CP317A Software Engineering

week 6-1 – Implementation

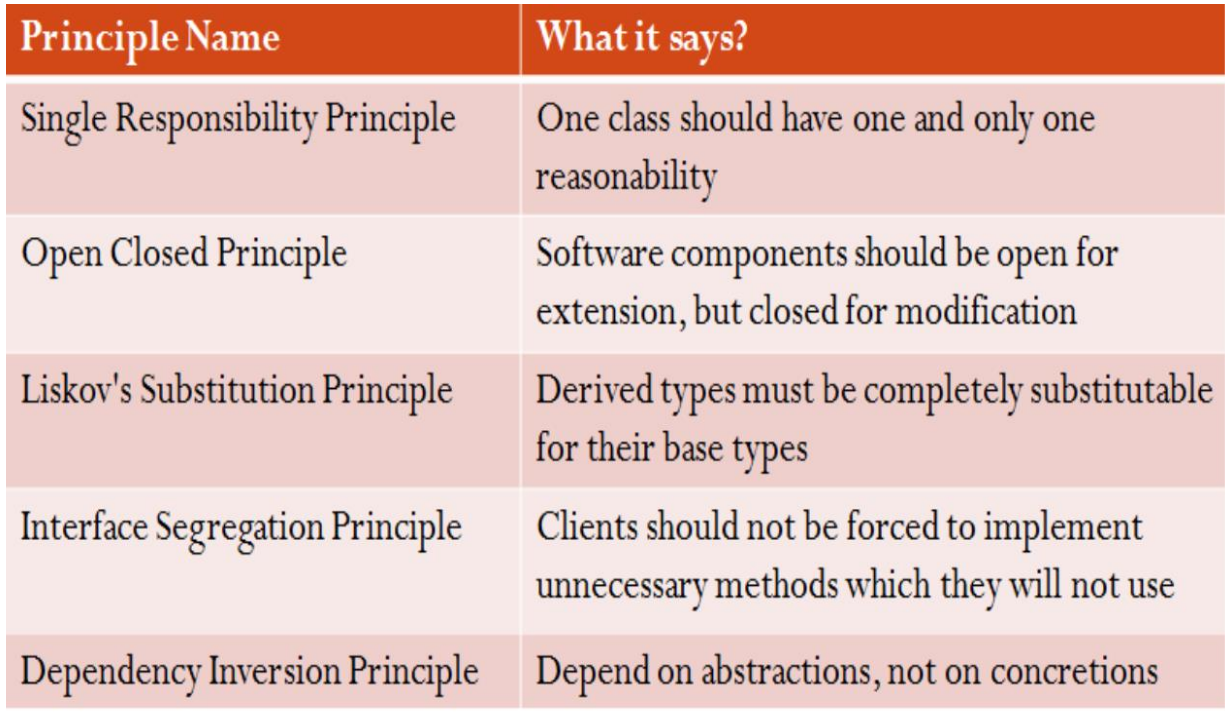
Shaun Gao, Ph.D., P.Eng.

Agenda

- Review week5 topics
- Introduction
  - Good programmer?
  - Computer security
- Coding standard
  - Concept, benefits of using coding standard
- Source code control
- Understanding API
- Recommendations
  - Avoid side effect in programming
  - Offensive programming
  - Exception handling
- Tips for developers
  - Memory consumption
  - Function calls
  - Safety consideration
- Summary

Review week 5 topics

- Object-oriented design
  - Association
  - Aggregation
  - Composition
- SOLID principles
  - Single responsibility principle
  - Open-Close principle
  - Liskov substitution principle
  - Interface segregation principle
  - Dependency inversion principle



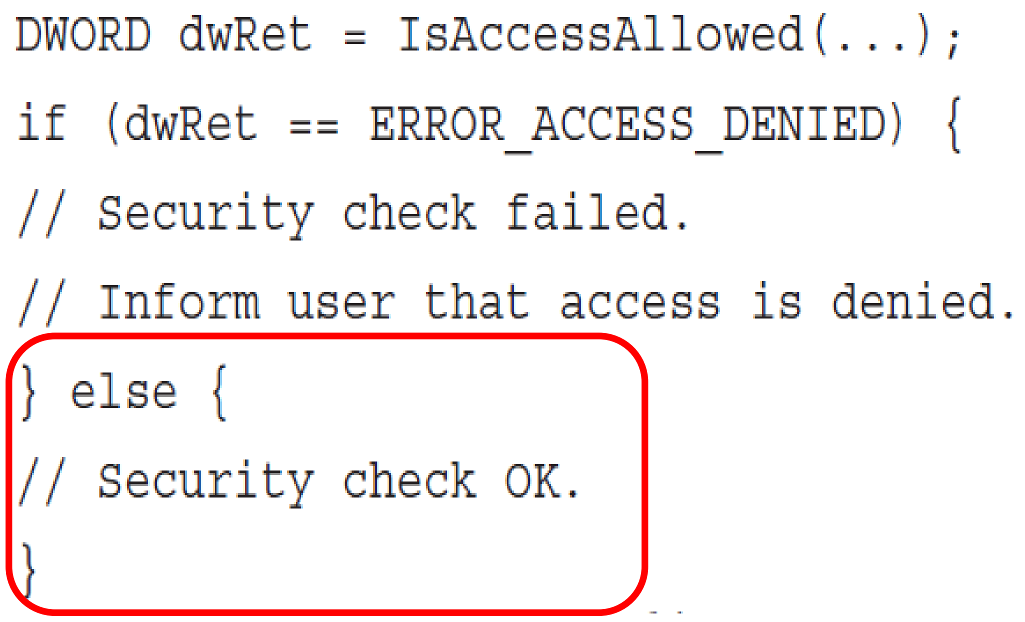
Principle Name	What it says?
Single Responsibility Principle	One class should have one and only one reasonability
Open Closed Principle	Software components should be open for extension, but closed for modification
Liskov's Substitution Principle	Derived types must be completely substitutable for their base types
Interface Segregation Principle	Clients should not be forced to implement unnecessary methods which they will not use
Dependency Inversion Principle	Depend on abstractions, not on concretions

Introduction

- Implementation = coding
- The measurement of becoming a great programmer?
  - Clarify requirements before starting
  - Be a meticulous developer
  - Focus on problem solving
  - Learn to debug (step by step, ASCII table)
  - Review your own code (applying the SOLID principle)
  - Keep learning
  - Get good at Googling

Introduction – cont.

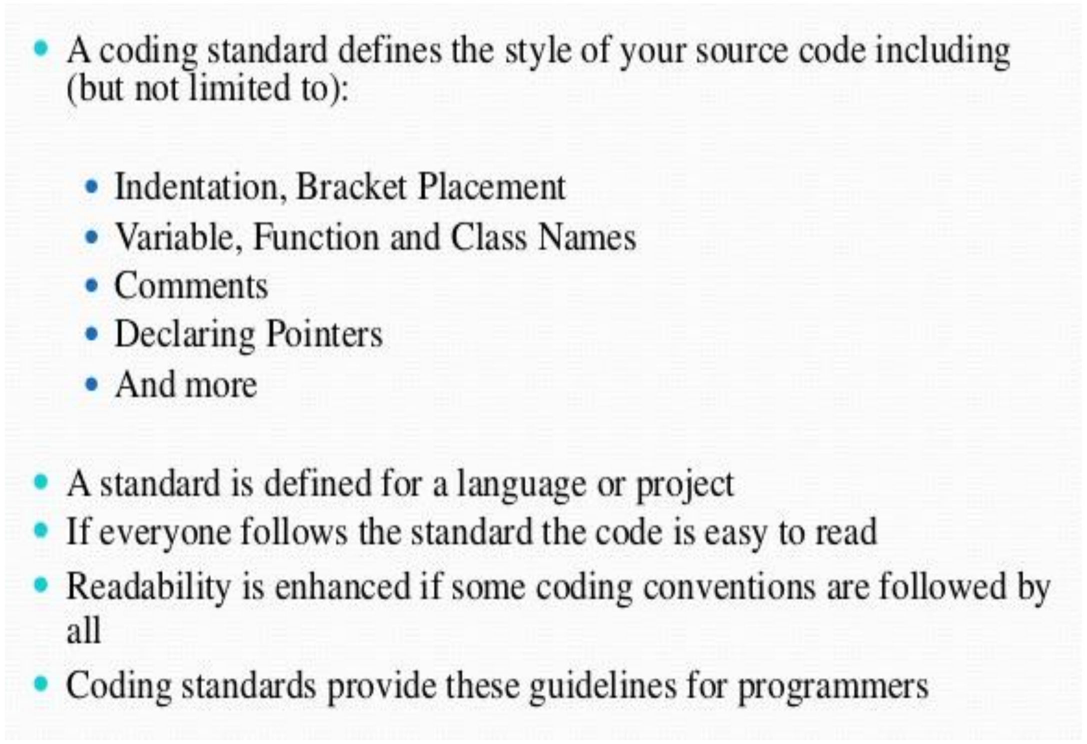
- Software security considerations
- Implementation is a crucial step
- Incorrect coding can cause severe accidents, create system vulnerabilities.
  - Example: Buffer overflow
- Fail-safe default: it states that unless a subject is given explicit access to an object, it should be denied access to that object.
- Examples,



```
DWORD dwRet = IsAccessAllowed(...);
if (dwRet == ERROR_ACCESS_DENIED) {
// Security check failed.
// Inform user that access is denied.
} else {
// Security check OK.
}
```

Coding standard

- A coding standard is a set of guidelines, rules, programming styles and conventions that software developers adhere to when writing source code for a project.
- The benefits of using coding standard
  - Improve software quality
  - Quicker to debug code
  - Easier to code review work
  - Avoid conflict
  - Code is easier to collaborate on



- A coding standard defines the style of your source code including (but not limited to):
  - Indentation, Bracket Placement
  - Variable, Function and Class Names
  - Comments
  - Declaring Pointers
  - And more
- A standard is defined for a language or project
- If everyone follows the standard the code is easy to read
- Readability is enhanced if some coding conventions are followed by all
- Coding standards provide these guidelines for programmers

Tools used for programming

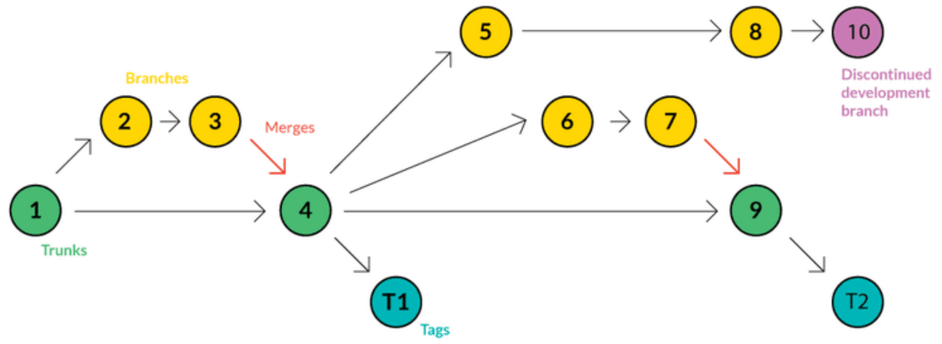
- Compiler and Linker
  - Coding
  - Debugging

- Integrated Development Environment (IDE)

  - Visual Studio - (comprehensive IDE, C/C++, C#, VB, …)
  - Visual Studio Code – (scripting languages, web development)
  - Eclipse
  - Qt…

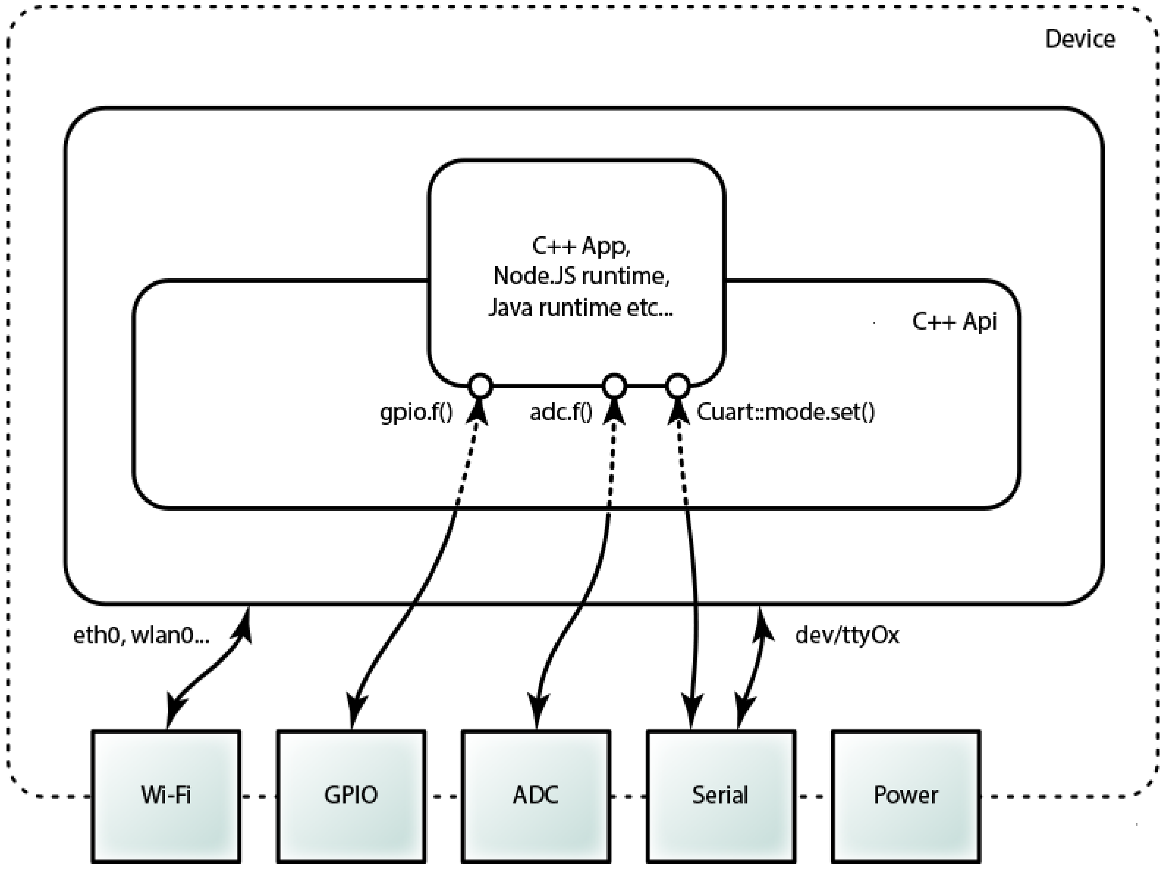
Software code/version control

- Software code/version control is the management of changes to documents, computer programs, and other collections of information.
- Benefits of using software version control
  - Automatic backup
  - Sharing on multiple computers
  - Version control
  - Maintaining different versions
- Version Control system (VCS): Github, ClearCase



Understanding API

- Understand APIs
  - functionalities, inputs, outputs, requiring both computer language skills and operating system knowledge
  - How to use
- Object-oriented programming
  - C++ APIs
  - .NET C# APIs
  - Java API
- **Foundation**: theory and mechanism

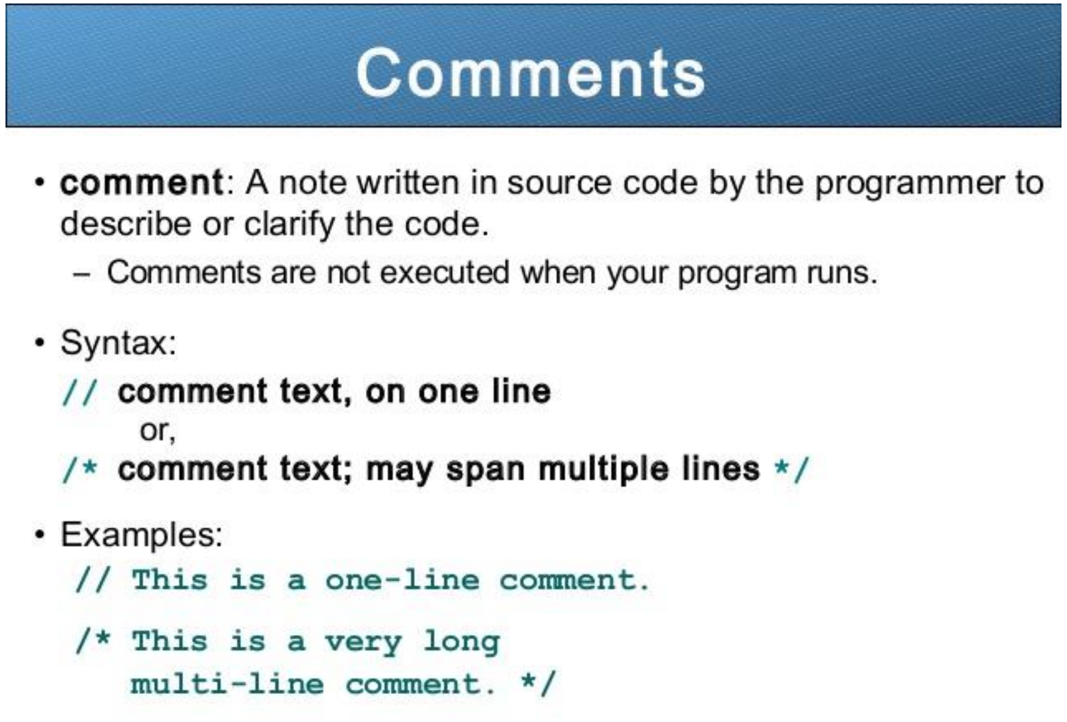


Recommendations

- Be alert:
  - Writing good code can be difficult
    - Working is not enough, high cohesion, low coupling, SOLID principles
    - Coding is another level of design
    - A good programmer must be a good code designer
  - Unexpected situations
    - Example:
      - Call an API, if the API failed, the process continue or stop?
  - Creative thinking
  - Meticulous

Recommendations – cont.

- Comments:
- Write sufficient comments
- Comment outline should be mentioned in the **coding standard**
- Benefits of comments
  - Easy to understand the code
  - Easier maintenance
- Usually specified in coding standard



Comments

- **comment**: A note written in source code by the programmer to describe or clarify the code.
  - Comments are not executed when your program runs.
- Syntax:

```
// comment text, on one line
    or,
/* comment text; may span multiple lines */
```

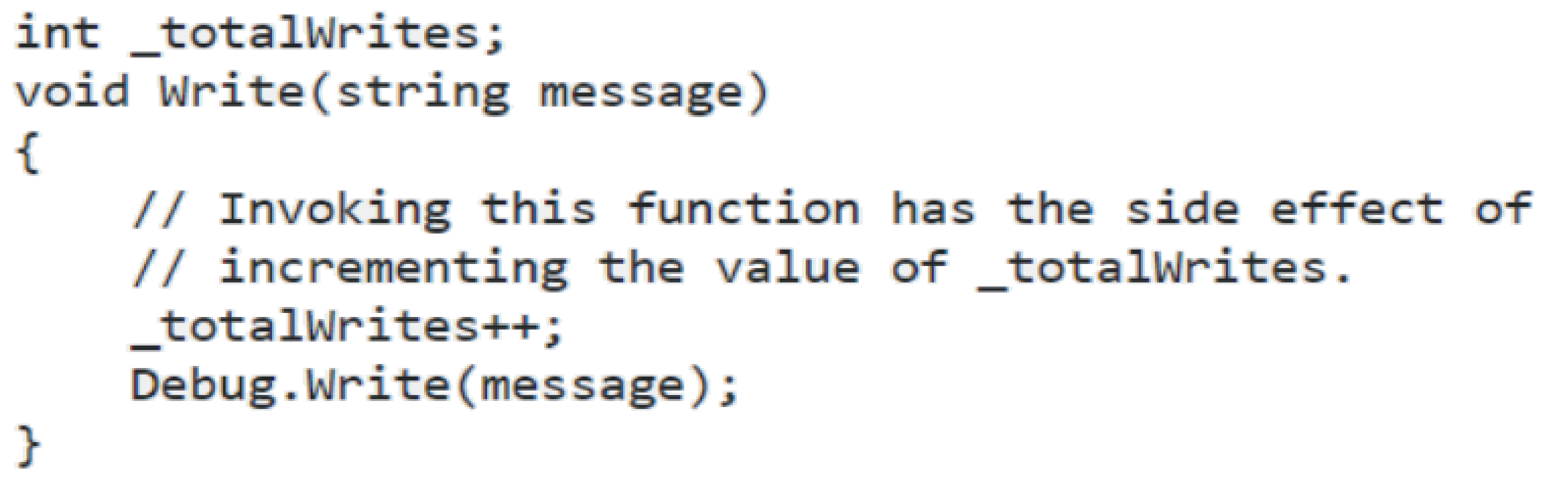
- Examples:

```
// This is a one-line comment.

/* This is a very long
   multi-line comment. */
```

Recommendations – cont.

- Avoid side effects
- What is side effects in software engineering?
  - In software engineering, a function is said to have a side effect if it modifies global variables or has an observable interaction with the outside of its scope
- An example:



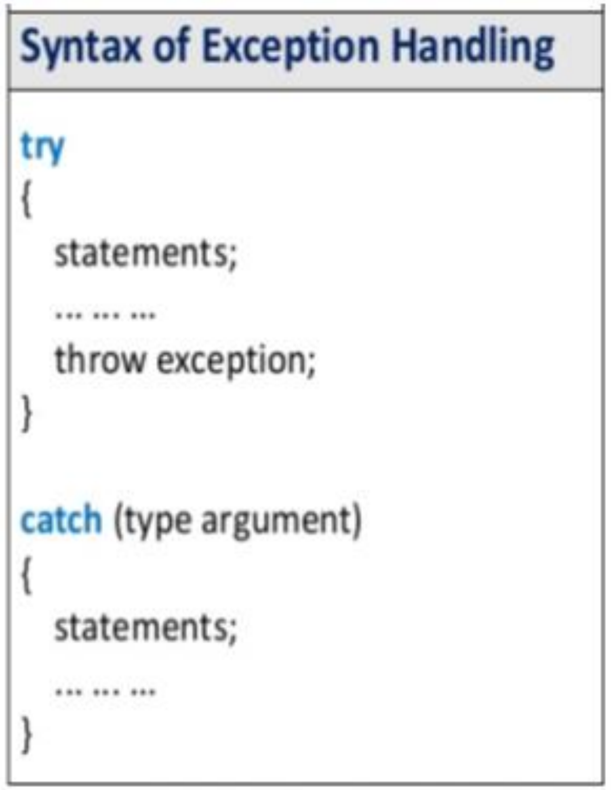
```
int _totalWrites;
void Write(string message)
{
    // Invoking this function has the side effect of
    // incrementing the value of _totalWrites.
    _totalWrites++;
    Debug.Write(message);
}
```

Recommendations – cont.

- Practice offensive programming
- Defensive programming
  - Definition: defensive programming is the process of designing and implementing software so that it continues to function when it is under attack
  - The idea behind defensive programming is that you should expect erroneous input to your functions, or methods.
  - Defensive programming is a good idea?
  - “The best defense is a good offense” – Anonymous
- offensive programming:
  - It departs from defensive principles when dealing with errors resulting from software bugs.
  - It adds an explicit priority of NOT tolerating errors.

Recommendations – cont.

- Exception handling
- What is exception?
- Exception is an unexpected event (could be either inside or outside) that happens while a program is running.
- Benefits:
  - It helps programmers to create reliable system
  - It separates the exception handling code from the main logic
  - It helps to find out problems



Syntax of Exception Handling

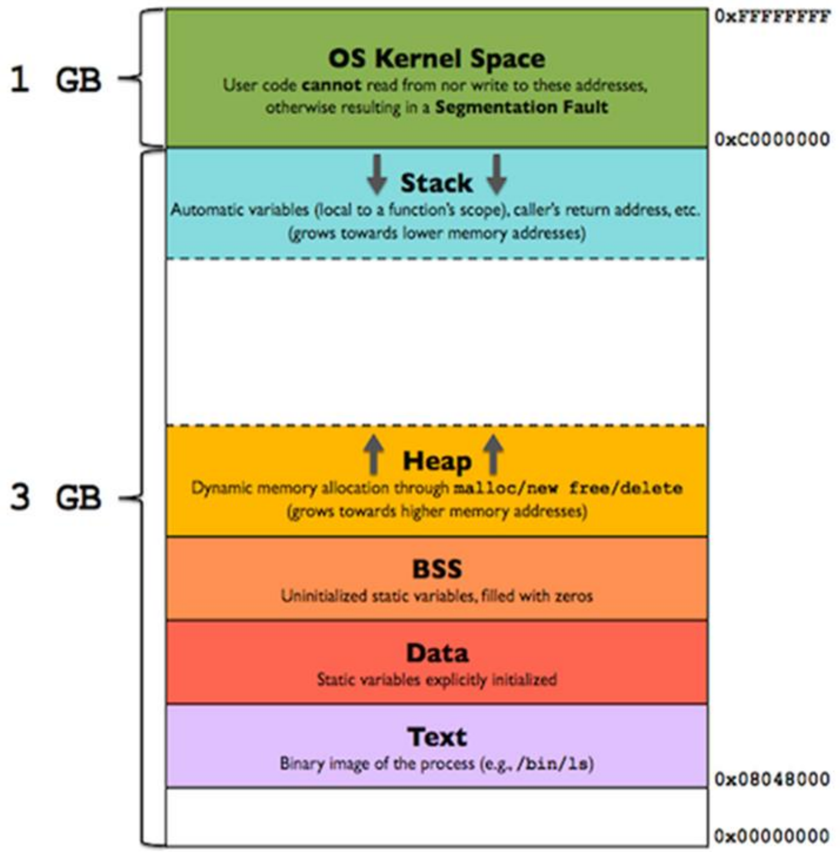
```
try
{
   statements;
   ... ... ...
   throw exception;
}

catch (type argument)
{
   statements;
   ... ... ...
}
```

Tips for developers

- **Memory consumption**
  - **Buffer overflow**:
  - Buffer: it is a region of a physical memory storage used to store data.
  - Buffer overflow is an occurrence when a process or program writes more data into a buffer than the buffer can hold

- Stack based buffer overflow = stack overflow
- Heap based buffer overflow = heap overflow



0xFFFFFFFF
1 GB
OS Kernel Space
User code cannot read from nor write to these addresses, otherwise resulting in a Segmentation Fault
0xC0000000
Stack
Automatic variables (local to a function's scope), caller's return address, etc.
(grows towards lower memory addresses)
3 GB
Heap
Dynamic memory allocation through malloc/new free/delete
(grows towards higher memory addresses)
BSS
Uninitialized static variables, filled with zeros
Data
Static variables explicitly initialized
Text
Binary image of the process (e.g., /bin/ls)
0x08048000
0x00000000

Tips for developers – cont.

- **Memory consumption**
  - **Stack overflow**:
  - A stack overflow is an undesirable condition in which a particular computer program tries to use more memory space than the stack size.

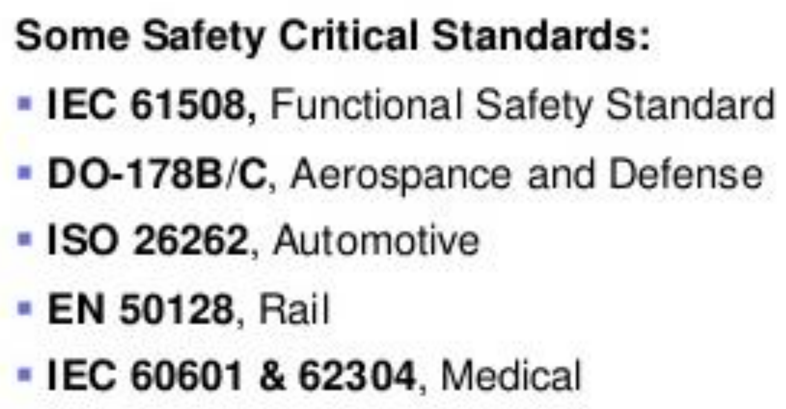
  - **Memory leak**
  - A memory leak is a particular type of memory consumption by running software where the software fails to release memory when it is no longer needed.
    - Result in causing impaired performance or failure.
  - Example: repeat allocating memory, but forget to release them

Tips for developers – cont.

- **Function calls**
  - Minimize the number of local variables
    - If the number of local variables in a function is less, the compiler will be able to fit them into registers. Hence, it will be avoiding frame pointer operations on local variables that are kept on stack. This can improve performance (C/C++)
  - Reduce the number of function parameters.
    - Function calls with large number of parameters may be expensive due to large number of parameter pushes on stack on each call.
  - Avoid cascaded function calls, scripting language may be OK, but not compile language
    - FuncX(funcA(), funcB());
    - Hard for debugging

Tips for developers – cont.

- Safety consideration
  - Safety critical systems: A safety-critical system is a system whose failure or malfunction may result in serious injury or even death of people.
    - Medical devices,
    - Airplanes,
    - Automatic control train system
  - Real time operating systems (RTOS)
    - A RTOS processes data and executes tasks within strict time constraints and with a high degree of reliability and precision.

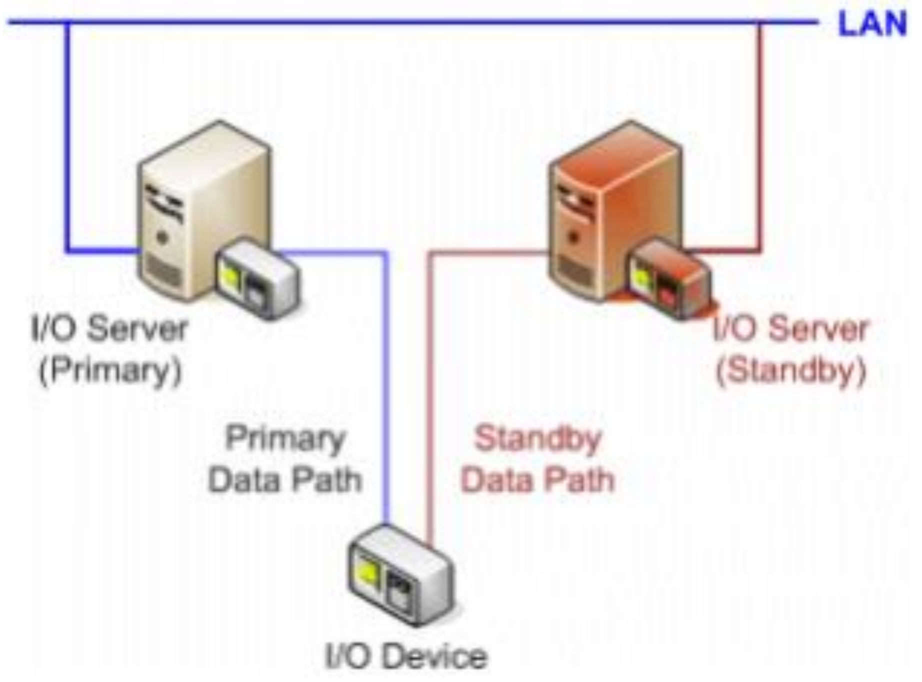


Some Safety Critical Standards:

- **IEC 61508,** Functional Safety Standard
- **DO-178B/C**, Aerospance and Defense
- **ISO 26262**, Automotive
- **EN 50128**, Rail
- **IEC 60601 & 62304**, Medical

Tips for developers – cont.

- Safety consideration – cont.
  - Redundancy: In software engineering, it is a design method in which a component is duplicated so if it fails there will be a backup.
    - Redundancy is good from safety perspective.
  - System redundancy: Redundancy is a widely used method of improving the reliability of computerized systems.
  - Examples: online banking systems, control systems, Hadoop
- Note:
  - Redundancy sometimes desirable, sometime not
    - Example: Database design



Tips for developers – cont.

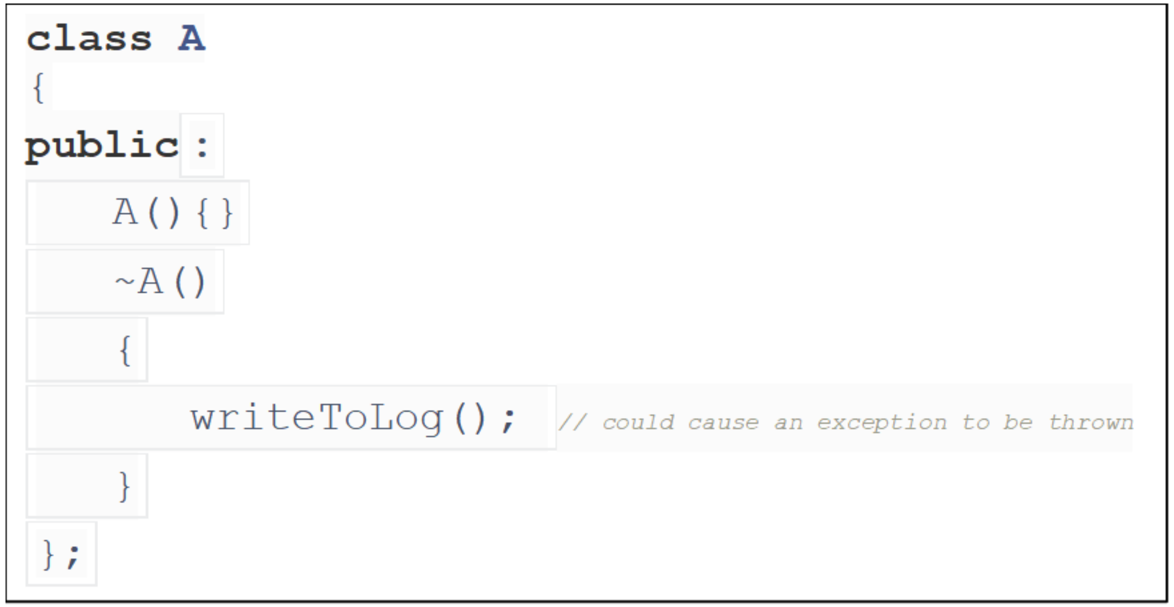
- **Constructor and destructors in Object Oriented Programming**
  - Default constructor and destructor

- Apply Single-Responsibility principle
  - Constructors as lightweight (simple) as possible
    - Only perform necessary initialization including resource allocations
  - Destructors
    - Focus on releasing resources that have been allocated in heap

  - Use initialization instead of assignment
    Color c(black); is faster than Color c; c = black;.

Tips for developers – cont.

- **Keep Destructors simple**



```
class A
{
public:
    A(){}
    ~A()
    {
        writeToLog(); // could cause an exception to be thrown
    }
};
```

  - the general rule is: never allow exceptions to leave destructors.

Summary

- Coding standard
    - Concept, benefits of using coding standard
- Source code control
- Understanding API
- Recommendations
    - Avoid side effect in programming
    - Offensive programming
    - Exception handling
- Tips for developers
    - Memory managements
    - Function calls
    - Safety consideration
    - Constructor/destructor
    - Keep destructor focus on releasing resources only

Announcement

- Start group project

- 9 of you do not have a group. Please let me know if you need help.