7COM1025
Programming for Software Engineers
Lecture 15

Dr. Renato C. de Amorim

INHERITANCE VS COMPOSITION

Inheritance

'Is a' relationship

- But not always, a square is a rectangle but it would be bad design for a square to inherit from rectangle.
- This is because the functions in rectangle should work even if the object is actually a square.
- In a rectangle, changing the width has no impact on height, while this is not true in a square.

Functions may be inherited

- You may not need to re-implement them.

You can run into problems if the destructor of the base class is not virtual.

Composition

'Has a' relationship

You can make sure that any input is treated before you feed it to an object of a different class.

You will need to re-write every single setter and getter, even if no input treatment is necessary.

OVERLOADING <<

Neither inserter(<<) or extractor (>>) can be member of a class

```
#include <iostream>
#include <string>
using namespace std;
class ThreeD{
public:
    int x,y,z;
    ThreeD(int a, int b, int c){x=a;y=b;z=c;}
    };
    ostream &operator<<(ostream &stream, ThreeD obj){
        stream <<obj.x<<", "<<obj.y<<", "<<obj.z<<endl;
        return stream;
    }
int main(){
    ThreeD point(10,15,30);
    cout<<point;
    return 0;
}
```

```
#include <iostream>
#include <string>
using namespace std;
class ThreeD{
    int x,y,z;
public:
    ThreeD(int a, int b, int c){x=a;y=b;z=c;}
    friend ostream &operator<<(ostream &stream, ThreeD obj);
    };
    ostream &operator<<(ostream &stream, ThreeD obj){
        stream <<obj.x<<", "<<obj.y<<", "<<obj.z<<endl;
        return stream;
    }
int main(){
    ThreeD point(10,15,30);
    cout<<point;
    return 0;
}
```

PROBLEM 15.1

Write a class that holds a vector containing 5 integers (composition).

Make sure you write an API that allows the user (another programmer) to set the values of these 5 integers.

Overload the << operator so that the user can show on the screen the 5 integers using a single cout.

OPENING FILES

You open a file by linking it to a stream.

There are three types: input (ifstream), output (ofstream), input/output (fstream).

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ofstream out("test");
    if (!out){
        cout<<"Cannot open file"<<endl;
        return 1;
    }
    out<<"String 1"<<endl;
    out<<"string 2"<<endl;
    out<<"string 3"<<endl;
    out.close();
    return 0;
}
```

READING FROM FILES

```
#include <iostream>
#include <fstream>
#include<string>
using namespace std;
int main()
{
    string str;
    ifstream in("test");
    if (!in){
        cout<<"cannot open file for input."<<endl;
        return 1;
    }
    while(getline(in,str))
    {
        cout<<str<<endl;
    }
    in.close();
    return 0;
}
```

BLOCKS OF DATA IN BINARY

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    int n[5]={1,2,3,4,5};
    register int i;
    ofstream out("test2", ios::out |
ios::binary);
    if(!out){
        cout<<"Cannot open file."<<endl;
        return 1;
    }
    out.write((char *) &n, sizeof(n));
    out.close();
return 0;
}
```

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    int n[5];
    register int i;
    for(i=0;i<5;i++)
        n[i]=0;
    ifstream in ("test2", ios::in|ios::binary);
    if(!in){
        cout<<"Cannot open file."<<endl;
        return 1;
    }
    in.read((char *) &n, sizeof(n));
    for (i=0;i<5;i++)
        cout<<n[i]<<endl;
    in.close();
    return 0;
}
```

PROBLEM 15.2

Write a class whose objective is to compare two files. The user (another programmer) should receive a true if the files are equal and a false otherwise.

Extra exercise:
Using a single function make sure the user knows if the files are equal (true/false) AND how many lines are different.

EXCEPTION HANDLING

```
#include <iostream>
using namespace std;
int main(){
  cout<<"start"<<endl;
  try{
    cout<<"Inside try block"<<endl;
    throw 99;
    cout<<"this will not execute."<<endl;
  }
  catch (int i){
    cout<<"Catch an exception, the value is "<<i<<endl;
  }
  cout<<"end."<<endl;
  return 0;
}
```

The arguments must match

THROWING AN EXCEPTION FROM A FUNCTION

```
#include <iostream>
using namespace std;
void Xtest(int test){
    cout<<"Inside Xtest, test is "<<test<<endl;
    if (test)
        throw test;
}
int main(){
    cout<<"Start"<<endl;
    try{
        cout<<"Inside try block"<<endl;
        Xtest(0);
        Xtest(1);
        Xtest(2);
    }
    catch (int i){
        cout<<"Catch an exception, value: "<<i<<endl;
    }
    cout<<"end"<<endl;
    return 0;
}
```

MULTIPLE CATCHS

```
#include <iostream>
using namespace std;
void Xtest(int test){
    cout<<"Inside Xtest, test is "<<test<<endl;
    if (test)
        throw test;
    else
        throw "value is zero";
}
```

```
int main(){
    cout<<"Start"<<endl;
    try{
        cout<<"Inside try block"<<endl;
        Xtest(0);
        Xtest(1);
        Xtest(2);
    }
    catch (int i){
        cout<<"Catch an exception, value: "<<i<<endl;
    }
    catch (const char *str){
        cout<<"Catch an exception, string: "<<str<<endl;
    }
    cout<<"end"<<endl;
    return 0;
}
```

PROBLEM 15.3

Update the previous exercise so that the program throws an exception in case something is wrong (eg. The files cannot be opened or if no file name was passed to the function)