

Resources and RAI

Agenda

- Resources
 - Definition
 - Examples
- RAIL
 - Definition
 - Concept
 - Example
 - Reasons (not) to use
- Demo



Resources

Resources

Definition:

- Managed parts of program
- Sometimes limited
- Have to be acquired and freed → possible source of problems

Examples:

- Files
- Heap
- Sockets
- Interfaces
- Functionalities beyond scope of program

The image features a dark blue background with abstract teal lines. In the top-left corner, there are several parallel lines that form a corner-like shape. In the bottom-right corner, there are three parallel diagonal lines extending from the bottom edge towards the right edge.

RAII

RAII - General

- „Resource Allocation Is Initialization“
- Widely spread concept in higher programming languages
- Reason: Avoiding problems while managing resources
- Part of some programming language specification (e.g. `std::vector`, `std::string` in C++)

RAII - Concept

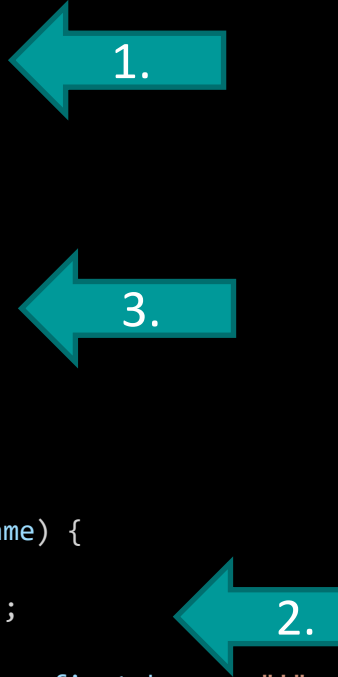
Using classes:

1. Acquire resource in class constructor
2. Use resource
3. Free resource in class destructor

```
#include <iostream>
class FileHandle {
public:
    FileHandle(const char* name) {
        f_ = fopen(name, "r");
    }
    FILE* file() {
        return f_;
    }
    ~FileHandle() {
        if (f_ != nullptr) {
            fclose(f_);
        }
    }
private:
    FILE* f_;
};

void printFirstChar(std::string filename) {
    FileHandle handle(filename.c_str());
    int firstchar = fgetc(handle.file());

    std::cout << "first char in file:" << firstchar << "\n";
}
```



Example in C++

RAII - Example in Java


Try-with-resources:

```
static String readFirstLine(String path) throws IOException {  
    try (BufferedReader br = new BufferedReader(new FileReader(path))) {  
        return br.readLine();  
    }  
}
```

Without try-with-resources:

```
static String readFirstLine(String path) throws IOException {  
    BufferedReader br = new BufferedReader(new FileReader(path));  
    String text = br.readLine();  
    br.close();  
    return text;  
}
```

Freeing resource



RAII - Reasons (not) to use it

+

- Resource handling ensured even if problem while using resource
- Already used in many programming libraries
- Clear distinguishing between resources and other parts of program
- Preventing resource leaks
- "You do not have to care about it while using the resource"

-

- Some very rare cases do not allow use of RAII
- (You feel like „Taking risks is a very good idea. Let's not write clean code.“)



Demo



Thanks.

YOU MAY NOW FREE THIS RESOURCE.