that displays

```
1  [KoalaBot] [SERIAL]
```

Followed by a newline, `[SERIAL]` is the serial of the `KoalaBot`, followed by information on each pieces of the `KoalaBot` in the following order:

- * Arms
- * Legs
- * Head



This member function does not modify the calling instance.

- It possesses the member function `bool status()` that returns `true` if all the pieces of the `KoalaBot` are running, otherwise it returns `false`



This member function does not modify the calling instance.

The following code must compile and display the following output:


```
1 int main()
2 {
3     KoalaBot kb;
4
5     std::cout << std::boolalpha << kb.status() << std::endl;
6
7     kb.informations();
8
9     return 0;
10 }
```

Output:

```
1 Kinder:ex01$ g++ -W -Wall -Werror *.cpp
2 Kinder:ex01$ ./a.out | cat -e
3 true$
4 [KoalaBot] Bob-01$
5     [Parts] Arms A-01 status : OK$
6     [Parts] Legs L-01 status : OK$
7     [Parts] Head H-01 status : OK$
8 Kinder:ex_1$
```

Chapter IV

Exercise 2

	Exercise : 02	points : 5
Houston, we have a problem		
Turn-in directory: (piscine_cpp_d07a)/ex02		
Compiler: g++	Compilation flags: -Wall -Wextra -Werror	
Makefile: No	Rules: n/a	
Files to turn in : KreogCom.h, KreogCom.cpp		
Remarks : n/a		
Forbidden functions : malloc, free, *printf		

You are a team as close to each other as Parisians in the subway during rush hour. And this, when you are taking a well deserved rest in the barracks as well as when you're being spread all over the jungle. In order to communicate and most importantly to locate your teammates, this is the pinnacle of our technology: the `KreogCom`. This technological jewel locates your teammates in real time.

The `KreogCom` works in the following way:

```
class KreogCom
{
    public:
        KreogCom(int x, int y, int serial);
        ~KreogCom();

    public:
        void addCom(int x, int y, int serial);
        KreogCom *getCom();
        void removeCom();

    public:
        void ping();
        void locateSquad();

    private:
        [...]
```

```
private:
    [...]

private:
    const int m_serial;
};
```

Explanation:

- A KreogCom possesses a serial represented by a constant integer, as well as an X position and a Y position.



It is up to you to add member data, but this data must be private.

- It possesses a constructor allowing to create a `KreogCom` , at the position (`[x]` , `[y]`), and with a serial `[serial]`
- It possesses a member function `addCom` that creates a new `KreogCom` and links it to the actual `Com` . If the current `KreogCom` is not linked to any `KreogCom` , then it is linked with the `KreogCom` recently created (see below).

(---> = links to)

```
-----
| this | ---> | new KreogCom x, y, serial |
-----
```

It the current `KreogCom` is already linked to another `KreogCom` , then the new `KreogCom` will replace it. (see below)

(---> = links to)

```
-----
| this | ---> | new KreogCom x, y, serial | ---> | KreogCom that was linked to this |
-----
```

- It possesses a member function `getCom` that returns a pointer on the `KreogCom` linked to our `KreogCom` . If it is not linked, this member function returns 0.
- It possesses a member function `removeCom` that removes the linked `KreogCom` .



Be very careful not to break the communication chain.

- It possesses a member function `ping` , that displays the following information on the standard output (followed by a new line):

```
1 KreogCom [SERIAL] currently at [X] [Y]
```

[SERIAL], [X] and [Y] are the member data of your `KreogCom`



This member function doesn't modify the calling instance.

- It possesses a member function `locateSquad` that displays information about all linked `KreogCom` , and then its own information.

```
1 [Displays info on linked KreogCom]
2 [Displays info on current KreogCom]
```



This member function doesn't modify the calling instance.



During the destruction of a `KreogCom`, the communication chain must not be broken.



For those of you who actually read the exercise to the end, kreogCom
are chained with each other :p

The following code must compile and must display the following output:


```
1 int main()
2 {
3     KreogCom k(42, 42, 101010);
4     k.addCom(56, 25, 65);
5     k.addCom(73, 34, 51);
6
7     k.locateSquad();
8
9     k.removeCom();
10    k.removeCom();
11
12    return 0;
13 }
```

Output :

```
1 Kinder:ex02$ g++ -W -Wall -Werror *.cpp
2 Kinder:ex02$ ./a.out | cat -e
3 KreogCom 101010 initialised$
4 KreogCom 65 initialised$
5 KreogCom 51 initialised$
6 KreogCom 51 currently at 73 34$
7 KreogCom 65 currently at 56 25$
8 KreogCom 101010 currently at 42 42$
9 KreogCom 51 shutting down$
10 KreogCom 65 shutting down$
11 KreogCom 101010 shutting down$
12 Kinder:ex_2$
```

Chapter V

Exercise 3

	Exercise : 03	points : 5
Lock'n load baby		
Turn-in directory: (piscine_cpp_d07a)/ex03		
Compiler: g++	Compilation flags: -Wall -Wextra -Werror	
Makefile: No	Rules: n/a	
Files to turn in : Phaser.h, Phaser.cpp, Sounds.h		
Remarks : n/a		
Forbidden functions : malloc, free, *printf		

Now that you went through basic training, it's time to have some real fun: This is the Phaser Kreog'o Blaster mk2, your best friend for the rest of your life. From now on, you will train with it, eat with it and sleep with it. This sweetheart possesses three different modes:

- REGULAR , This is for clean jobs.
- PLASMA , This is to warm up the mood.
- ROCKET , This is for surgical strikes.

However, due to budget cut, the weapon is delivered in spare parts. Thus, if you want to blast some Dingos it's time to roll up your sleeves and to get to work.

In this matter you have to implement the following classes: (To complete)

```
class Phaser
{
    public:
        enum AmmoType
        { ... };

    public:
```

```

    Phaser(int maxAmmo, AmmoType type);
    ~Phaser();

public:
    void fire();
    void ejectClip();
    void changeType(AmmoType newType);
    void reload();
    void addAmmo(AmmoType type);
public:
    int getCurrentAmmos();

private:
    static const int Empty;

private:
    [...]
};

```

You also need to implement the class `Sounds` . This class possesses the following constant class variables:

- `std::string Regular` .
- `std::string Plasma` .
- `std::string Rocket` .



These variables must not be assigned to within the files you turn in
! We will do it in the main test of the correction.

Explanation:

- A `Phaser` has a maximum number of ammunitions, a number representing the current number of ammunition loaded, a sound for each type of firing mode, a magazine with ammunitions and a default type of munition for the weapon.
- A `Phaser` has a default magazine of 20 bullets maximum from the type `REGULAR`
- A `Phaser` is fully loaded when created.
- The variable `Empty` represents an empty magazine, it is strongly advised to use it in your program. You **MUST** initialize it with the value 0.

- When calling the member function `fire`, you must display the chain

```
1      Clip empty, please reload
```

followed by a newline character, if the magazine is empty. If the magazine is not empty, you display the sound of the ammunition loaded in the first case of the magazine. Once done, the size of the magazine is reduced by 1, which corresponds to the ammunition just fired.

- The member function `ejectClip` ejects the magazine in the weapon and reduces the number of ammunition to 0.
- The member function `changeType` displays on the standard output:

```
1      Switching ammo to type : [TYPE]
```

followed by a newline character, this `[TYPE]` is equal to: the value of the parameter passed in lower case (ex : `regular` for `REGULAR`). A call to this member function changes the default type of the Phaser.

- The member Function `reload` displays:

```
1      Reloading ...
```

on the standard output. Then it reloads the weapon with its default ammunition type.

- The member function `addAmmo` adds a munition with the type `type` at the end of the magazine. If the number of current ammunition is equal to the maximum number of munitions for the weapon, display the string:

```
1      Clip full
```

on the standard output, and do nothing. Otherwise add the ammunition to the magazine.

- The member function `getCurrentAmmos` returns the number of ammunition in the magazine of the weapon.



The member function `getCurrentAmmos` can be called on immutable Phaser objects.

The following code must compile:

```
1 int main()
2 {
3     Phaser p(5, Phaser::ROCKET);
4     p.fire();
5     p.reload();
6
7     std::cout << "Firing all ammunitions" << std::endl;
8     while (p.getCurrentAmmos() > 0)
9         p.fire();
10
11     return 0;
12 }
```

And produces the following output:


```
1 Kinder:ex03$ g++ -W -Wall -Werror *.cpp
2 Kinder:ex03$ ./a.out | cat -e
3 Boooooooooom$
4 Reloading ...$
5 Firing all ammunitions$
6 Boooooooooom$
7 Boooooooooom$
8 Boooooooooom$
9 Boooooooooom$
10 Boooooooooom$
```



In this case, the rocket sound is "Boooooooooom", it's up to you to find out how to initialize it.

Chapter VI

Exercise 4

	Exercise : 04	points : 5
G-Squad		
Turn-in directory: (piscine_cpp_d07a)/ex04		
Compiler: g++	Compilation flags: -Wall -Wextra -Werror	
Makefile: No	Rules: n/a	
Files to turn in : Skat.h, Skat.cpp, Phaser.h, Phaser.cpp, Sounds.h, KreogCom.h, KreogCom.cpp, Squad.h, Squad.cpp		
Remarks : n/a		
Forbidden functions : None		

Ok guys, I have nothing else to teach you. However, before blasting some Dingos, You have to get equipped, because we won't leave you on the ice-pad in your underwear.
Explanation:

- A Skat possesses now a `Phaser` and a `KreogCom` .
- You need to add a constructor with the following signature:

```
Skat(std::string const& m_name, int stimPaks,
     int serial, int x, int y, Phaser::AmmoType type);
```

Parameters `serial` , `x` , `y` and `type` correspond to the necessary informations to construct the `KreogCom` and the `Phaser`.

- Each Skat get a Phaser with 20 munitions.



This class doesn't possess a default constructor anymore.

o You need to add the following member functions:

- * `void fire()` so that your Skat opens fire.
- * `void locate()` that gives the position of your Skat.
- * `void reload()` so that your Skat reloads.
- * `KreogCom& com()` that gives the communicator of your Skat.



You have to figure out by yourself which one of these member functions are constant.

- You have to add the following member function to the class `KreogCom` : `void addCom(KreogCom* com)` , which role is to link the current `KreogCom` with the one passed as a parameter.



Correction won't call the member function `removeCom` . It is useless to change it.

Good, now that you look like something decent, it becomes necessary to get organized.
In this matter you have to implement the following class:

```

1 class Squad
2 {
3     public:
4         Squad(int posXBegin, int posYBegin, Phaser::AmmoType ammoType,
5             int size);
6         ~Squad();
7
8     public:
9         void fire();
10        void localisation();
11
12    public:
13        [...] skats();
14        [...] at(int idx);
15        int size();
16
```



```

17     private:
18         [...]
19 };

```

Sounds from the class `Sounds` must be initialized in the file `Squad.cpp`, with the following values:

- Regular : “Bang”
- Plasma : “Fwooosh”
- Rocket : “Boouuuuum” !!!

Explanation:

- A squad is made of `[size]` `Skats`. By default, a squad is made of 5 `Skat`.
- The last teammate of a squad must be 0.
- Teammate’s communicators serials are equivalent to the place of the `Skat` within the squad. (`Skat 0` -> `com serial 0`, ..., `Skat n` -> `com serial n`)
- Regarding their position, the first `Skat` is located at the position `posXBegin`, `posYBegin`.
- The position X is incremented by 10 for each `Skat` of the squad.
- The position Y is incremented of 15 for each `Skat` of the squad.

```

ex : (Skat 0 -> X = 0, Y = 0; Skat 1 -> X = 10, Y = 15; ...;
      Skat n -> X = (n) * 10, Y = (n) * 15)

```

- Each communicator of each skats are linked together, based on the following scheme:

```

( ---> = link to )
-----
| com Skat 0 | ---> | com Skat 1 | ---> | ... | ---> | com Skat n |
-----

```

- The member function `fire` starts opening fire for every squad members.
- The member function `localisation` displays the position of each squad members, from the `FIRST` member.
- The member function `skats` returns all the squad members.

- The member function `at` returns the `Skat` from the `idx` index. If this index is not valid, it returns 0.
- The member function `size` returns the size of the squad.

In order to interact more easily with your squad, you have to implement the following function:

`void foreach([...] beginIdx, [...] actionPtr)` In the files related to the squad. This function accepts for parameter an index representing the beginning of a group of `Skat`, and a pointer on a member function from the class `Skat`, that doesn't accept parameters and that returns `void`.



This function is called with all the member functions of the class `Skat` with no parameters and that returns `void`.



Is a member function different from a `const` member function with the exact same signature? `int class::fct() ==? int class::fct() const`

The following code must compile and must display the following output:

```
1 int main()
2 {
3     Squad s(0, 0, Phaser::REGULAR);
4
5     s.fire();
6
7     return 0;
8 }
```

And must generate the following output:

```
1 Kinder:ex04$ g++ -W -Wall -Werror *.cpp
2 Kinder:ex04$ ./a.out | cat -e
3 KreogCom 0 initialised$
4 KreogCom 1 initialised$
5 KreogCom 2 initialised$
6 KreogCom 3 initialised$
7 KreogCom 4 initialised$
8 Bang$
9 Bang$
10 Bang$
11 Bang$
12 Bang$
13 KreogCom 0 shutting down$
14 KreogCom 1 shutting down$
15 KreogCom 2 shutting down$
16 KreogCom 3 shutting down$
17 KreogCom 4 shutting down$
18 Kinder:ex04$
```

This is it. Break now private, and good luck for your mission.