CME 211: Lecture 23

Topic: C++ Object Oriented Programming

A simple class

src/class1.cpp:

```
#include <string>

class user {
  // data members
  int id;
  std::string name;
};

int main()
{
  user u; // object (instance of the class)
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/class1.cpp -o src/class1
src/class1.cpp:5:7: warning: private field 'id' is not used [-Wunused-private-field]
  int id;
      ^
1 warning generated.
$ ./src/class1
```

Member access

src/class2.cpp:

```
#include <iostream>
#include <string>

class user
{
  int id;
  std::string name;
};

int main()
{
  user u;
  u.id = 7; // Member access via dot notation
  std::cout << "u.id = " << u.id << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/class2.cpp -o src/class2
src/class2.cpp:13:5: error: 'id' is a private member of 'user'
  u.id = 7; // Member access via dot notation
    ^
src/class2.cpp:6:7: note: implicitly declared private here
  int id;
      ^
src/class2.cpp:14:31: error: 'id' is a private member of 'user'
  std::cout << "u.id = " << u.id << std::endl;
                              ^
src/class2.cpp:6:7: note: implicitly declared private here
  int id;
      ^
2 errors generated.
```

Member access

src/struct1.cpp:

```cpp
#include <iostream>
#include <string>

struct user
{
  int id;
  std::string name;
};

int main()
{
  user u;
  u.id = 7;
  std::cout << "u.id = " << u.id << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/struct1.cpp -o src/struct1
$ ./src/struct1
u.id = 7
```

Member access

- C++ is strict about member access
- Need to know about default behavior
- And how to override defaults via access specifiers

Access specifiers

- `private`: data or method member only accessible from within member(s) of the same class
- `public`: data or method member accessible by anyone using dot notation

- Default access specifier for `class` is `private`
- Default access specifier for `struct` is `public`

Overriding default access

src/class3.cpp:

```
#include <iostream>
#include <string>

class user {
 public: // everything after this will be public
  int id;
  std::string name;
};

int main() {
  user u;
  u.id = 7;
  u.name = "Leland";
  std::cout << "u.id = " << u.id << std::endl;
  std::cout << "u.name = " << u.name << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/class3.cpp -o src/class3
$ ./src/class3
u.id = 7
u.name = Leland
```

Overriding default access

src/struct2.cpp:

```
#include <iostream>
#include <string>

struct user {
  int id;
 private: // everything after this will be private
  std::string name;
};

int main() {
  user u;
  u.id = 7;
  u.name = "Leland";
  std::cout << "u.id = " << u.id << std::endl;
  std::cout << "u.name = " << u.name << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/struct2.cpp -o src/struct2
src/struct2.cpp:13:5: error: 'name' is a private member of 'user'
  u.name = "Leland";
    ^
src/struct2.cpp:7:15: note: declared private here
  std::string name;
              ^
src/struct2.cpp:15:33: error: 'name' is a private member of 'user'
  std::cout << "u.name = " << u.name << std::endl;
                                ^
src/struct2.cpp:7:15: note: declared private here
  std::string name;
              ^
2 errors generated.
```

Mix and match

src/class4.cpp:

```
#include <iostream>
#include <string>

class user {
  int id;
 public:
  std::string name;
 private:
  int age;
};

int main() {
  user u;
  u.id = 7;
  u.name = "Leland";
  u.age = 12;

  std::cout << "u.id = " << u.id << std::endl;
  std::cout << "u.name = " << u.name << std::endl;
  std::cout << "u.age = " << u.age << std::endl;

  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/class4.cpp -o src/class4
src/class4.cpp:14:5: error: 'id' is a private member of 'user'
  u.id = 7;
    ^
src/class4.cpp:5:7: note: implicitly declared private here
  int id;
      ^
src/class4.cpp:16:5: error: 'age' is a private member of 'user'
  u.age = 12;
    ^
```

```
src/class4.cpp:9:7: note: declared private here
  int age;
      ^
src/class4.cpp:18:31: error: 'id' is a private member of 'user'
  std::cout << "u.id = " << u.id << std::endl;
                              ^
src/class4.cpp:5:7: note: implicitly declared private here
  int id;
      ^
src/class4.cpp:20:32: error: 'age' is a private member of 'user'
  std::cout << "u.age = " << u.age << std::endl;
                               ^
src/class4.cpp:9:7: note: declared private here
  int age;
      ^
4 errors generated.
```

"Adding" a member

src/struct3.cpp:

```
#include <iostream>

struct user {
  int id;
};

int main() {
  user u;
  u.id = 7;
  u.age = 12;
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/struct3.cpp -o src/struct3
src/struct3.cpp:10:5: error: no member named 'age' in 'user'
  u.age = 12;
  ~ ^
1 error generated.
```

Our first method

src/class5.cpp:

```
#include <iostream>

class user {
  // data member initialization is a C++11 feature
  int id = 7;
  int getId(void) {
    return id;
  }
};
```

```
int main() {
  user u;
  std::cout << "u.getId() = " << u.getId() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -std=c++11 -Wall -Wextra -Wconversion src/class5.cpp -o src/class5
src/class5.cpp:13:36: error: 'getId' is a private member of 'user'
  std::cout << "u.getId() = " << u.getId() << std::endl;
                                   ^
src/class5.cpp:6:7: note: implicitly declared private here
  int getId(void) {
      ^
1 error generated.
```

Our first method

src/class6.cpp:

```
#include <iostream>

class user {
  int id = 7;
 public:
  int getId(void) {
    return id;
  }
};

int main() {
  user u;
  std::cout << "u.getId() = " << u.getId() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -std=c++11 -Wall -Wextra -Wconversion src/class6.cpp -o src/class6
$ ./src/class6
u.getId() = 7
```

Naming

src/class7.cpp:

```
#include <iostream>

class user {
  int id = 1;
 public:
  int getId(void) { return id; }
  void setId(int id) { id = id; }
};
```

```
int main() {
  user u;
  u.setId(7);
  std::cout << "u.getId() = " << u.getId() << std::endl;
  u.setId(42);
  std::cout << "u.getId() = " << u.getId() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -std=c++11 -Wall -Wextra -Wconversion src/class7.cpp -o src/class7
src/class7.cpp:7:27: warning: explicitly assigning value of variable of type 'int' to itself [-Wself-ass
  void setId(int id) { id = id; }
                       ~~ ^ ~~
1 warning generated.
$ ./src/class7
u.getId() = 1
u.getId() = 1
```

One solution

src/class8.cpp:

```
#include <iostream>

class user {
  int id = 1;
 public:
  int getId(void) { return id; }
  void setId(int id_) { id = id_; }
};

int main()
{
  user u;
  u.setId(7);
  std::cout << "u.getId() = " << u.getId() << std::endl;
  u.setId(42);
  std::cout << "u.getId() = " << u.getId() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -std=c++11 -Wall -Wextra -Wconversion src/class8.cpp -o src/class8
$ ./src/class8
u.getId() = 7
u.getId() = 42
```

this

src/class9.cpp:

```cpp
#include <iostream>

class user {
  int id = 1;
 public:
  int getId(void) { return id; }
  void setId(int id) { this->id = id; }
};

int main() {
  user u;
  u.setId(7);
  std::cout << "u.getId() = " << u.getId() << std::endl;
  u.setId(42);
  std::cout << "u.getId() = " << u.getId() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -std=c++11 -Wall -Wextra -Wconversion src/class9.cpp -o src/class9
$ ./src/class9
u.getId() = 7
u.getId() = 42
```

Constructor

- Special method called when a new object of the class is created
- C++ provides a default constructor that takes no arguments
- You can replace the default constructor with a custom constructor by defining a method name with the same name as the class
- Like other methods, the constructor can take arguments
- Does not return anything, not even void

Constructor example

src/class10.cpp:

```cpp
#include <iostream>

class user {
  int id;
 public:
  user(int id) { this->id = id; }
  int getId(void) { return id; }
};

int main() {
  user u(13);
  std::cout << "u.getId() = " << u.getId() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -std=c++11 -Wall -Wextra -Wconversion src/class10.cpp -o src/class10
$ ./src/class10
u.getId() = 13
```

Constructor example

src/class11.cpp:

```
#include <iostream>

class user {
  int id;
 public:
  user(int id) { this->id = id; }
  int getId(void) { return id; }
};

int main() {
  user u;
  std::cout << "u.getId() = " << u.getId() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -std=c++11 -Wall -Wextra -Wconversion src/class11.cpp -o src/class11
src/class11.cpp:11:8: error: no matching constructor for initialization of 'user'
  user u;
       ^
src/class11.cpp:6:3: note: candidate constructor not viable: requires single argument 'id', but no argu
  user(int id) { this->id = id; }
  ^
src/class11.cpp:3:7: note: candidate constructor (the implicit copy constructor) not viable: requires 1
class user {
      ^
src/class11.cpp:3:7: note: candidate constructor (the implicit move constructor) not viable: requires 1
1 error generated.
```

Circle example

src/circle1.cpp:

```
#include <cmath>
#include <iostream>

class circle {
  double x, y, r;
 public:
  circle(double x, double y, double r) {
    this->x = x;
    this->y = y;
    this->r = r;
  }
```

```
  double getArea(void) {
    return M_PI*r*r;
  }
};

int main() {
  circle c(1.2, 3.4, 2.);
  std::cout << "c.getArea() = " << c.getArea() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/circle1.cpp -o src/circle1
$ ./src/circle1
c.getArea() = 12.5664
```

Multiple files

src/circle2.hpp:

```
#ifndef CIRCLE2_HPP
#define CIRCLE2_HPP

class circle {
  double x, y, r;
 public:
  circle(double x, double y, double r);
  double getArea(void);
};

#endif /* CIRCLE2_HPP */
```

src/circle2.cpp:

```
#include <cmath>

#include "circle2.hpp"

circle::circle(double x, double y, double r) {
  this->x = x;
  this->y = y;
  this->r = r;
}

double circle::getArea(void) {
  return M_PI*r*r;
}
```

src/main2.cpp:

```
#include <iostream>

#include "circle2.hpp"

int main() {
  circle c(1.2, 3.4, 2.);
```

```
  std::cout << "c.getArea() = " << c.getArea() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/circle2.cpp src/main2.cpp -o src/main2 -I./src
$ ./src/main2
c.getArea() = 12.5664
```

Multiple files, example 2

src/circle3.hpp:

```
#ifndef CIRCLE3_HPP
#define CIRCLE3_HPP

namespace geometry {

class circle {
  double x, y, r;
 public:
  circle(double x, double y, double r);
  double getArea(void);
  double getPerimeter(void);
};

}

#endif /* CIRCLE3_HPP */
```

src/circle3a.cpp:

```
#include <cmath>

#include "circle3.hpp"

namespace geometry {

circle::circle(double x, double y, double r) {
  this->x = x;
  this->y = y;
  this->r = r;
}

double circle::getArea(void) {
  return M_PI*r*r;
}

}
```

src/circle3b.cpp:

```
#include <cmath>

#include "circle3.hpp"
```

```
namespace geometry {

double circle::getPerimeter(void) {
  return 2.*M_PI*r;
}

}
```

src/main3.cpp:

```
#include <iostream>

#include "circle3.hpp"

int main() {
  geometry::circle c(1.2, 3.4, 1.8);
  std::cout << "c.getArea() = " << c.getArea() << std::endl;
  std::cout << "c.getPerimeter() = " << c.getPerimeter() << std::endl;
  return 0;
}
```

Output:

```
$ clang++ -Wall -Wextra -Wconversion src/circle3a.cpp src/circle3b.cpp src/main3.cpp -o src/main3 -I./s
$ ./src/main3
c.getArea() = 10.1788
c.getPerimeter() = 11.3097
```

Objects and containers

src/container.cpp:

```
#include <iostream>
#include <vector>

#include "circle3.hpp"

int main() {
  std::vector<geometry::circle> circles;
  circles.emplace_back(8.3, 4.7, 0.5);
  circles.emplace_back(4.1, 2.3, 1.4);
  circles.emplace_back(-3.2, 0.8, 14.4);

  for(auto& c : circles) {
    std::cout << "c.getArea() = " << c.getArea() << std::endl;
  }

  return 0;
}
```

Output:

```
$ clang++ -std=c++11 -Wall -Wextra -Wconversion src/circle3a.cpp src/circle3b.cpp src/container.cpp -o s
$ ./src/container
c.getArea() = 0.785398
c.getArea() = 6.15752
c.getArea() = 651.441
```

Reading

C++ Primer, Fifth Edition by Lippman et al.

- Section 1.5: Introducing Classes