Intermediate programming(C++)-

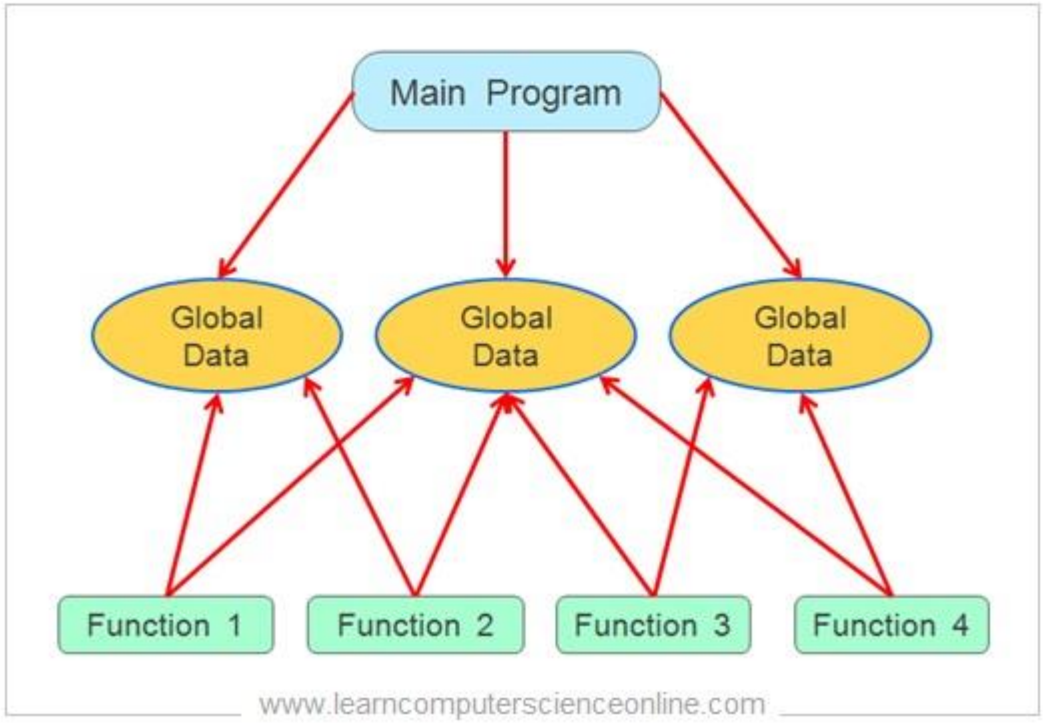
Lab 9: OOP – Part 1

CSCI112
Nile University

Content

Procedural programming

- The program is organized as a set of functions.
- The data and functions are global/ public and accessible in any part of the program.

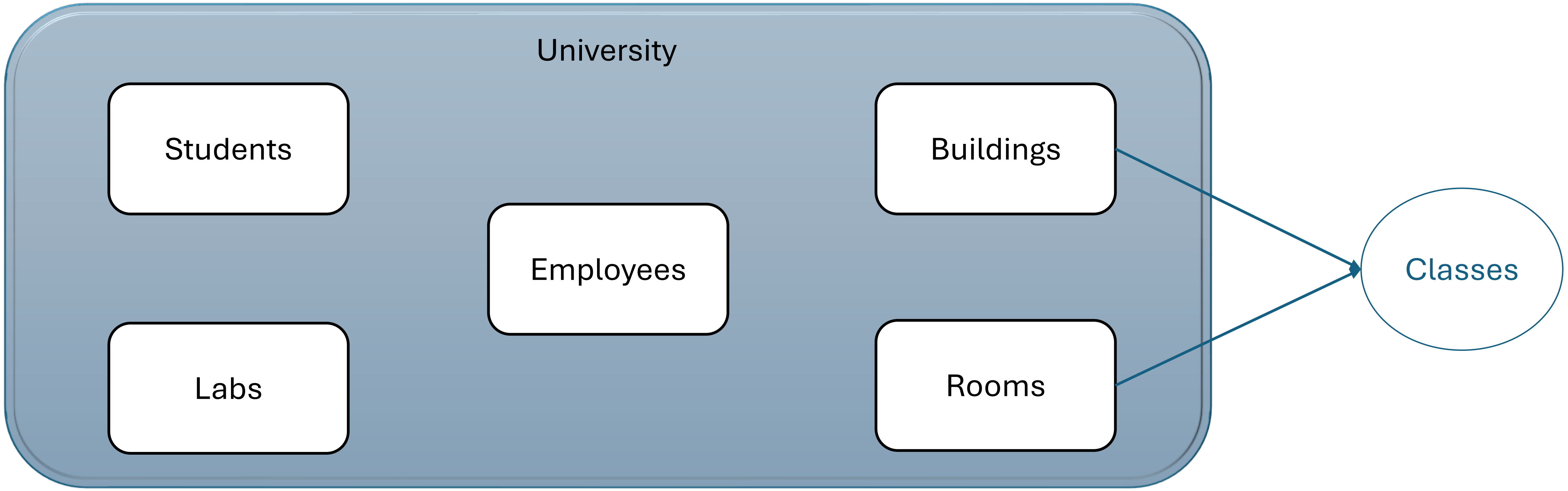


https://www.learncomputerscienceonline.com/procedural-programming/

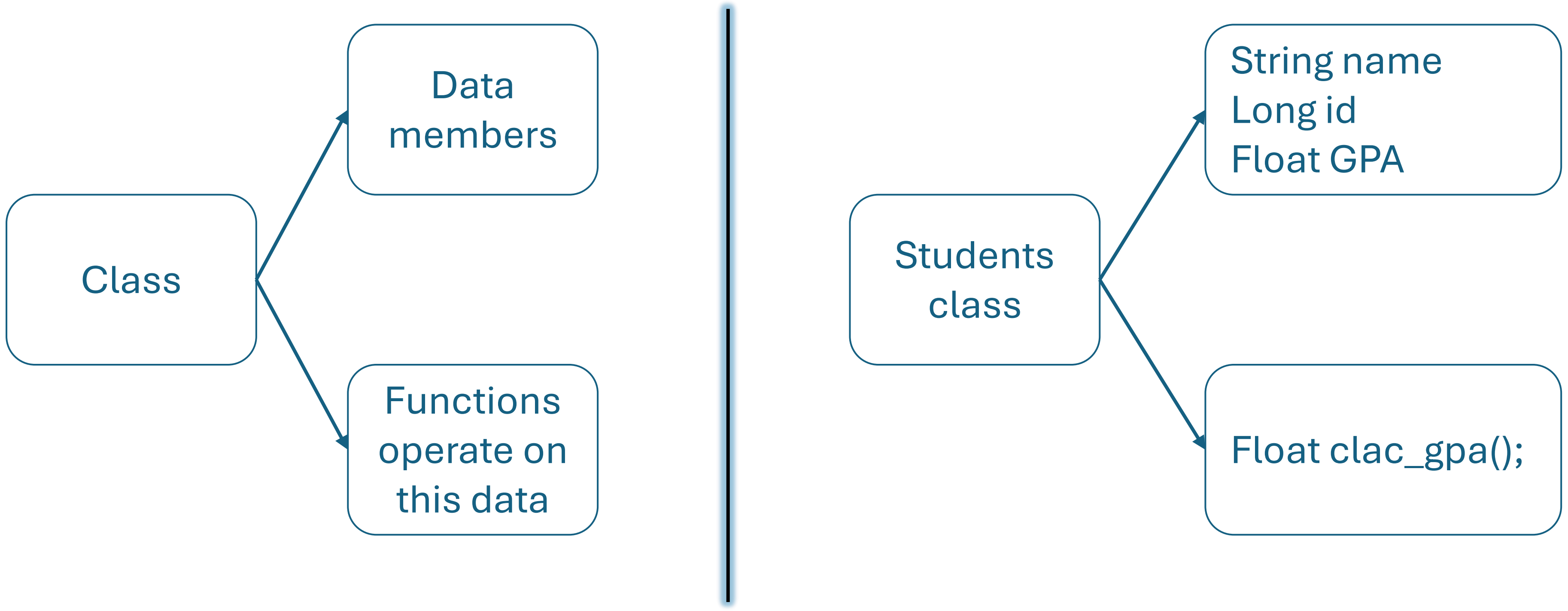
Object oriented programming

A programming(Design) style, it doesn't refer to a specific programming language, but it is a way to build the program by making simulation of reality.

The big program/ organization is divided into entities (e.g., smaller parts), for example, the University is divided into students, Employees, Buildings, Labs, Rooms, etc..



Classes


Class
Data members
Functions operate on this data
Students class
String name
Long id
Float GPA
Float clac_gpa();

Example

```
class Students {
    // Data members

    string name;

    long int id;

    float gpa;

    // functions members

    float calc_gpa(int course1_grade, int course2_grade, int course3_grade) {

        int sum_marks = course1_grade + course2_grade + course3_grade;

        return (sum_marks / 100) * 100;
    }
};
```

Both data and functions members are private by default/ not accessible at any other part of the program → OOP Provides data hiding features → The ability to disable accessing particular important data outside the class

How to use the class in main?

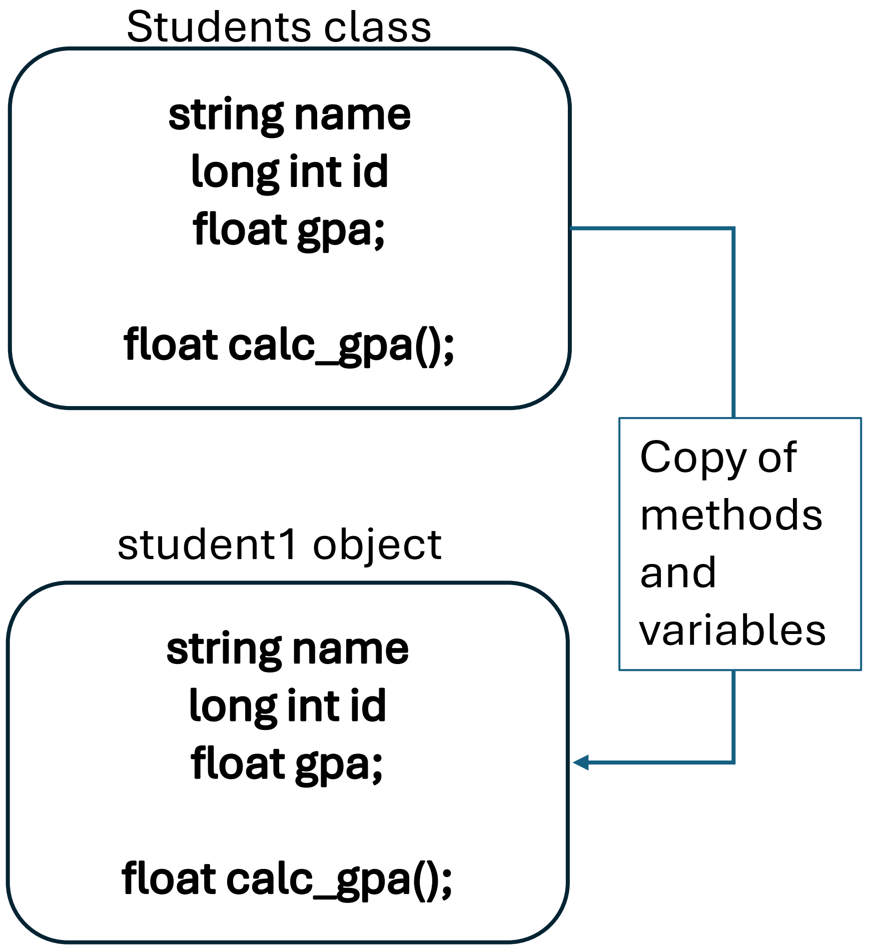
To be able to use classes in the main function, we need to create an object/ instance of it.

If we created a class called Students and we have inside it 3 member data/ variables [string name, long int id, float gpa], and one method/ function members [float calc_gpa()], and we want to use this class for different students in main function.

- Firstly, we create the object: `Students student 1;`
- Secondly, to access any of member variable or methods, we use . Operator:

```
student1.name = “Ahmed Mohsen”;
student1.id = 239001;
student1.gpa = 3.4;
```

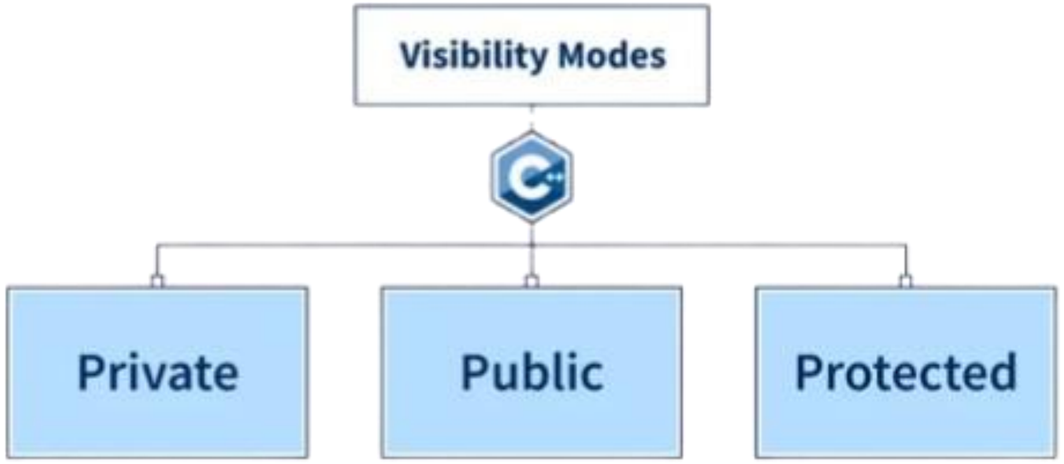
- When we create an instance of the class, a separate copy of each member variable and method is taken for each instance.

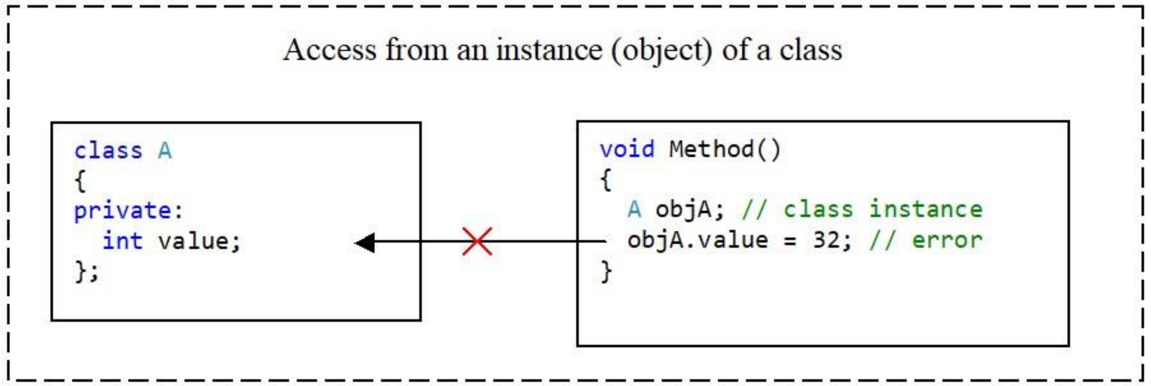


Example

```
class Students {
private:
    // Data members
    string name;
    long int id;
    float gpa;
public:
    // functions members
    float calc_gpa(int course1_grade, int course2_grade, int course3_grade) {
        int sum_marks = course1_grade + course2_grade + course3_grade;
        return (sum_marks / 100) * 100;
    }
};
```

Access Modifiers (e.g., access specifiers/ visibility modes)





```
class A
{
private:
  int value;
};
```

```
void Method()
{
  A objA; // class instance
  objA.value = 32; // error
}
```

- **Private**: Accessible inside the class only
- **Public**: Accessible inside and outside the class.
- **Protected**: Later..

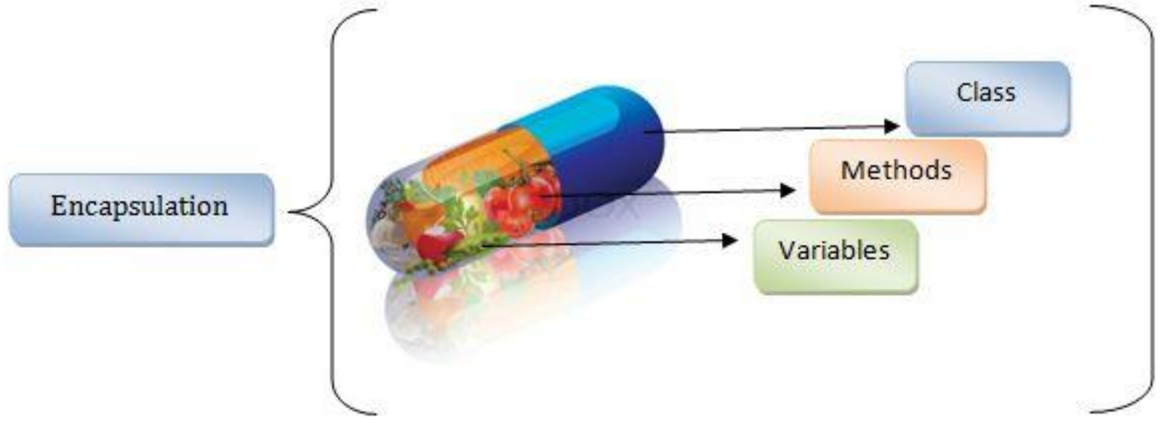
public:

```
class Students {

public:
    // Data members

    string name;

    long int id;

    float gpa;

    // functions members
    float calc_gpa(int course1_grade, int course2_grade, int course3_grade) {
       int sum_marks = course1_grade + course2_grade + course3_grade;
       return (sum_marks / 100) * 100;
    }
};


int main() {
    Students student1;
    student1.name = "Ahmed Mohsen";
    student1.id = 23100;
    student1.gpa = 3.8;
    student1.calc_gpa(80, 90, 99);
}
```

Private:

```
#include <iostream>
using namespace std;

class Students {
private:
    // Data members
    string name;
    long int id;
    float gpa;

public:
    // functions members
    float calc_gpa(int course1_grade, int course2_grade, int course3_grade) {
        int sum_marks = course1_grade + course2_grade + course3_grade;
        return (sum_marks / 100) * 100;
    }
};


int main() {
    Students student1;
    student1.name = "Ahmed Mohsen";
    student1.id = 23100;
    student1.gpa = 3.8;
    student1.calc_gpa(80, 90, 99);
}
```

Encapsulation

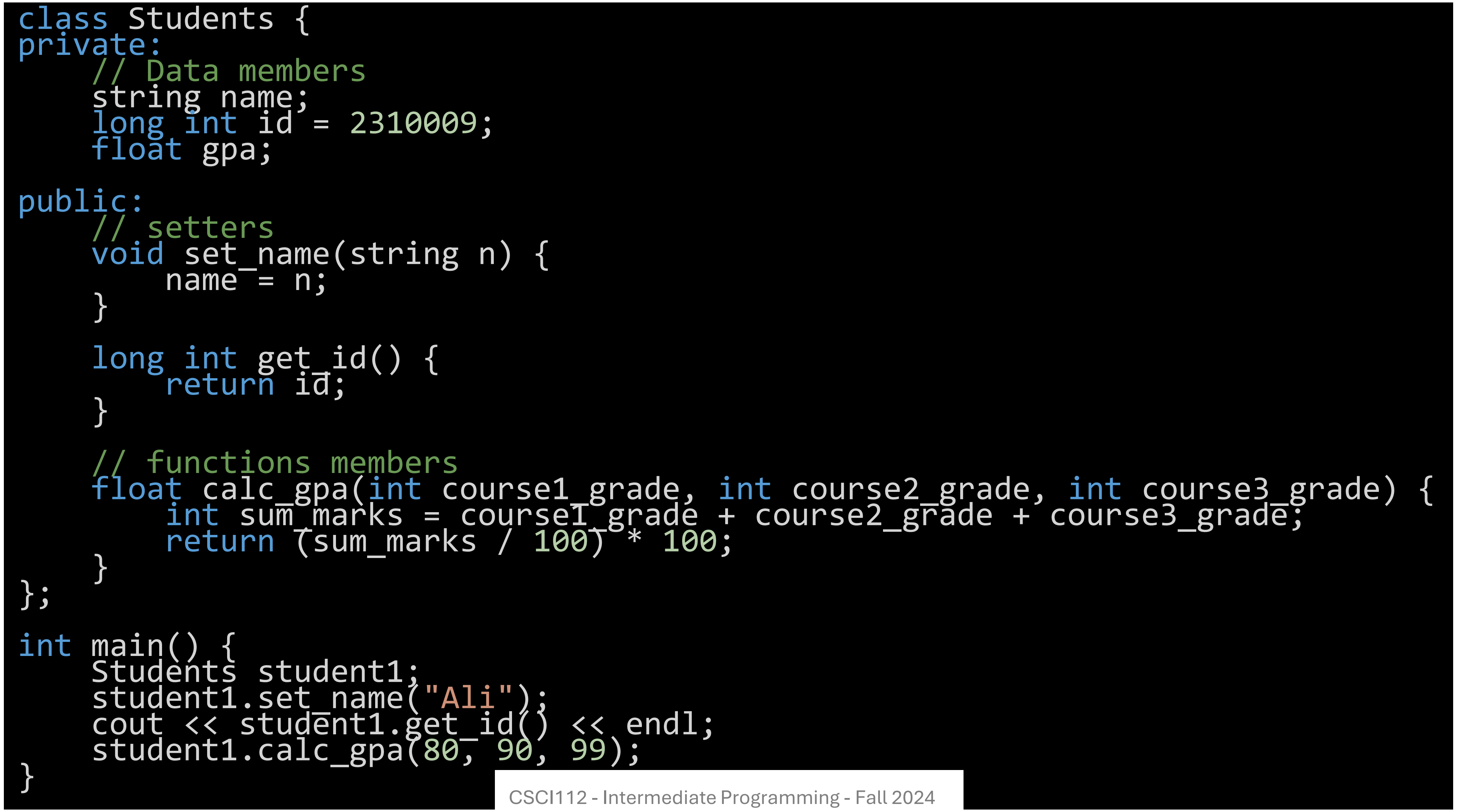


Class encapsulate its components (e.g., methods and variables) in one closed block, meaning that the only one who control the accessibility of these components is the owner of the class.

Setters

```cpp
class Students {
private:
    // Data members
    string name;
    long int id;
    float gpa;

public:
    // setters
    void set_name(string n) {
        name = n;
    }

    void set_id(long int ID) {
        id = ID;
    }

    // functions members
    float calc_gpa(int course1_grade, int course2_grade, int course3_grade) {
        int sum_marks = course1_grade + course2_grade + course3_grade;
        return (sum_marks / 100) * 100;
    }
};

int main() {
    Students student1;
    student1.set_name("Ali");
    student1.set_id(2310009);
    student1.calc_gpa(80, 90, 99);
}
```

Getters



```cpp
class Students {
private:
    // Data members
    string name;
    long int id = 2310009;
    float gpa;

public:
    // setters
    void set_name(string n) {
        name = n;
    }

    long int get_id() {
        return id;
    }

    // functions members
    float calc_gpa(int course1_grade, int course2_grade, int course3_grade) {
        int sum_marks = course1_grade + course2_grade + course3_grade;
        return (sum_marks / 100) * 100;
    }
};

int main() {
    Students student1;
    student1.set_name("Ali");
    cout << student1.get_id() << endl;
    student1.calc_gpa(80, 90, 99);
}
```

Constructors:: Empty constructor

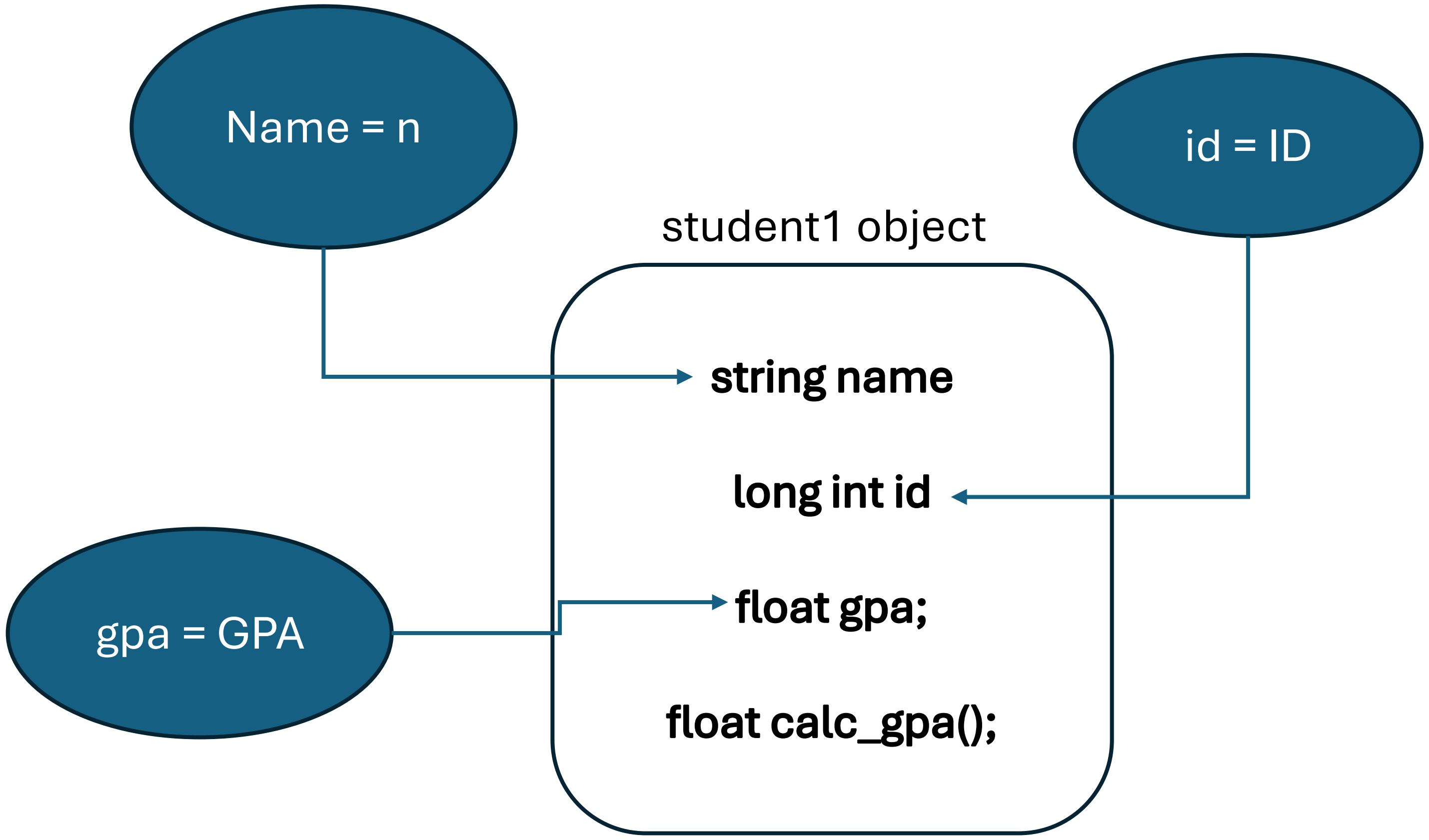
```cpp
class Students {
public:
    // Data members
    string name;

    long int id;

    float gpa;

    Students() {
        name = "No one";
        id = 0;
        gpa = 0.0;
    }
};


int main() {
    Students student1;
    cout << "name: " << student1.name << endl;
    cout << "ID: " << student1.id << endl;
    cout << "GPA:" << student1.gpa << endl;
}
```

Constructors:: Paramterized constructor

```cpp
class Students {
public:
    // Data members
    string name;
    long int id;
    float gpa;

    Students(string n, long int ID, float GPA) {
        name = n;
        id = ID;
        gpa = GPA;
    }
};

int main() {
    Students student1 ("Ali", 24590, 4);
    cout << "name: " << student1.name << endl;  // Ali
    cout << "ID: " << student1.id << endl;  // 24590
    cout << "GPA:" << student1.gpa << endl;  // 4
}
```

Constructors:: Paramterized constructor

```
Students(string n, long int ID, float GPA);
```



Task: Manage a Library Book System

You are tasked with creating a program that simulates a basic library system. The system should manage books and their statuses.

Requirements:

- **Create a Book Class:**
  - **Private members:**
    - title (string): the title of the book.
    - author (string): the name of the author.
    - isBorrowed (bool): whether the book is currently borrowed or not.
  - **Public members:**
    - **Constructors:**
      - Empty constructor: Initializes title and author to empty strings and isBorrowed to false.
      - Parameterized constructor: Takes title and author as arguments and initializes isBorrowed to false.
    - **Setters and getters:**
      - Provide setters and getters for title and author.
      - Provide a getter for isBorrowed (no setter for isBorrowed to enforce controlled access).
    - **Other methods:**
      - borrowBook(): Sets isBorrowed to true if the book is not already borrowed.
      - Prints a message if the book is already borrowed.
      - returnBook(): Sets isBorrowed to false.
- **The main function:**
  - Create 3 books using the parameterized constructor.
  - Display the details of all books (use getters).
  - Borrow a book and attempt to borrow the same book again to test the logic.
  - Return a book and verify its status.