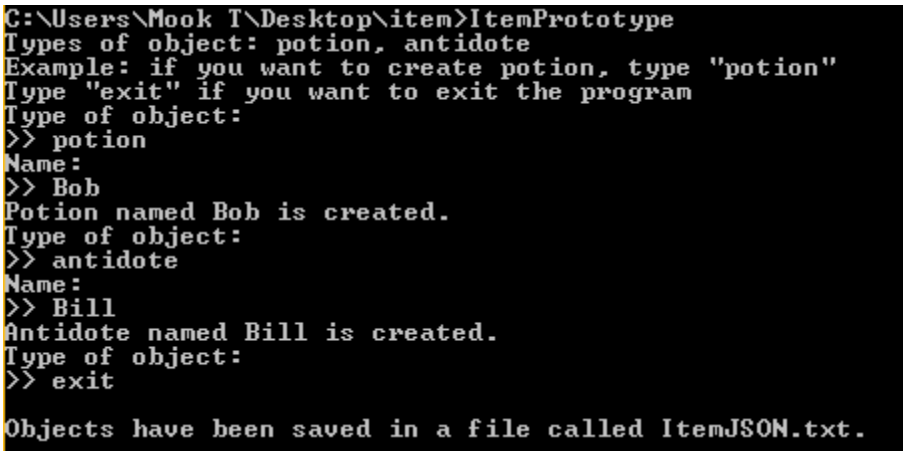
SENG 330 ASSINGMENT #2

Github: https://github.com/mookTungs/SENG330ASS2

The purpose of this program is to create items, which would be useful for the player base on my group project, using prototype design pattern.

The project contains a base class called Item and two derived classes called Potion and Antidote. The ItemFactory class stores both Potion prototype and Antidote and is also responsible for calling the clone() of the two objects.

The program will create object at run-time by prompt the user for the types of object, which are potion and antidote, and the object name.



You can exit the program by typing “exit”. The program will automatically save the created objects in a file called ItemJSON.txt.

```
{
        "items" :
        [
                {
                        "name" : "Bob",
                        "type" : "Potion"
                },
                {
                        "name" : "Bill",
                        "type" : "Antidote"
                }
        ]
}
```

The next time you loaded the program it’ll read the ItemJSON.txt and recreate all the objects that have been created last time you’ve used the program.

There is an executable program called ItemPrototype.exe which will run the program without having to compile anything.

How to build:

1) Download all the source code from the src folder from https://github.com/mookTungs/SENG330ASS2
2) Compile all the .cpp files
3) Run the executable

Source Code:

ItemPrototype.h

```
#ifndef _ItemPrototype_H_
#define _ItemPrototype_H_

#include <string>
#include "json.h"
#include <iostream>

using namespace std;

//!Abstract Item Class
class Item
{
public:
	//!Pure virtual clone method
	virtual Item* clone(string name) = 0;
	//!Print out statement confirming object has been created.
	void printItem();
	//!Return the type of the object.
	string getType();
	//!Return the name of the object.
	string getName();
	//!creating a JSON object
	Json::Value toJson() const;
protected:
	//!Store the type of the object.
	string itemType;
	//!Store the name of the object.
	string itemName;
};

//!A Potion class inherited from Item class.
/*!
```

```
  Potion class inherited from Item class. It contains a constructor and a clone function.
  It is used to increase the player's health.
*/
class Potion : public Item
{
public:
	//!Constructs an object Potion.
	Potion(string name);
	//!Return a new object Potion.
	Item* clone(string name);
};

//!An Antidote class.
/*!
  Antidote class inherited from Item class. It contains a constructor and a clone function.
  It is used to cure the player from poison.
*/
class Antidote : public Item
{
public:
	//!Constructs an object Antidote.
	Antidote(string name);
	//!Return a new object Antidote.
	Item* clone(string name);
};

//!An ItemFactory class
/*!
  ItemFactory store the prototypes and calls the clone() on that object,
  and return the object.
*/
class ItemFactory
{
public:
	//!A pointer to the Potion prototype.
	static Item* potion;
	//!A pointer to the Antidote prototype.
	static Item* antidote;
	//!Initialized the prototypes.
	static void initialize();
	//!Call the clone() of Potion and return the object.
	static Item* makePotion(string name);
	//!Call the clone() of Antidote and return the object.
```

```
        static Item* makeAntidote(string name);
};

#endif
```

ItemPrototype.cpp

```
#include "ItemPrototype.h"

using namespace std;

void Item::printItem()
{
        cout<< getType() << " named " << getName() << " is created." << endl;
}

string Item::getType()
{
        return itemType;
}

string Item::getName()
{
        return itemName;
}

Json::Value Item::toJson() const
{
        Json::Value object(Json::objectValue);
        object["name"] = itemName;
        object["type"] = itemType;
        return object;
}

Potion::Potion(string name)
{
        itemName = name;
        itemType = "Potion";
}

Item* Potion::clone(string name)
{
        return new Potion(name);
```

```
}

Antidote::Antidote(string name)
{
	itemName = name;
	itemType = "Antidote";
}

Item* Antidote::clone(string name)
{
	return new Antidote(name);
}

void ItemFactory::initialize()
{
	potion = new Potion("Prototype Potion");
	antidote = new Antidote("Prototype Antidote");
}

Item* ItemFactory::makePotion(string name)
{
	return potion->clone(name);
}

Item* ItemFactory::makeAntidote(string name)
{
	return antidote->clone(name);
}

Item* ItemFactory::potion = 0;
Item* ItemFactory::antidote = 0;
```

main.cpp

```
#include "ItemPrototype.h"
#include "JSONParsing.h"

int main()
{
	ItemFactory::initialize();
	Item* object;
	string objectType;
	string objectName;
```

```
	vector<Item*> objectList;
	JsonLoad(objectList);
	cout << "Types of object: potion, antidote" << endl;
	cout << "Example: if you want to create potion, type \"potion\"" << endl;
	cout << "Type \"exit\" if you want to exit the program" << endl;
	while(true)
	{
		cout << "Type of object:" << endl;
		cout << ">> ";
		cin >> objectType;
		if(objectType.compare("potion") == 0)
		{
			cout << "Name:" << endl;
			cout << ">> ";
			cin >> objectName;
			object = ItemFactory::makePotion(objectName);
			objectList.push_back(object);
			object->printItem();
		}
		else if(objectType.compare("antidote") == 0)
		{
			cout << "Name: " << endl;
			cout << ">> ";
			cin >> objectName;
			object = ItemFactory::makeAntidote(objectName);
			objectList.push_back(object);
			object->printItem();
		}
		else if(objectType.compare("exit") == 0)
		{
			break;
		}
		else
		{
			cout << "Invalid object. Please enter either \"potion\" or \"antidote\"" <<
endl;
			cout << "Type \"exit\" to exit the program." << endl;
		}
	}
	if(!objectList.empty())
	{
		JsonSave(objectList);
		cout << endl;
```

```
            cout << "Objects have been saved in a file called ItemJSON.txt." << endl;
        }
        while(!objectList.empty())
        {
            object = objectList.back();
            objectList.pop_back();
            delete object;
        }
        delete ItemFactory::potion;
        delete ItemFactory::antidote;
        return 0;
}
```

The source code for JSONParsing.h and JSONParsing.cpp can be found in github under src folder.

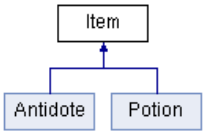
Example of doxygen document.

Item Class Reference abstract

Abstract **Item** Class. More...

`#include <ItemPrototype.h>`

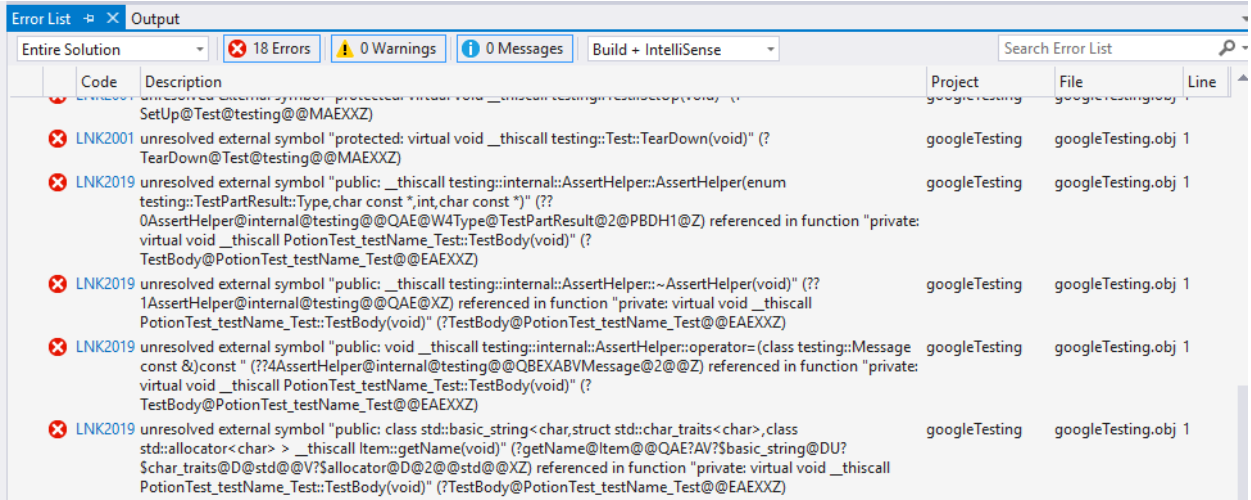
Inheritance diagram for Item:



Public Member Functions

virtual **Item** *	**clone** (string name)=0 Pure virtual clone method.
void	**printItem** () Print out statement confirming object has been created.
string	**getType** () Return the type of the object.
string	**getName** () Return the name of the object.
Json::Value	**toJson** () const creating a JSON object

This project did not include google testing framework due to the difficulty of installing and trying to get it to work on windows. Some examples of the error for the testing include.



JSON was also used in place of google protocol buffer because it wouldn't install on my laptop.