

Copying and Moving Workshop

Can you finish this for me...?

- `std::thread` is a move-only type in the Standard Library
 - The details of what it does are not important for this exercise
- The code on the next slide is a partially-completed RAII class for managing `std::thread` objects
 - The programmer responsible for it has gone on holiday
 - However, he has written the destructor, which he says is the only non-obvious part of the code
 - He has also left instructions that the class should be move-only and not copyable

Can you finish this for me...? (contd)

```
#include <thread>

class thread_guard {
    std::thread t;
public:
    ~thread_guard() {
        if (t.joinable()) {
            t.join();
        }
    }
};
```

Can you finish this for me...? (contd)

- Complete the class by adding constructors and assignment operators as necessary
- Write a program to test your class
- Check that it supports move operations but not copy operations
- (Depending on your environment, you may need to link against a thread library to get this to compile. Please ask if uncertain!)

- A Camera uses a Memory Card to store images
 - A Memory Card is inserted into the Camera, which takes ownership of it
 - A Memory Card has a storage capacity that is reduced every time the camera takes a picture
 - When the storage capacity reaches zero, the Memory Card is full and is replaced by a fresh Memory Card
- A Memory Card class is shown on the next page

Memory Card class

```
class MemoryCard {  
    int capacity{1000};  
public:  
    class CardFull {};  
    void store() {  
        if (capacity == 0)  
            throw CardFull{};  
        else  
            --capacity;  
    }  
};
```

- Create a Camera class that fulfils the above description and properly manages its resources
 - Camera should be moveable but not copyable
 - Memory Card is both moveable and copyable
- Check your class works with the test program provided on the next slide

Test code

```
Camera makeCamera() {  
    Camera c{new MemoryCard};  
    return c;  
}
```

```
int main() {  
    Camera c = makeCamera();  
    c.take_picture();  
    int images = 1400;  
    // Continued...
```


Test code (contd)

```
while (images--) {  
    try {  
        c.take_picture();  
    }  
    catch (MemoryCard::CardFull) {  
        std::cout << "Memory card is full! Replacing..." << std::endl;  
        c.replace(new MemoryCard);  
    }  
}  
}
```