



Piscine C++ - Exam 01

Looking For Kreog

Maxime "zaz" Montinet zaz@42.fr

Résumé: Ce document est le sujet de votre second exam de la piscine C++ de 42. Il est librement inspiré du webcomic "Looking For Group" par R.Sohmer et L.DeSouza. Si vous ne le connaissez pas, honte à vous, filez demander à Google après l'exam.

Table des matières

I	Détails administratifs	2
I.1	Consignes générales	2
I.2	Code	3
II	Foreword	5
III	Exercice 00 : This day is fantastic !	6
IV	Exercice 01 : You like what I do ?	9
V	Exercice 02 : Don't fwoosh me, bro !	12
VI	Exercice 03 : I. LIKE. TO. KILL. THINGS. How's that not clear by now ?	15

Chapitre I

Détails administratifs

I.1 Consignes générales

- Aucune communication n'est permise
- Ceci est un examen, vous n'avez pas le droit de discuter, écouter de la musique, faire du bruit ... Plus généralement, vous n'avez pas le droit de faire quoi que ce soit qui puisse perturber le travail des autres étudiants de quelque façon que ce soit.
- Vos téléphones et autres appareils technologiques doivent être éteints **et** rangés. Si un téléphone sonne, toute la rangée concernée sera disqualifiée de l'examen et immédiatement renvoyée.
- Votre répertoire home contient deux répertoires : "**rendu**" et "**sujet**".
- Le répertoire "**sujet**" contient le sujet de l'examen, mais visiblement vous le savez déjà.
- Le répertoire "**rendu**" est un clone de votre dépôt Git de rendu. Vous devez travailler dedans, et rendre comme avec n'importe quel autre dépôt.
- Seul ce qui est pushé sur votre dépôt Git sera noté. Le système va arrêter d'accepter les push à l'heure précise de fin de l'examen. N'attendez PAS la dernière minute pour pusher votre travail.
- Vous ne pouvez exécuter vos programmes que dans le répertoire "**rendu**" ou l'un de ses sous-répertoires.
- Chaque exercice doit être réalisé dans le répertoire approprié, précisé dans le header de chaque exercice.
- Vous devez rendre, à la racine de votre dépôt, un fichier nommé "**auteur**" dans lequel figure votre login suivi d'un saut de ligne. Si ce fichier est manquant, ou faux, vous ne serez PAS noté.

Exemple :

```
$> cat -e ~/rendu/auteur  
xlogin$
```

- Certaines exercices requièrent une lecture du man ...
- Vous serez noté par un programme. Vous devez donc respecter les noms de fichiers/-répertoires/fonctions/classes À LA LETTRE.
- Les exercices vont TOUJOURS préciser quels fichiers seront ramassés :
 - Quand un exercice demande des fichiers spécifiques, ils seront explicitement listés. Par exemple, "file1.cpp file1.hpp".
 - Autrement, quand les noms de fichiers sont à votre discrétion, l'exercice précisera quelque chose du genre "*.cpp *.hpp".
 - Quand un Makefile sera requis, ce sera toujours explicitement précisé.
- En cas de problème technique, question sur le sujet, ou tout autre problème, vous devez vous lever et attendre en silence qu'un membre du staff vienne vous voir. Vous ne devez en aucun cas demander à vos voisins ni appeler verbalement un membre du staff, ceci afin de préserver le silence dans la salle.
- Tout équipement qui n'est pas explicitement autorisé est implicitement interdit.
- Toute sortie est définitive.
- Les membres du staff peuvent vous renvoyer de la salle sans préavis s'ils l'estiment nécessaire.
- Vous avez le droit à des feuilles de papier vierges et un stylo. Pas de cahier, livres, ou aide d'aucune sorte.
- Tous les exercices sont obligatoires, la notation s'arrête au premier faux.
- Lisez chaque exercice EN ENTIER avant de commencer. Vraiment.
- Par Odin, par Thor ! Réfléchissez!!!
- Pour toute question après l'examen, faites un ticket.

I.2 Code

- La correction est entièrement automatique et effectuée par un programme appelé la **Moulinette**
- Dans cet examen, vous ne devez pas rendre de fonction **main**, car nous utiliserons la nôtre pour tester votre code, et si vous en avez une aussi, ça ne compilera pas et vous aurez 0 (Et ce sera bien fait)
- Toute fonction implémentée dans un header (Sauf templates) ou headers non protégé signifie 0 à l'exercice.
- Toute sortie va vers la sortie standard et est terminée par un saut de ligne, sauf contre-indication explicite.
- Rappelez vous que vous codez en **C++** maintenant et non plus en **C**, donc les fonctions

suivantes sont interdites et leur utilisation sera sanctionnée par un -42 : `*alloc`, `*printf` et `free`

- Notez également que sauf contre-indication explicite, les mots-clés `"using namespace"` et `"friend"` sont tout aussi interdits.
- Puisque vous n'avez le droit qu'aux notions de C++ vues en piscine, vous n'avez pas le droit à des bibliothèques externes, aux constructs de C++11, à Boost ou autres trucs que votre pote super doué en C++ vous a dit être `"trop génial"` ou `"indispensable"`.
- Chacun de vos includes doit pouvoir être inclus indépendamment des autres. Chaque include doit donc évidemment inclure les autres include dont il dépend.
- Quand une sortie d'exemple fait un saut de ligne, il doit être ignoré s'il est précédé d'un backslash (`"\"`). Pareil pour une string entre double quotes. Par exemple, la chaîne `"For \`
`pony!"` doit être comprise comme `"For pony!"`.

Chapitre II

Foreword

Richard est un Warlock. On est pas certains de ce que ça veut vraiment dire, mais ça sonne plutôt badass, non ?

Quoi qu'il en soit, on est sûrs de quelques trucs : Il aime tuer des choses, il passe une super journée quand il peut massacrer une famille entière devant leurs proches, il aime bien les poneys, et il est le maire d'un petit village sur la côte.


Du coup, on aimerait le comprendre un peu mieux, et comme vous êtes du genre serviable, vous allez nous coder un authentique Warlock domestique, qu'on puisse étudier jusqu'à plus soif. Et peut-être même qu'il se rendra utile en cramant quelques rats.

Au boulot !



Chapitre III

Exercice 00 : This day is fantastic !

	Exercice : 00
This day is fantastic !	
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : Warlock. [hpp,cpp]	
Fonctions interdites : Aucune	
Remarques : n/a	

Créez une classe **Warlock** en forme de Coplien.

Elle a les attributs privés suivants :

- **name** (string)

- **title** (string)

Vu qu'ils sont privés, vous aurez besoin des getters suivants :

- **getName**, retourne une référence vers string constante

- **getTitle**, retourne une référence vers string constante

Ces deux fonctions doivent pouvoir être appelées sur un **Warlock** constant.

Votre **Warlock** n'apprécierait pas qu'on change son nom, donc on va se contenter de pouvoir changer son titre de temps en temps. Après tout, c'est bien la variété. Écrivez donc le setter suivant :

- **setTitle**, retourne void et prend une référence sur string constante.

Votre **Warlock** va aussi avoir, en plus de tout ce qui est requis par la forme de Coplien, un constructeur prenant dans l'ordre son nom et son titre. Il ne peut ni être copié, ni instancié par copie, ni instancié sans nom et titre.

Par exemple :

```
Warlock bob; //Ne compile pas
Warlock bob("Bob", "the magnificent"); //Compile
Warlock jim("Jim", "the nauseating"); //Compile
```

```
bob = jim;                                //Ne compile pas
Warlock jack(jim);                        //Ne compile pas
```

À la création, le Warlock dit :

<NAME>: This looks like another boring day.

Bien évidemment, quand on utilise des placeholders comme <NAME>, <TITLE>, <QUOTE1>, etc... dans les sorties, vous les remplacerez par la valeur appropriée. Si vous ne le faites pas, on vous fera asseoir quelque part dans le Bocal afin qu'on puisse rire de vous pendant une heure ou deux. (Non, ce n'est pas une blague)

À sa mort, il dit :

<NAME>: Ahhh, I see it clearly. This is the plane of SUCK...

Il devra également pouvoir se présenter tout en se mettant bien en valeur. Normal, me direz vous, il a un énorme ... euh, ego.

Écrivez donc la fonction suivante :

- void introduce() const ;

Qui affichera :

<NAME>: I am <NAME>, <TITLE> !

Voici un exemple de fonction main avec sa sortie :

```
int main()
{
    Warlock const richard("Richard", "Mistress of Magma");
    richard.introduce();
    std::cout << richard.getName() << " - " << richard.getTitle() << std::endl;

    Warlock* jack = new Warlock("Jack", "the Long");
    jack->introduce();
    jack->setTitle("the Not-So-Long-After-All");
    jack->introduce();


    delete jack;

    return (0);
}
```

```
zaz@naqadah:~$ ./a.out | cat -e
Richard: This looks like another boring day.$
Richard: I am Richard, Mistress of Magma !$
Richard - Mistress of Magma$
Jack: This looks like another boring day.$
Jack: I am Jack, the Long !$
Jack: I am Jack, the Not-So-Long-After-All !$
Jack: Ahhh, I see it clearly. This is the plane of SUCK...$
Richard: Ahhh, I see it clearly. This is the plane of SUCK...$
zaz@naqadah:~$
```

Chapitre IV

Exercice 01 : You like what I do ?

	Exercice : 01
You like what I do ?	
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre : Warlock. [hpp,cpp]	
Fonctions interdites : Aucune	
Remarques : n/a	

On me souffle à l'oreille que notre Warlock n'est pas encore complet.

En effet, le vrai Richard a plusieurs titres, et non pas un seul. Il est également bien plus loquace que notre Warlock actuel !

Modifiez le Warlock de telle sorte que :

- ~~Le constructeur qui prenait un nom et un titre ne prend plus qu'un nom.~~
- ~~L'attribut `title` est maintenant une liste de strings et s'appelle `titles`.~~
- ~~Les fonctions `getTitle` et `setTitle` disparaissent.~~
- ~~La fonction `introduce` doit maintenant afficher tous les titres du Warlock, plus un autre, de la façon suivante :~~

<NAME>: I am <NAME>, <TITLE1>, <TITLE2> [...], <TITLEn> ! And the mayor of a little village on the coast. Very scenic during springtime, you should visit sometime !

Par exemple, si notre Warlock appelé Richard a les titres suivants :

- Lord of the Undead Ponies
 - Master of the Wine Cellar
- Vous devez afficher :

Richard: I am Richard, Lord of the Undead Ponies, Master of the Wine \ Cellar ! And the mayor of a little village on the coast. Very scenic during \ springtime, you should visit sometime !

Si notre **Warlock** n'a pas de titre, contentez-vous de mentionner qu'il a un petit village sur la côte (Oui, c'est important) :

```
Richard: I am Richard ! And the mayor of a little village on the coast. \
Very scenic during springtime, you should visit sometime !
```

Vu qu'il va probablement falloir manipuler ses titres à un moment ou un autre, et qu'on ne va pas lui demander de le faire tout seul (Il a mieux à faire, quand même), vous allez faire des fonctions pour gérer cela :

- **addTitle**, qui prend une référence sur string constante et ajoute un nouveau titre à la collection du **Warlock**.
- **removeTitle**, qui prend une référence sur string constante et efface le titre en question s'il existe, sinon ne fait rien.

Mieux. Maintenant que notre **Warlock** est capable de dérouler sa longue ... liste de titres, on va devoir faire en sorte qu'il puisse parler. Parce qu'un **Warlock** qui ne peut pas lâcher deux ou trois blagues glauques et bien choisies devant le malheur des autres est à peu près aussi utile qu'un développeur web.

Ajoutez donc un attribut privé nommé **quotes**, qui sera une liste (ou un vector, selon ce que vous pensez être mieux) de strings. Un choix est évidemment meilleur que l'autre.

Comme pour les titres, faites les fonctions **addQuote** et **removeQuote**.

Quand ce sera fait, ajoutez une fonction **talk**, qui ne prend rien, ne retourne rien et est callable sur un **Warlock** constant.

Cette fonction va choisir une quote au hasard parmi celles que notre **Warlock** connaît, et affiche :

```
<NAME>: <QUOTE>
```

Dans le cas (que l'on espère très rare) où notre **Warlock** n'a rien à dire, cette fonction ne fait rien.



Pour choisir une quote au hasard, utilisez `rand() % nombre_de_quotes_connues`. N'utilisez PAS `srand()` dans votre code. Jamais. Sinon la Moulinette n'aura pas la même random seed que vous, et vous aurez 0. Ca serait con.

Voici un main d'exemple avec sa sortie associée. Bien évidemment, l'ordre des quotes peut différer, vu que, ben, random ...

```
int main()
{
    Warlock richard("Richard");

    richard.addTitle("Chief Warlock of the Brothers of Darkness");
    richard.addTitle("Lord of the Thirteen Hells");
    richard.addTitle("Emperor of the Black");
    richard.addTitle("Lord of the Undead");
    richard.addTitle("Mistress of Magma");
    richard.introduce();

    richard.removeTitle("Mistress of Magma");
    richard.introduce();

    richard.addQuote("You've just been Dick-rolled !");
    richard.addQuote("This day is fantastic ...");
    richard.addQuote("That orphanage attacked ME. It was self-defense !");
    richard.addQuote("You like what I do ?");


    richard.talk();
    richard.talk();
}
```

```
zaz@naqadah:~$ ./a.out | cat -e
Richard: This looks like another boring day.$
Richard: I am Richard, Chief Warlock of the Brothers of Darkness, Lord of the
Thirteen Hells, Emperor of the Black, Lord of the Undead, Mistress of Magma !
And the mayor of a little village on the coast. Very scenic during springtime,
you should visit sometime !$
Richard: I am Richard, Chief Warlock of the Brothers of Darkness, Lord of the
Thirteen Hells, Emperor of the Black, Lord of the Undead ! And the mayor of a
little village on the coast. Very scenic during springtime, you should visit
sometime !$
Richard: This day is fantastic ...$
Richard: You've just been Dick-rolled !$
Richard: Ahhh, I see it clearly. This is the plane of SUCK...$
zaz@naqadah:~$
```



Chapitre V

Exercice 02 : Don't fwoosh me, bro !

	Exercice : 02
Don't fwoosh me, bro !	
Dossier de rendu : <i>ex02/</i>	
Fichiers à rendre : <code>Warlock.[hpp,cpp]</code> <code>ASpell.[hpp,cpp]</code> <code>ATarget.[hpp,cpp]</code> <code>Fwoosh.[hpp,cpp]</code> <code>LittleKid.[hpp,cpp]</code>	
Fonctions interdites : Aucune	
Remarques : n/a	



Dans la classe `Warlock`, le `switch` est strictement INTERDIT et son utilisation résulterait en un -42.

Maintenant on peut commencer à se poiler. Enfin, c'est ce que dirait notre `Warlock` s'il savait ce qu'on a en réserve pour lui.

Maintenant qu'il peut parler, et se la péter tranquillement, on va devoir lui trouver des trucs à faire. Si il ne peut pas tuer des gens, il s'ennuie. Pauvre de lui.

Créez une classe abstraite `ASpell`, en form de Coplien, qui a les attributs protégés suivants :

- `name` (string)
- `effects` (string)

Les deux auront des getters (`getName` et `getEffects`) qui retournent des strings.

Ajoutez une méthode pure `clone()` qui retourne un pointeur sur `ASpell`.

Toutes ces fonctions peuvent être appelées sur un objet constant.

ASpell a un constructeur qui prend son nom et ses effets, dans cet ordre.

Maintenant vous allez créer une classe abstraite **ATarget**, en forme de Coplien, qui a un attribut **type** (string), et son gette associé **getType**, qui retourne une référence sur string constante.

Pareil que pour **ASpell**, faites lui une méthode pure **clone()**.

Ces fonctions peuvent être appelées sur un objet constant.

Elle a un constructeur qui prend son type.

Ensuite, ajoutez à **ATarget** une fonction **getHitBySpell** qui prend une référence vers **ASpell** constant.

Elle affichera :

<TYPE> has been <EFFECTS> !

<TYPE> est le type du **ATarget**, et **<EFFECTS>** est le retour de la fonction **getEffects** du **ASpell**.

Enfin, ajoutez à votre **ASpell** une fonction **launch** qui prend une référence vers **ATarget** constant. Celle-ci va simplement appeler le **getHitBySpell** de l'objet passé en paramètre, en lui donnant en paramètre l'instance courante.

Ceci étant fait, implémentez votre classe **ASpell** avec une classe **Fwoosh**. Son constructeur par défaut va mettre son nom à "Fwoosh" et ses effets à "fwooshed". Bien évidemment, vous devez implémenter la méthode **clone()**, qui dans ce cas va retourner un pointeur vers un nouveau **Fwoosh**.

Dans le même style, créez un **ATarget** concret appelé **LittleKid**, dont le type sera "Little Kid". Évidemment vous devez implémenter sa méthode **clone()**.

Maintenant qu'on a des sorts et des cibles, il va falloir rendre le **Warlock** capable de les utiliser, sinon il serait toujours aussi inutile qu'un joueur console.

Ajoutez les fonctions membres suivantes au **Warlock** :

- **learnSpell**, prend un pointeur sur **ASpell**, qui apprend un sort au **Warlock**
- **forgetSpell**, prend une string correspondant au nom d'un sort, et le fait oublier au **Warlock** s'il le connaît, sinon ne fait rien.
- **launchSpell**, prend une string correspondant au nom d'un sort et une référence sur **ATarget**, qui lance le sort en question sur la cible s'il est connu, sinon ne fait rien.

Il vous faudra un nouvel attribut pour stocker les sorts de votre **Warlock**. Plusieurs types font l'affaire, à vous de choisir le plus adapté.

Ci-dessous un possible main et sa sortie :

```
int main()
{
    Warlock richard("Richard");

    richard.addTitle("Chief Warlock of the Brothers of Darkness");
    richard.addTitle("Lord of the Thirteen Hells");
    richard.addTitle("Emperor of the Black");
    richard.addTitle("Lord of the Undead");
    richard.addTitle("Mistress of Magma");
    richard.addQuote("You've just been Dick-rolled !");
    richard.addQuote("This day is fantastic ...");
    richard.addQuote("That orphanage attacked ME. It was self-defense !");
    richard.addQuote("You like what I do ?");

    LittleKid bob;
    Fwoosh* fwoosh = new Fwoosh();

    richard.learnSpell(fwoosh);

    richard.introduce();
    richard.launchSpell("Fwoosh", bob);
    richard.talk();


    richard.forgetSpell("Fwoosh");
    richard.launchSpell("Fwoosh", bob);
}
```

```
zaz@naqadah:~$ ./a.out | cat -e
Richard: This looks like another boring day.$
Richard: I am Richard, Chief Warlock of the Brothers of Darkness, Lord of the
Thirteen Hells, Emperor of the Black, Lord of the Undead, Mistress of Magma !
And the mayor of a little village on the coast. Very scenic during springtime
, you should visit sometime !$
Little Kid has been fwooshed !$
Richard: That orphanage attacked ME. It was self-defense !$
Richard: Ahhh, I see it clearly. This is the plane of SUCK...$
```



Chapitre VI

Exercice 03 : I. LIKE. TO. KILL. THINGS. How's that not clear by now ?

	Exercice : 03
I. LIKE. TO. KILL. THINGS. How's that not clear by now ?	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre : Warlock.[hpp,cpp] ASpell.[hpp,cpp] ATarget.[hpp,cpp] Fwoosh.[hpp,cpp] LittleKid.[hpp,cpp] Sprotch.[hpp,cpp] Ponymorph.[hpp,cpp] InnocentWoman.[hpp,cpp] SpellBook.[hpp,cpp] TargetGenerator.[hpp,cpp]	
Fonctions interdites : Aucune	
Remarques : n/a	



Dans les classes Warlock, SpellBook et TargetGenerator , le switch est tout à fait INTERDIT et son utilisation donnerait un -42.

Notre Warlock, après un après-midi délicieux passé à brûler vifs de petits enfants sans défense (Selon lui, ça "sentait le chocolat". Creeeeppyyyyy ...), s'ennuie déjà. Ben, du coup, normal, il n'a qu'un seul sort. Ca serait un peu comme de devoir coder en C# pour tout le reste de votre vie.

Aidez-le à enfin être heureux, et créez les deux sorts suivants sur le même modèle que Fwoosh :

- Sprotch (Nom : "Sprotch", Effets : "sprotched")
- Ponymorph (Nom : "Ponymorph", Effets : "turned into a nice, cute little pony")

En plus de ça, histoire de pas avoir que des enfants à massacrer, faites lui une autre cible, qui sera InnocentWoman (Type : "Innocent Young Woman").

Bon, du coup, ça devrait le faire.

Ou pas. On vient d'apprendre que le vrai Richard savait combiner ses sorts, pour des morts toujours plus inventives ... ("Eh, les mecs, les mecs! J'ai fait un poney zombie!!")

Du coup, il va falloir que vous le fassiez aussi.

Modifiez votre `ASpell` de la façon suivante :

- Ajoutez un attribut protégé `combined_with`, de type pointeur sur `ASpell`, initialisé à 0. On considère que le sort est combiné si cet attribut n'est pas 0.
- Ajoutez une fonction `combine`, qui retourne void et prend un pointeur sur `ASpell`, qui marchera comme un setter pour l'attribut `combined_with`.
- Ajoutez une fonction `isCombined`, qui retourne True si le sort est combiné.
- Modifiez `getEffects` pour retourner, en plus de son propre effet, l'effet du sort avec lequel il est combiné (si applicable). La valeur de retour doit avoir cette tête là (Si ce n'est pas clair, regardez l'exemple) :
 - Si le sort est combiné :
 - Si le sort "intérieur" est combiné aussi : "<EFFECT1>, <EFFECT2>"
 - Sinon : "<EFFECT1> and <EFFECT2>"
 - Sinon : "<EFFECT1>"
- Accessoirement, modifiez le `getName` de la même façon, pour retourner le nom du sort courant plus celui du sort "intérieur", si applicable. Ce sera "<NAME1>-<NAME2>" si appelé sur un sort combiné, juste "<NAME1>" sinon.
- Par exemple, pour un `Fwoosh` qui "contient" un `Ponymorph`, l'effet sera "fwooshed and turned into a nice, cute little pony", et le nom sera "Fwoosh-Ponymorph".
- Ajoutez une fonction `cloneCombine`, même signature que `clone`, qui réalisera une copie profonde du sort courant, en incluant tous les sorts combinés. Vous n'avez PAS besoin de gérer le cas d'une boucle de combinaisons.

Vu que combiner des sorts, c'est chiant, notre `Warlock` va devoir avoir un livre de sorts où il pourra stocker ses combinaisons favorites.

Faites une classe `SpellBook`, en forme canonique, qui ne peut être copiée ni instanciée par copie. Elle aura les fonctions suivantes :

- `void learnSpell(ASpell*)`, qui COPIE un sort dans le livre
- `void forgetSpell(string const &)`, qui efface un sort du livre s'il s'y trouve
- `ASpell* createSpell(string const &)`, qui crée le sort correspondant et le retourne.

Modifiez maintenant le `Warlock`, pour qu'il aie un `SpellBook` avec le même cycle de vie. Faites également en sorte que ses fonctions `learnSpell` et `forgetSpell` appellent celles de son livre. (Note de l'auteur : à ce moment précis, ma pizza vient d'arriver. Et

vous, vous avez la dalle?)

La fonction `launchSpell` devra utiliser le `SpellBook` pour créer le sort qu'elle essaie de lancer.

Enfin, pour qu'on puisse trouver des gens qui ne manqueront à personne sans nous balader dans les locaux de `Microsoft`, vous allez créer un générateur de cibles.

Faites donc une classe `TargetGenerator` en forme canonique, non copiable (Comme le `SpellBook`).

Il aura les fonctions suivantes :

- `void learnTargetType(ATarget*)`, apprend une cible au générateur
- `void forgetTargetType(string const &)`, fait oublier une cible au générateur s'il la connaît
- `ATarget* createTarget(string const &)`, qui crée une cible du type spécifié.

Bon, ben ça, c'est fait. Voici un main de test. Comme là je suis en train de bouffer, il n'est pas très rigoureux, et ne teste qu'une partie des fonctionnalités demandées. Mais vous êtes grands, là, vous allez y arriver, non ?

```

int main()
{
    Warlock richard("Richard");

    richard.addTitle("Chief Warlock of the Brothers of Darkness");
    richard.addTitle("Lord of the Thirteen Hells");
    richard.addTitle("Emperor of the Black");
    richard.addTitle("Lord of the Undead");
    richard.addTitle("Mistress of Magma");
    richard.addQuote("You've just been Dick-rolled !");
    richard.addQuote("This day is fantastic ...");
    richard.addQuote("Please ignore the fact that my hand is on fire. It's not\
        meant to be an aggressive gesture, it's how I say hello !");
    richard.addQuote("You like what I do ?");

    Ponymorph* ponymorph = new Ponymorph();

    richard.learnSpell(ponymorph);

    Sprotch* sprotch_ponymorph = new Sprotch();
    sprotch_ponymorph->combine(ponymorph);

    richard.learnSpell(sprotch_ponymorph);

    InnocentWoman stephanieDeMonaco;

    richard.introduce();
    richard.launchSpell("Ponymorph", stephanieDeMonaco);
    richard.launchSpell("Sprotch-Ponymorph", stephanieDeMonaco);
    richard.talk();
}

```

```

zaz@naqadah:~$ ./a.out | cat -e
Richard: This looks like another boring day.$
Richard: I am Richard, Chief Warlock of the Brothers of Darkness, Lord of the
Thirteen Hells, Emperor of the Black, Lord of the Undead, Mistress of Magma !
And the mayor of a little village on the coast. Very scenic during springtime
, you should visit sometime !$
Innocent Young Woman has been turned into a nice, cute little pony !$
Innocent Young Woman has been sprotched and turned into a nice, cute little
pony !$
Richard: Please ignore the fact that my hand is on fire. It's not meant to be
an aggressive gesture, it's how I say hello !$
Richard: Ahhh, I see it clearly. This is the plane of SUCK...$

```

