```
/*
 *  CASP_Return.h
 *  Defines a generic return object for the program
 *
 *  Created: 4/4/2017 by Ryan Tedeschi
 */

#ifndef CASP_RETURN_H
#define CASP_RETURN_H

#include <iostream>
#include <algorithm>
#include <string>
#include <unordered_map>
#include <vector>
#include "../Printable/Printable.h"

using namespace std;

class GenericData : public Printable {
    public:
        string GetType() {
            return type;
        };
        virtual void Print();

    private:
        string type = "";
};

template<typename T>
class GenericLeaf : public GenericData {
    public:
        GenericLeaf(T data) {
            this->data = data;
        };

        virtual void Print() {
            try {
                cout << specialLookups.at(data);
            } catch (...) {
                cout << data;
            }
        };

        void Assign(T data) {
            this->data = data;
        };

        void AddSpecial(T input, string output) {
            specialLookups[input] = output;
        };

    protected:
        string type = "Leaf";
        T data;
        unordered_map<T, string> specialLookups;
};

template<typename T>
static inline
GenericLeaf<T>* CreateLeaf(T data) {
    GenericLeaf<T>* leaf = new GenericLeaf<T>(data);
    return leaf;
};
template<>
```

```
static inline
GenericLeaf<bool>* CreateLeaf(bool data) {
    GenericLeaf<bool>* leaf = new GenericLeaf<bool>(data);
    leaf->AddSpecial(true, "true");
    leaf->AddSpecial(false, "false");
    return leaf;
};
template<>
static inline
GenericLeaf<string>* CreateLeaf<string>(string data) {

    int index = -1;
    while ((index = data.find("\"", index + 1)) != -1) {
        data = data.substr(0, index) + "\\" + data.substr(index, data.size());

        index++;
    }

    GenericLeaf<string>* leaf = new GenericLeaf<string>("\"" + data + "\"");
    return leaf;
};

class GenericObject : public GenericData {
    public:
        GenericObject();
        GenericObject(unordered_map<string, GenericData*>);
        virtual void Print();
        void Add(string, GenericData*);
        GenericData* At(string);

    protected:
        string type = "Object";
        unordered_map<string, GenericData*> data;

};
GenericObject* CreateObject();
GenericObject* CreateObject(unordered_map<string, GenericData*>);

class GenericArray : public GenericData {
    public:
        GenericArray();
        GenericArray(vector<GenericData*>);
        virtual void Print();
        void Add(GenericData*);
        GenericData* At(int);

    protected:
        string type = "Array";
        vector<GenericData*> data;
};

GenericArray* CreateArray();
GenericArray* CreateArray(vector<GenericData*>);

class CASP_Return : public GenericObject {
    public:
        CASP_Return();

        GenericArray* Errors();
        GenericArray* Warnings();
        GenericObject* Data();

        void AddStandardWarning(string, int = -1);
        void AddStandardError(string, int = -1);

    private:
};
```

```
#endif
```