Topic 4-1

Classes and Objects

Previously...

- Real world problems contain a number of entities (similar or different) that interact with each other in interesting and specific ways.
- We introduced the idea of objects to capture the entities in the problem domain
    - objects have state: attributes and values
    - objects have behaviour: functionality
    - objects interact with each other

Today ...

- We will see how this happens in practice:
  - introduction to classes
  - class vs object
  - instantiation

Classes vs Objects

- A class is a user-defined data type whose variables are objects.
    - These objects can have both member variables and member functions.
    - An object is a runtime **instantiation** of a class

Encapsulation

Combining a number of items, such as variables and functions, into a single package, such as an object of some class, is called encapsulation.

OOP Design

Encapsulation

Objects

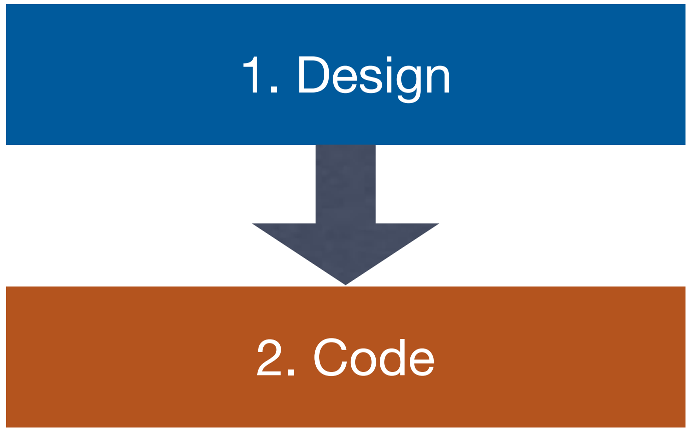
object hierarchy

data structure

responsibilities

...

interaction



program testing

debugging

OOP Design 101

- We start from a use case description (e.g. a paragraph of text).
- Then
    - Identify nouns as potential objects and their data members.
    - Identify verbs as interactions between these objects.
    - Identify the hierarchical relationship between the objects.
    - Identify the responsibilities for each object (who does what).

Use Case

Write a program to manage the student's records in a classroom. Each student has a unique ID, name and grade. The program should be able to print all the students in the classroom, find the grade of a student based on their ID and update the grade of a student. The program should write the students' records to a file and also read the records from a file.

testing functions classes

program

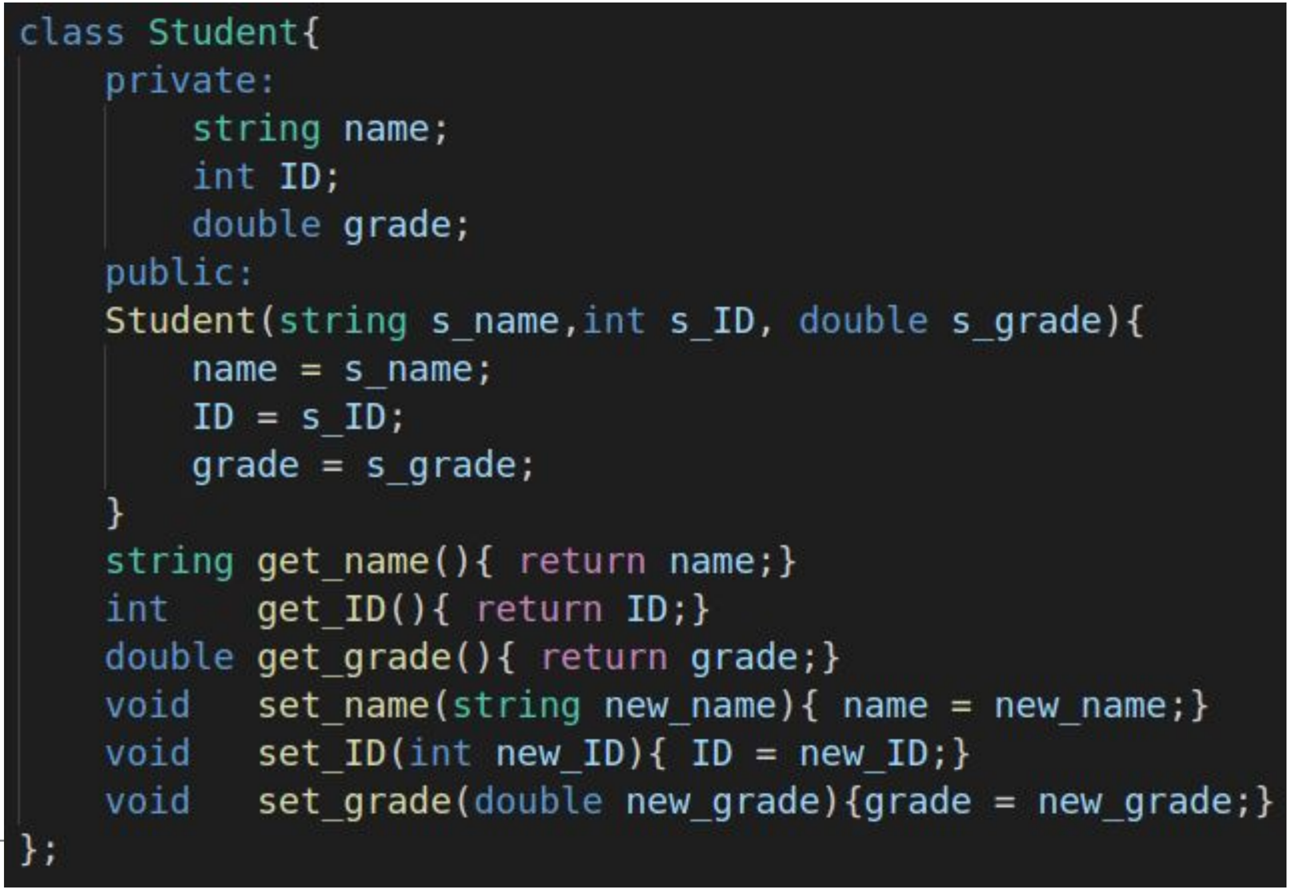
2. Code

debugging

- Step 1. From the design, identify the classes
  - class: Studnet; class: Classroom
- Step 2. Define classes in code
  - (generic) state and behaviour
- Step 3. Instantiate objects from classes
  - instantiate state
- Step 4. Invoke behaviour

The “Student” Class

- state:
  - ID,
  - Name
  - Grade
- Behaviour:
  - set/get name
  - set/get ID
  - set/get Grade



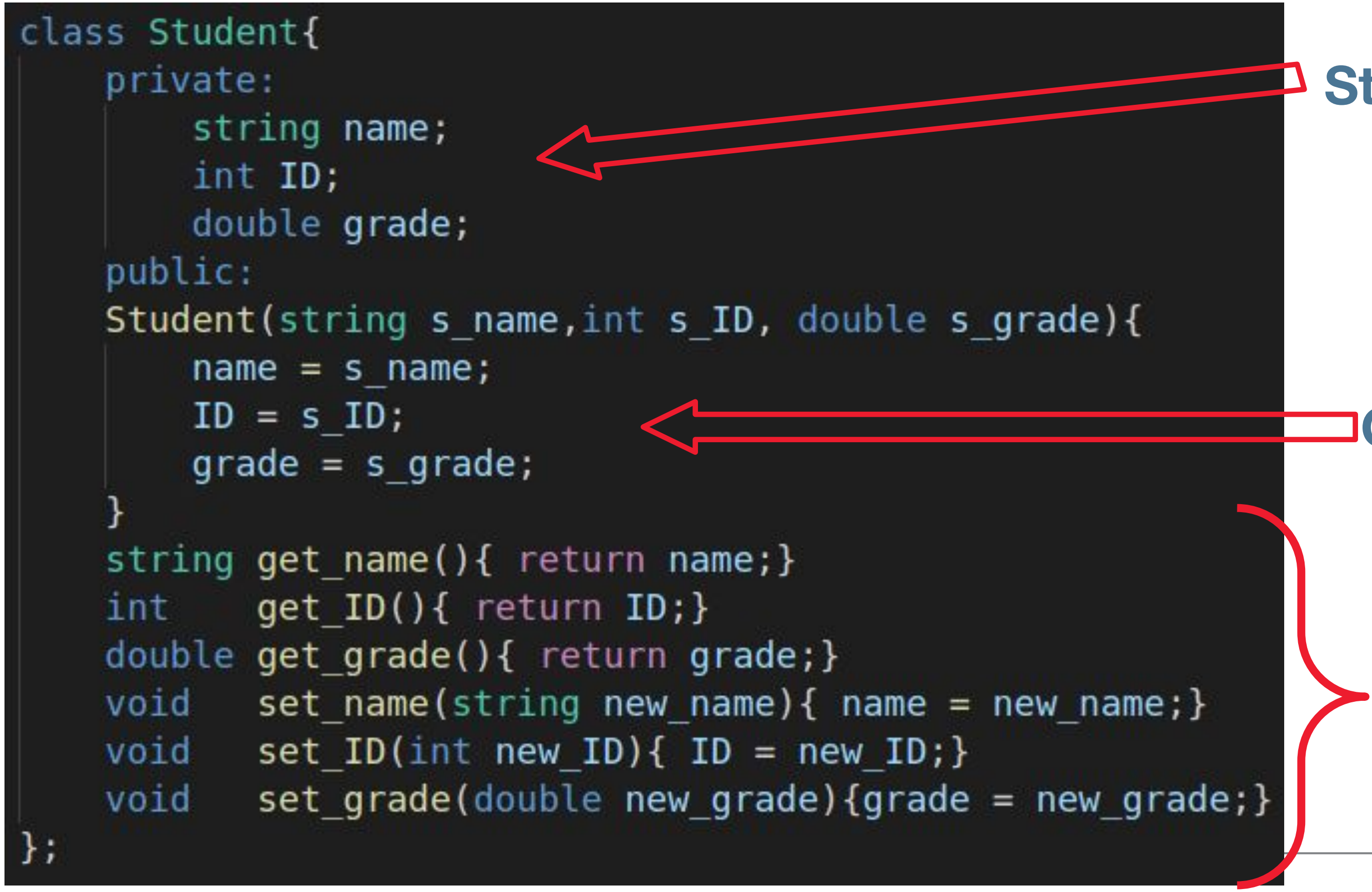
```
class Student{
    private:
        string name;
        int ID;
        double grade;
    public:
    Student(string s_name,int s_ID, double s_grade){
        name = s_name;
        ID = s_ID;
        grade = s_grade;
    }
    string get_name(){ return name;}
    int    get_ID(){ return ID;}
    double get_grade(){ return grade;}
    void   set_name(string new_name){ name = new_name;}
    void   set_ID(int new_ID){ ID = new_ID;}
    void   set_grade(double new_grade){grade = new_grade;}
};
```

Demo 1

Private Members

- If you insert the keyword private and a colon in the list of member variables and member functions, all the members that follow the label **private:** will be private members and functions.
- Private members: means that they cannot be directly accessed in the program except within the definition of a member function.

Step 2. Define classes in code



```
class Student{
    private:
        string name;
        int ID;
        double grade;
    public:
    Student(string s_name,int s_ID, double s_grade){
        name = s_name;
        ID = s_ID;
        grade = s_grade;
    }
    string get_name(){ return name;}
    int    get_ID(){ return ID;}
    double get_grade(){ return grade;}
    void   set_name(string new_name){ name = new_name;}
    void   set_ID(int new_ID){ ID = new_ID;}
    void   set_grade(double new_grade){grade = new_grade;}
};
```

State variables

Constructors

Behaviour

Constructors

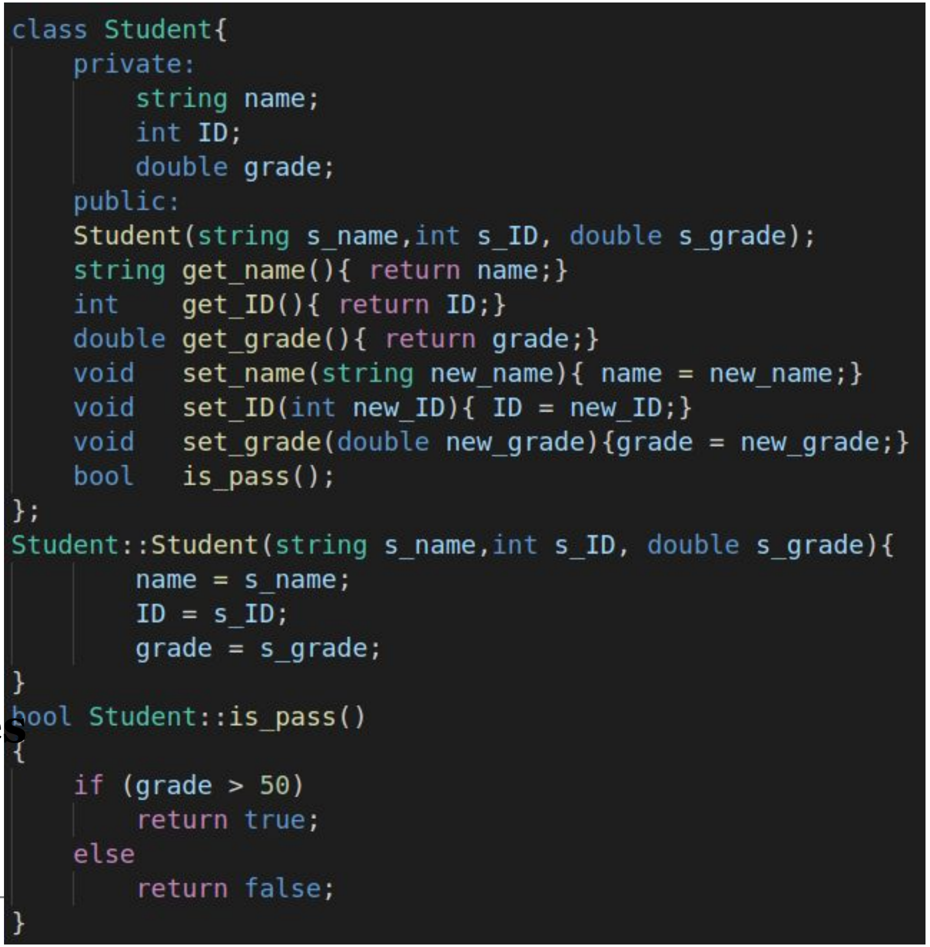
A constructor is a member function of a class that has the same name as the class. A constructor is called automatically when an object of the class is declared. Constructors are used to initialize objects. A constructor must have the same name as the class of which it is a member.

If you give no constructor, the compiler will generate a default constructor that does nothing. This constructor will be called if class objects are declared.

Member Function Definition

Inside the class for simple function members

Separately for the more complex ones



```
class Student{
    private:
        string name;
        int ID;
        double grade;
    public:
    Student(string s_name,int s_ID, double s_grade);
    string get_name(){ return name;}
    int    get_ID(){ return ID;}
    double get_grade(){ return grade;}
    void   set_name(string new_name){ name = new_name;}
    void   set_ID(int new_ID){ ID = new_ID;}
    void   set_grade(double new_grade){grade = new_grade;}
    bool   is_pass();
};
Student::Student(string s_name,int s_ID, double s_grade){
        name = s_name;
        ID = s_ID;
        grade = s_grade;
}
bool Student::is_pass()
{
    if (grade > 50)
        return true;
    else
        return false;
}
```

Summary

- Objects are programming constructs that abstract entities from the problem domain
    - they have state: attributes and their values
    - they have behaviour: what they can do
- Classes define the behaviour of objects; classes have generic states and are ***instantiated*** by objects