



UNIVERSITY OF
LIVERPOOL

Application Development with C++ (ELEC362)

Lecture 6: Classes & Object Oriented
Programming

mihasan@liverpool.ac.uk

Previous lecture

- The definition of pointers, their use and their advantages were discussed.
- Differences between two types of memory were discussed.
- Functions definitions and declarations were discussed.
- Functions overloading and templates were discussed .
- Exceptions and error handling were discussed.

This lecture

- What is covered in this lecture?
 1. Object Oriented Programming (OOP).
 2. Introduction to classes.
- Why it is covered?
 1. Classes are the building block of any software.
 2. OOP is the standard paradigm in software development.
- How are topics covered in this lecture:

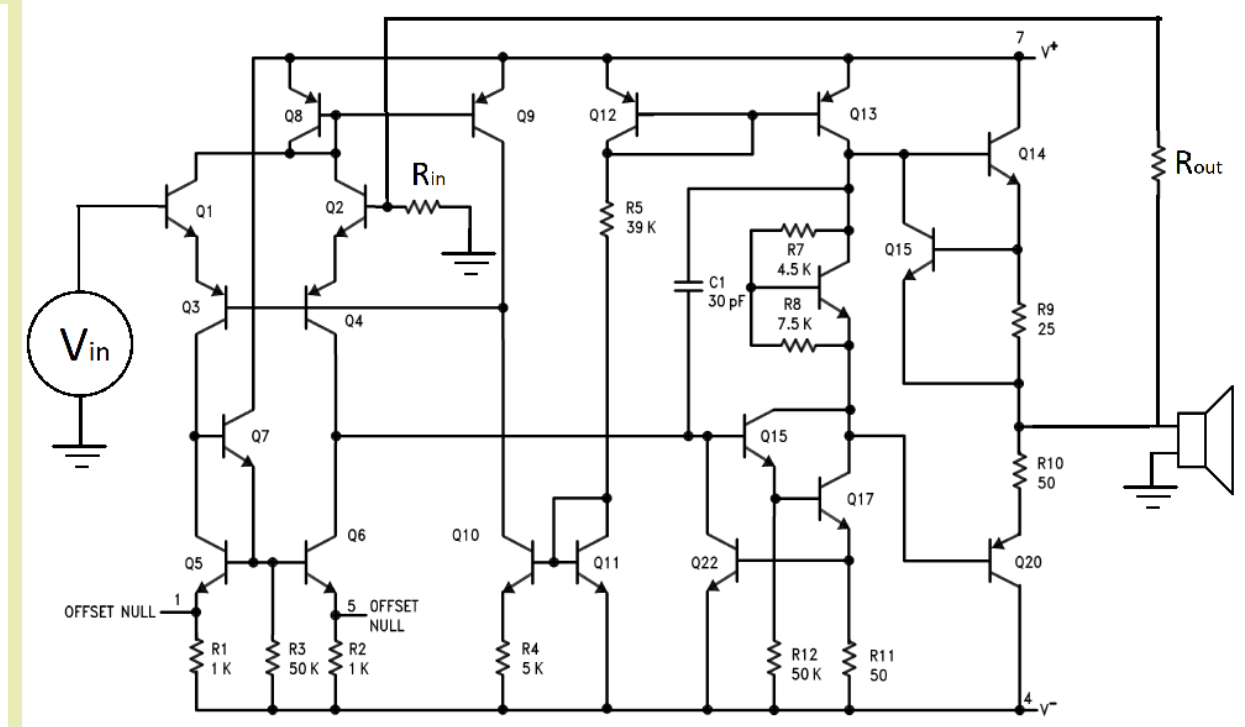
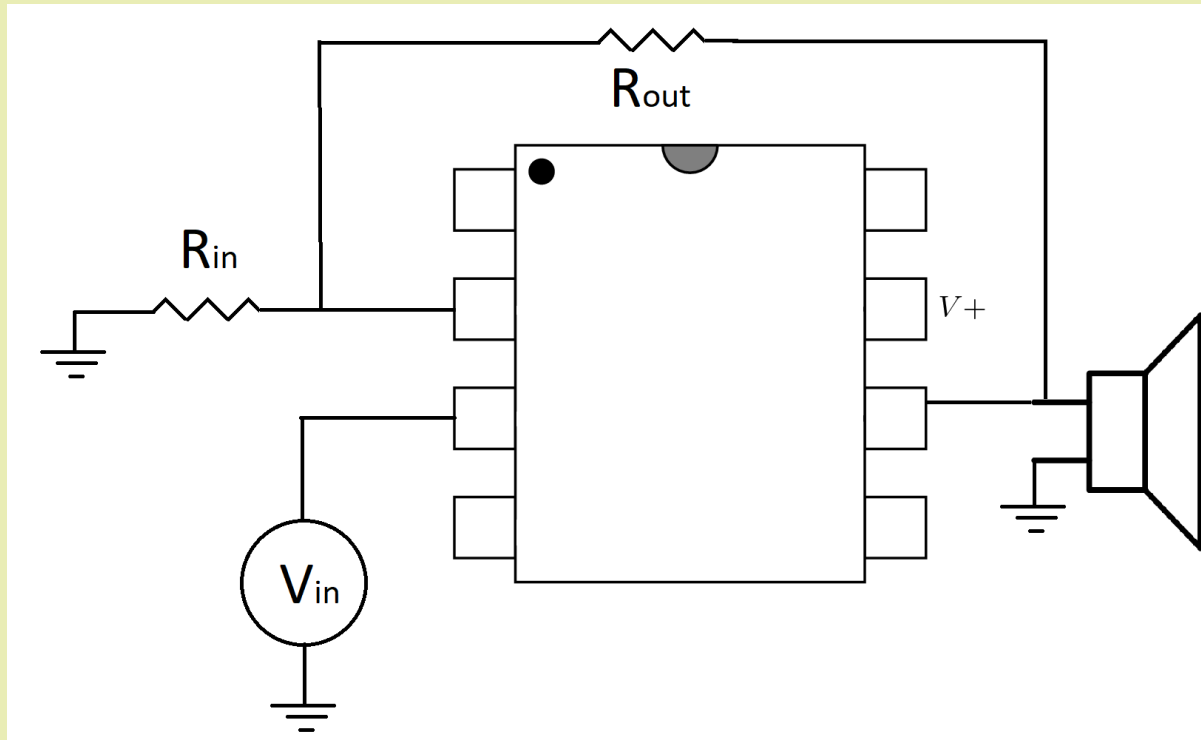
A side story, 2 source codes, and a practical example.

Side story

Side story:
Integrated Circuits
(IC) in Electronics

ICs versus basic components

- If you are asked to build an amplifier circuit, which one will you build?



- Do you prefer using Arduino/Raspberry Pi or basic components in your FYP?

ICs versus basic components

- The vast majority of electronics engineers prefer using ICs for many reasons including:
 1. It is easier: Fewer design decisions to make.
 2. It saves a lot of time: No need to repeat what others have built.
 3. It gives a consistent performance: Same performance for same IC.
 4. It is cheaper: Saving production costs and time.
- Since their invention in 1958, ICs revolutionised technology, leading its inventor to win a Nobel Prize in Physics in 2000
(<https://www.nobelprize.org/prizes/physics/2000/summary/>)

Side story

End of the side
story

Object Oriented Programming

- Some computer scientists in the 60s were like:
- Object Oriented Programming (OOP)
revolutionised software just like ICs
revolutionised all electronics.
- The core idea is: Instead of building codes using basic variables, they are built using higher-level variables know as “objects”.



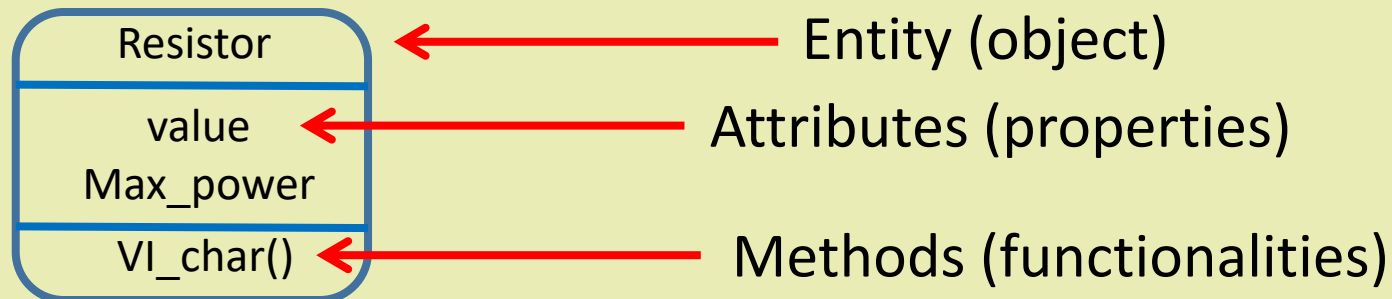
Object Oriented Programming

- A conceptual comparison between ICs and OOP:

Aspect	Modern Electronics	Modern software
Building block	Integrated circuits	Objects
What is the building block made from	Made from basic components such as resistors, transistors, capacitors, etc.	Made from basic variables datatypes and control structures such as double, integer, bool, and for loops
Requirements for using building block	Having the datasheet	Having the documentation

Object Oriented Programming

- Object Oriented Programming (OOP) is a programming paradigm where the programming problem is defined as a collection of interacting objects describing real-world entities.
- Abstract concept of the object in software such as MultiSim:



Object Oriented Programming

- Another example:

Object	Properties	functionality
Mario	Number of lives, collected coins	Fire bullets, flying, sliding
Turtle	Points gained when beat	Range of motion, range of fly
Level	Time, terrane, map	??
Mystery box	Design, points hidden	Fire flower release, Mushroom motion

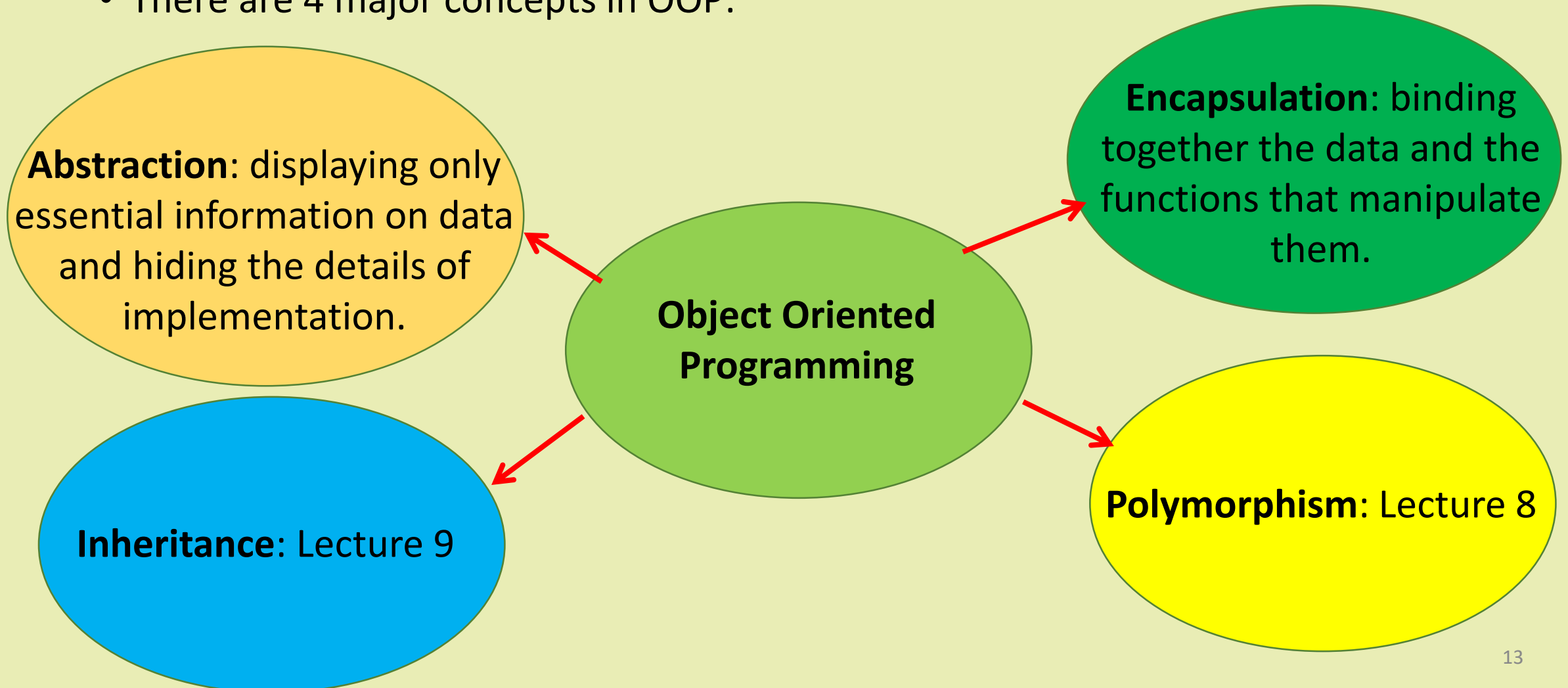


Object Oriented Software Design

- When designing a software based on OOP, the following steps has to be followed:
 1. Given the specification, Identify different entities of the software.
 2. Define the relationship between these entities.
 3. Define the attributes of these entities.
 4. Defined methods/functions of these entities.

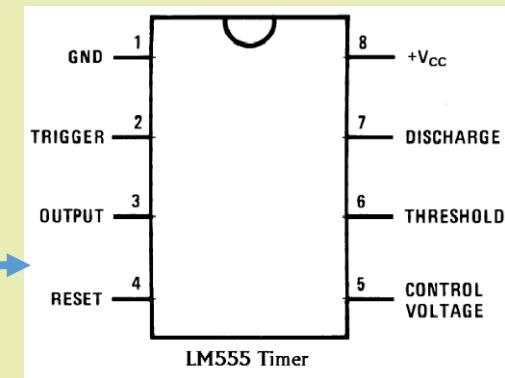
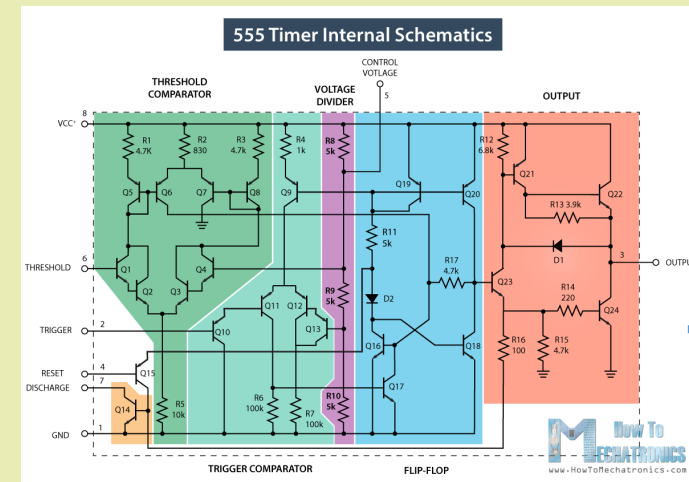
Object Oriented concepts

- There are 4 major concepts in OOP:



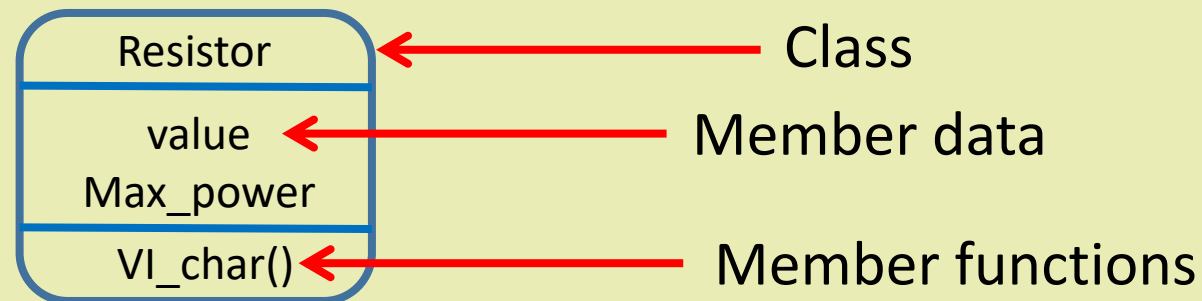
Encapsulation and Abstraction

- Encapsulation means hiding the details of the implementation inside the object.
- Abstraction is the concept of dealing with the object as a “black box”.
- Abstraction is a result of Encapsulation.
- A higher level of encapsulation leads to better abstraction, which leads to more efficient code development / user friendly software.
- Practical example: [MOOSE](#) vs [DEAL II](#)



Classes

- A class is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
- Code implementation of an example object in MultiSim:



- An instant of a particular class is called an Object.

Classes

- Syntax:

```
class ClassName {  
    public: ← Access specifier  
    //member data and functions  
    private:  
    //member data and functions  
    protected:  
    //member data and functions  
} ;
```

- Google style for naming classes is similar to that of functions.
- Go to [L6D1.cpp](#)

Access specifiers

- The class must be defined before it is used in the main function:

```
class MyClass{  
    //Class definition here;  
}  
  
int main () {  
    //Statements;  
}
```

- Access specifiers control access to class members from outside the class.
 1. **public**: allows access from outside the class.
 2. **private**: gives access only within the class.
 3. **protected**: gives access within the class and to derived classes (Lecture 9).

Class vs Object revisited


- The class is a user-defined datatype, whereas the object is an entity (or a variable) of that datatype.

Class

```
class Triangle {  
  
private:  
    double a,b,c,s;    // member data  
  
public:  
    // Methods or member functions  
    void SetSides(double x,double y, double z){};  
    double CalcPerimeter(){};  
    double CalcArea(){};  
};
```

Object

```
Triangle t;  
  
double x = 1, y = 2, z = 3;  
  
t.SetSides(x,y,z);
```



This is how to call a member function to operate on an object

Structure

- Structures are the historic origin of classes.
- Syntax:

```
struct MyStruct {  
  
    // member data  
  
    // Methods or member functions  
  
};
```

- The major difference between a class and a structure is that all member data/functions in a class are private by default, while in a structure they are public.
- Go to [L6D2.cpp](#)

Practical note: Use structures only for data saving.

Summary

- The definition of Object Oriented Programming and its 4 major components were discussed.
- Defining classes and objects in a code were discussed.
- The role of access specifiers was discussed.
- Structures were discussed and compared to classes.