Date:

Item 27: Minimise Casting.

→ The rule of C++ are designed to guarantee that type errors are impossible. If we do, we get compile time errors unless we apply casting.

→ C++ offers 4 new casts

(1) const_cast<T>(expression)

used to cast away constness of objects.

(2) Dynamic Cast

used to perform safe downcasting i.e. to determine whether an object is of a particular type in an inheritance hierarchy

(3) Reinterpret_cast

intended for low level cast e.g. casting a pointer to an int. Such casts should be rare.

(4) Static_cast

force implicit conversions. (eg. non-const object to const object, int to double)

Date:.....

void * pointers to typed ptrs, pointer-to-base to ptr-to-derived.
but not cost to non-const object only
const-cast can do that.

=> Old style cast continues to be legal, but we should prefer new style bcz
① easier to identify
② narrowly specified purpose of each cast

=> we can use old-style cast when we want to call explicit ctor to pass an object to a function.

```
class Widget
{ public:
    explicit Widget (int size);
    ...
};
void doSomething (const Widget &w);
```

=> doSomething (Widget(15)); // old style
=> doSomething (static_cast<Widget>(15)); better
// Static cast

Date:.....

The problem with the above code is that the cast does not invoke that function on the current object i.e. cast creates a temporary copy of the base class part of *this, suppose there are any modifictn done to object in base class function, OnResize will modify the copy of base class part *this and not the current object.

Now suppose Special Window OnResize fnctn also does some modificatn to current obj, then, current object will have changes of that, but it will not have changes of base fnctn.

Instead, we can use.

```
virtual void OnResize(){
    Window::OnResize();
}
};
```

Date:.....

Dynamic cast.

→ When we wants to perform derived class operations on what we believe to be derived class object, but we have only a ptr or reference to base through which to manipulate object.

→ If we want a funct in derived class only, instead of doing dynamic cast make the funct virtual in base class

Coz C++ generates code that's big & slow in dynamic-cast.

* Avoid cast wherever possible

* When casting is necessary, hide it inside a funct. Clients can call that instead of you putting cast in their own code.