Santa Claus

C++ Pool - Rush 2

Koalab koala@epitech.eu

Abstract: *This document is the subject of the Rush 2*

Contents

Chapter I

Santa's elves

Santa Claus has made an official request for Epitech to computerize his chain of gift wrapping. He is tired of these stupid elves and would like to automate that if possible. He asks you to design a chain packaging simulator.

Warning: This subject is deliberately opaque. You are therefore requested to reorganize things logically and to deduce the implied orders to meet all requirements. A FULL read of the subject is necessary to understand all interactions. No, koalas don't smoke eucalyptus before writing their subject!

Chapter II

Little Pony and Teddy

He provides you with a list describing the job of the elf on the gifts wrapping chain. Your project leader has choosen to take charge and has treated the functional analysis and some design aspects for you. He provides you a list of indications and constraints to implement different parts of the simulator.

Then he tells you:

- A `Teddy` is a `Toy`.
- It exists also a `Toy LittlePony`.
- `Teddy` screams "gra hu" when we take it (method `isTaken`).
- The `LittlePony` screams "yo man" when we take it (method `isTaken`).
- A `Toy` is an `Object`.

He asks you to write the classes `Toy`, `Teddy`, `LittlePony`...
He wants that you implement the function:

```
1  Object **MyUnitTests();
```

In this one, he wants you to test the construction of a `LittlePony` object with a title "gay pony", and a `Teddy` "bisounours". You must respectively return them into an array of 2 elements.

You can use iostreams to write on the standard output. You return to the ligne after each writing.

Chapter III

A box history

Now you have to be able to handle gift wrapping:

- A `Box` is a `Wrap`.
- A `GiftPaper` is also a `Wrap`.
- The elf wraps via the method `wrapMeThat` of `Wrap`.
- The elf must be able to put an `Object` (and only one) into a `Box` by wrapping it into this `Box` if and only if the `Box` is opened.
- A closed `Box` cannot wrap.
- A `GiftPaper` doesn't have to be opened to wrap.
- A `Wrap` containing something cannot wrap.
- The elf can open a `Wrap` to access to what has been wrapped (method `openMe`)
- Once the `Wrap` opened and the object taken by the elf, the `Wrap` is empty again.
- The elf can close an opened `Box` (method `closeMe`)
- When the elf takes a `Wrap`, he says "whistles while working"
- When the elf wraps, he says "tuuuut tuuut tuut"

class Box : public Wrapclass GIftPaper : public Wr

bool _isOpen;

What is Object ?

method Wrap in herited classes ?

_isOpen = true;

_isOpen = false;

He asks you to write the classes `Wrap`, `Box`, `GiftPaper`.
He wants you to implement the function:

```
1  Object *MyUnitTests(Object **);
```

He gives you an array of 3 elements (null element at the end), respectively containing 1 `Teddy`, 1 `Box`, and 1 `GiftPaper`. You must wrap everything and return a gift.

Errors cases must generate explicit messages on the error output. You are free to do more test programs on your side.

Chapter IV

A rolling carpet gathers no moss

You must now be able to manage the workstation:

- The elf has a `Table` in front of him and a `ConveyorBelt` beside him.
- We can't put anything on the `ConveyorBelt` if there is already something.
- The elf can Put and Take the `Object` on the `Table.`
- Ditto on the `ConveyorBelt.`
- When there is no more room on the `Table` , it colapses. It can contain a maximum of 10 Objects.
- The elf receives `Wrap` by pressing the button "IN" of the `ConveyorBelt` with a Hand, or sends to Santa Claus what currently is on the `ConveyorBelt` by pressing the button "OUT".
- A `Wrap` sent to Santa Claus is considered as vanished, the `ConveyorBelt` is free again.
- The elf can `Look` on the `Table` to know what is on it. He obtains an array of titles of the different objects. The last element of the array is `null.`
- The elf can `Look` on the conveyor belt to know what is on it.
- It is an error to push on a button of the conveyor belt if the input/output of the conveyor belt has not been initialized.

He asks you to write the interfaces `ITable` and `IConveyorBelt` and the classes `TablePePeNoel` and `ConveyorBeltPePeNoel` implement these interfaces.
The Santa's table and his conveyor belt contain enough to make 2 gifts. The organization/distribution of wraps/toys between the conveyor belt and the table is at your convenience.

He wants that you provide him two functions:

```
1 ITable *createTable();
2 IConveyorBelt *createConveyorBelt();
```

allowing to reify (instantiate) these 2 objects. The error cases must generate explicit messages on the error output.

Chapter V

Elf job (or why I will never make a Christmas list)

We must now manage the elf.

He asks you to write the `IElf` interface implementing all the elf actions as described in the whole subject.
Then, to implement an `ElfOfPePeNoel` implementing this interface.

When we ask the elf to wrap a gift, he does it on his own between the table and the conveyor belt to wrap up a gift and to send it by the conveyor belt by an `OUT`.

Chapter VI

Here's the box, bitch

Let's complicate things a bit. Implement a `TableRand` class and a `ConveyorBeltRand` class that present a random disposition of objects. Your elf must be able to make a maximum number of presents and send them to Santa.

When he can't do anything (not enough component to make a gift), he yells:

```
1  pepe ya un schmolle dans le bignou
```

Plus, `ConveyorBeltRand` has a strange behavior, and it makes lots of noise. The mechanic goblin says that it talks. "e speex'n'XML" as he says. With his goblin accent, we understand nothing at all.
However your conveyor belt must speex'n'XML. That is to say the conveyor belt describes the packet that you send starting by the wrap to the interior in XML. 'Serialization', that's how the goblin calls it. Do not panic, this is a goblin. When we push the green button Zwify, the `ConveyorBeltRand` spits an XML index card.

Chapter VII

Is Santa is quantum?*

*c'mon, deliver all days at the same time all around the world! What is this worthless ubiquity?

We must now implement a Santa Claus which is a separate program. Santa Claus is lazy and waits for gifts to put them into the sleigh. A gift is the result of a serialization of your objects in XML. These files are given to Santa as parameter on the command line.

```
1  >./santa gift1.xml gift2.xml gift3.xml
```

You must PROVE to the factory inspector that your Santa loads each gift and recreates them into memory by using what this grim goblin calls 'deserialization'. To do so, you must take each gift, open it, take the toy and put it back inside the box. And yeah, the quality control is important for Santa. Him, he is not stupid: to know if the cat is dead, he opens the box! And take the object inside will make yell the toy, which will be useful to know what is inside.

The following is more destinated to the tek3s. However the true warriors can rub up against.

Chapter VIII

The Warp Machine or a further degree before dementia.

Conceptual vision of the WarpMachine

Packets are coming from a machine called “The Warp Machine”. Any packet sent by the new conveyor belt with the OUT command is received by the warp machine, by one of these parts that looks like a big magic sock. Much like those that hang above the fireplace but really uglier. These are the dwarf Porte and PaNawak which gave us many years ago against the integral of Tommy verdâtre “song for corpus cavernosum penis”. However this sacred machine waits a mystical number to be connected to the elves’ magical conveyor belt. Number that we will have to give to Santa when we will launch it in warp mode with a ’-w’ parameter as «Warp Me Up Scotty».

The head of the two dwarfs explained to Santa how to write this number. Here is what have been said...*

To connect the magical sock to the warp,
this sock must be ‘figurated in ’u di pee multi-caste!?’.
The mystical number works by quadruplet.
The first one always starts between 224 and 239.
Ask my cousin gougle for more information.

*well, after an approximative translation of his dialect «the north shmolesque».

That's it. Modify your Santa Claus program so that it takes this mystical number as parameter and that it properly configures the warp machine connected with the magic sock. In addition, the mechanic goblin made a new version of `ConveyorBeltRand` : «the MagicalCarpet». As the `ConveyorBeltRand` e speex'n'XML and this noise is transmitted in the warp. This XML is captured by the sock and is played by a part of the warp machine that looks like a gramophone to Santa. Santa Claus therefore knows what happens on the conveyor belt. Santa Claus, thanks to this music, brings out the gift of the warp. And he proves to the inspector by vigorously grabbing the toy which is inside.

Thanks to that, several elf production lines can send their gifts to Santa Claus.

If several Santa use the same mystical number, they will receive all the gifts destined to this mystical number which is very convenient to duplicate the gifts. Several Santa Claus? But we said that he was quantum... be attentive!

Bonus: Try to impose the XML language to the other groups. In order to make possible the work between different elves of different groups with different Santa Claus.