Composition

Outline

- What is it
- Example
- Where to use

Composition - What is it

Think of composition as a ‘has a’ relationship.

A car ‘has an’ engine
A person ‘has a’ watch

Composition aggregates objects and calls on them when needed

Composition - Example

- Create a human object that takes a watch pointer
- Add a method that ask the human for the time;
  - if they have a watch they give the time
  - If not they say “I dont have a watch”

See ‘Composition Introduction’ project on course website

Composition - Example

```
class Watch {
public:
    Watch();
    virtual std::string getTime();
};
```

getTime returns the time as a string

Composition - Example

```
class Watch {
public:
    Watch();
    virtual std::string getTime();
};
```

```
class Human {
public:

    Human(Watch *w);
    virtual ~Human();

    //how to get time from a human? You ask.
    //this is called delegation
    //if this human has a watch,
    //return the time, otherwise return
    //"I dont have a watch"
    std::string getTime();

private:
    //pointer, if have no watch its null
    Watch *w;
};
```

Note that human takes a watch pointer
If w=null they do not have a watch
If w points to a watch object they do have one

Composition - Example

```
class Watch {
public:
    Watch();
    virtual std::string getTime();
};
```

```
class Human {
public:

    Human(Watch *w);
    virtual ~Human();

    //how to get time from a human? You ask.
    //this is called delegation
    //if this human has a watch,
    //return the time, otherwise return
    //"I dont have a watch"
    std::string getTime();

private:
    //pointer, if have no watch its null
    Watch *w;
};
```

getTime returns the time as a string, just like watch does. In fact you can just return w→getTime() if w is not null

Composition - Example

Hold on to the pointer
Presumably the caller sets original pointer to null
This is easier to do with unique pointers BTW
Because you cannot have 2 unique pointers pointing to same object

3

```
#include "Human.h"

Human::Human(Watch *w):w(w) {
    //if passed a watch then this human has a watch but
    //could be passed a null pointer as well
    //notice that this is a shallow copy, akin to giving
    //a watch,
}

Human::~Human() {
    //some probs with raw pointers
    //if I own a watch, I am responsible for it
    if(w)
        delete w;
}

//how to get time from a human? You ask.
//this is called delegation
//if has a watch, returns time, otherwise says
//I dont have a watch
std::string Human::getTime(){
    if (w)
        return w->getTime();
    else
        return "I dont have a watch";
}
```

Composition - Example

When you ask a human the time they look at their watch

Same here, the human object 'delegates' the getTime task to a watch if they have it, otherwise replies "dont have one"
(typically you use the same function name as the object you are delegating to, getTime in this case)

4

```
#include "Human.h"

Human::Human(Watch *w):w(w) {
    //if passed a watch then this human has a watch but
    //could be passed a null pointer as well
    //notice that this is a shallow copy, akin to giving
    //a watch,
}

Human::~Human() {
    //some probs with raw pointers
    //if I own a watch, I am responsible for it
    if(w)
        delete w;
}

//how to get time from a human? You ask.
//this is called delegation
//if has a watch, returns time, otherwise says
//I dont have a watch
std::string Human::getTime(){
    if (w)
        return w->getTime();
    else
        return "I dont have a watch";
}
```

Composition - Example

```
int main() {
    Human h1(0);    //this human has no watch

    //what if I asked this human the time?
    //a human is not a watch, but they can own one
    cout<<"when asked the time h1 says: "<<h1.getTime()<<endl;

    //how about we create a watch and give it to the human
    Watch *w = new Watch;
    Human h2(w);

    //gave the human the watch,
    //they are responsible for managing it
    //so set the original pointer to 0
    //this is much more bulletproof if you use unique_ptr's
    w=0;

    cout<<"when asked the time h2 says: "<<h2.getTime()<<endl;

    return 0;
}
```

1 Has no watch

2 Says “I dont have a watch”

3 Create a watch and give to human

4 Sets original pointer to null

5 Delegates to watch, watch returns time

```
Console  Terminal  History
<terminated> (exit value: 0) Composition_intro [C/C++ Application] /hor
when asked the time h1 says: I dont have a watch
when asked the time h2 says: The time is: 5 minutes after8
```

6

Composition – Summary

Remember composition represents a ‘has a’ relationship

- A car ‘has an’ engine
- A person ‘has a’ watch

In the real world stuff is composed of other stuff
Same here...
An object is composed of other objects
Use pointers to represent these other objects
If the pointer is null, the object is not present
If the pointer is not null delegate the task to the object

```
//how to get time from a human? You ask.
//this is called delegation
//if has a watch, returns time, otherwise says
//I dont have a watch
std::string Human::getTime(){
    if (w)
        return w->getTime();
    else
        return "I dont have a watch";
}
```