Lecture 4

Objects & Data Abstraction

Topics covered by last lecture

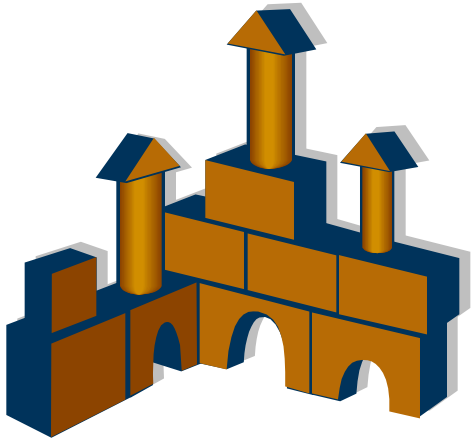
- Array declaration and initialization
- Common process in arrays:
  - linear search,
  - find the largest/smallest,
  - calculate average,
  - sorting,
  - …
- Array as function argument
- Multi-dimensional arrays

Topics covered by this lecture

- Objects
- Data abstraction
- Classes and objects
- Class definition
- Member functions
- Applications

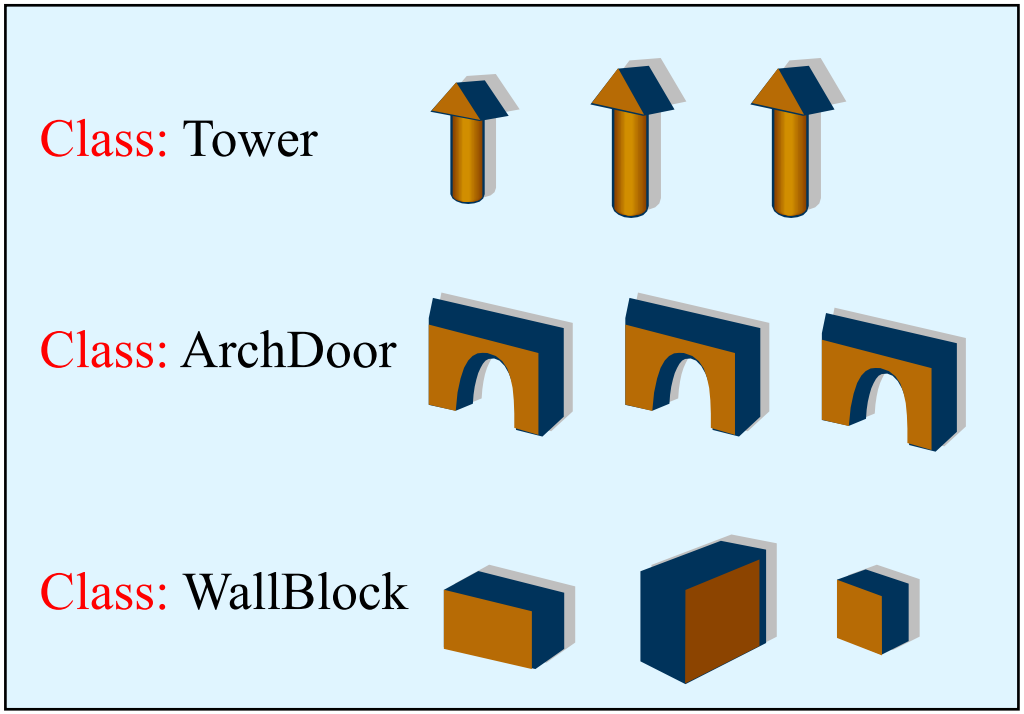
Object-Oriented Analysis and Design

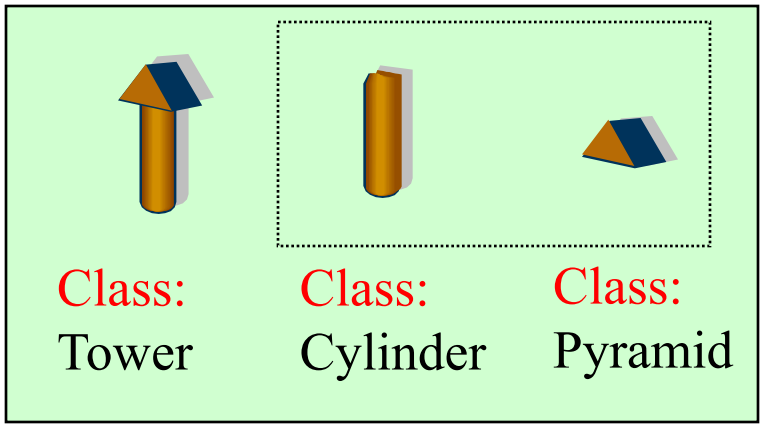
Procedural style

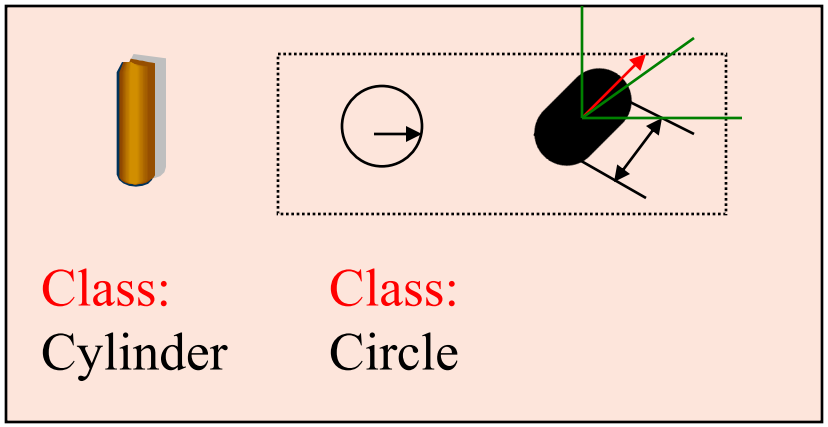


Class: Castle

OO style

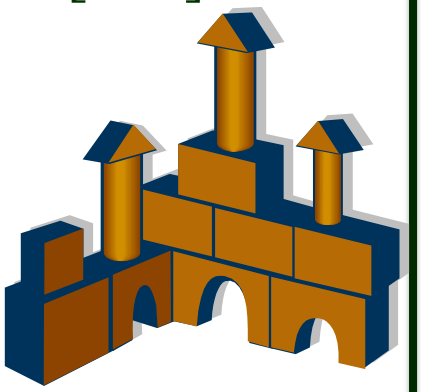






Object-Oriented Analysis and Design

- Objects
  - towers[3]
  - archDoors[3]
  - wallBloacks[200]
  - castle
- classes
  - Circle
  - Cylinder
  - Pyramid
  - Tower
  - ArchDoor
  - WallBlock
  - Castle



- Objects
  - board, players[2], game
- Classes
  - Board
  - Player: Random, Smart,…
  - Game

A preview of class implementation

```
class Date {
  public:
    int day;
    string month;
    int year;
};
```

```
struct Date {
    int day;
    string month;
    int year;
};
```

date.h

Class creator

an object of Date

```
int main() {
    Date d;
    d.day = 10;
    d.month = "Aug";
    d.year = 2021;
    cout << d.day <<" "<< d.month << "  "
          << d.year;
}
```

Alternative representation of date

```
int date[3];
date[0] = 10;
date[1] = 8;
date[2] = 2020;
```

Find the differences between C++ and Java

dateApp.cpp

Class client

A preview of class implementation

Class declaration

```
class Date {
  public:
    int day;
    string month;
    int year;
    void display();
};

void Date::display() {
    cout << day << "  "
         << month << "  "
         << year;

}
```

date1.h

Class application

```
int main() {
    Date d;
    d.day = 10;
    d.month = "Aug";
    d.year = 2021;
    d.display();
}
```

dateApp1.cpp

Class definition

Data Types

	Data Type	Definition	Declaration	Operators /methods	Use
Built-in	int	built-in	**int** i, j;	+ - * / =	i=0; j=2;
	float	built-in	**float** a, b;	+ - * / =	a=2.1; b=a*a;
	double	built-in	**double** x, y;	+ - * / =	x=3.1415*y;
Structure	Student	struct Student { string name; long studentID; } ;	Student a, b, c;	=	a.name = “dongmo”; b.studentID = 12345;

Class	Date	class Date
		{ int day;
		int month;
		int year;
		void display();
		};
			Date a, b, c;	=
				display()
					a.display();

Abstract Data Type

Abstract Data Type is a user-defined type that defines the types of data (data members) and the methods (member functions) that operate the data.

Integer type

data member:

int

operations:

+

-

*

/

Date type

data member:

int day;

string month;

int year;

methods:

void display();

Abstract Data Type

data members:

data item;

…

data item;

methods:

function;

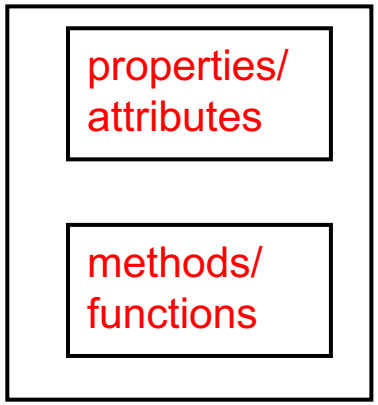
…

function;

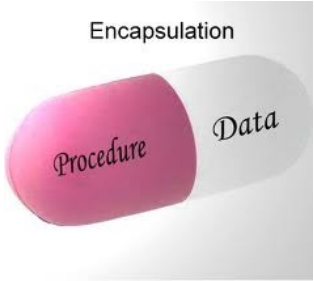
Abstraction, Encapsulation & Data Hiding

- *data abstraction*: generalisation of an object, focusing only on particular aspects of an object that are of specific interest to the problem.

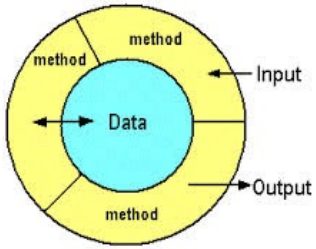
  e.g. `Student` – interested in *name*, *course*, *units* enrolled – not be interested in shoe size



- *encapsulation*: any object is encapsulated into a structure which consist of a set of *data items* (*attributes*) and a set of *methods* that operate the data.



- *data hiding*: ability to limit access to specific attributes and methods of an object. Avoid tightly "coupled" classes.



Class Declaration

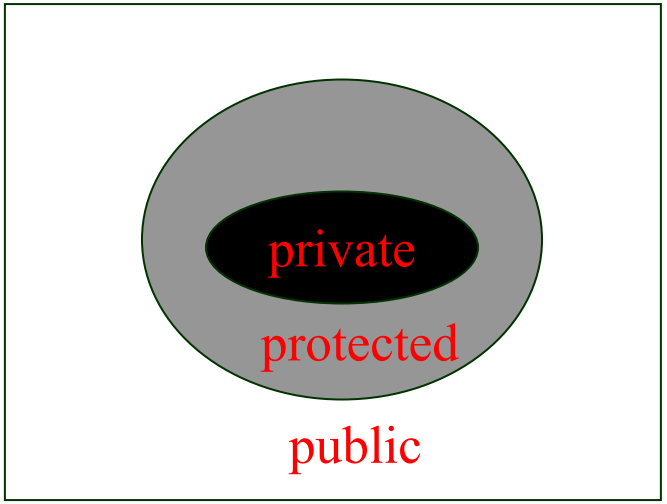
```
class  className {
  private:
      data items
      member functions
  protected:
      data items
      member functions
  public:
      data items
      member functions
};

member function definitions
```

access specifiers: valid until another type is defined

Be aware of the difference between C++ and Java

Class Declaration

- `access specifier` specifies access privileges.
  - **private :** accessible only to class methods
  - **public :** publicly accessible
  - **protected :** accessible from derived classes



A preview of class implementation

```
class Date {
private:
    int day;
    string month;
    int year;
public:
    void display();
};

void Date::display() {
    cout << day << "  "
         << month << "  "
         << year;
}
```

date3.h

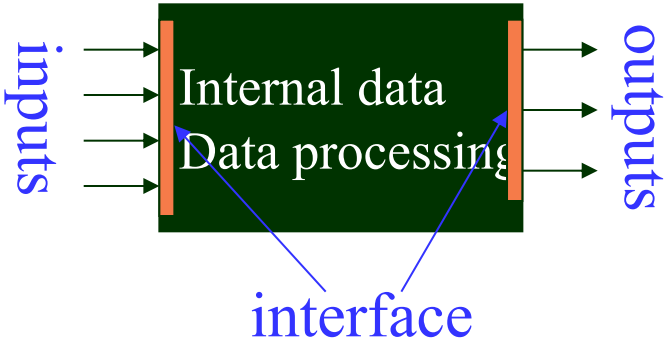
Cannot access private members declared in class 'Date'

```
int main() {
    Date d;
    d.day = 10;
    d.month = "Aug";
    d.year = 2021;
    d.display();
}
```

dateApp3.cpp

Interface of a class

- The interface of a class consists of all its public functions



- The interface of a class is normally used for input and output data
- Two typical interface functions:
  - **Accessor**: output data (`get` functions)
  - **Mutator**: takes input as parameters (`set` function)

date3.h

dateApp3.cpp

Data access

dateApp4.cpp

```
class Date {
  private:
     int day;
     string month;
     int year;
  public:
    void setDate(int,string,int);
    void display();
};

void Date::setDate(int d, string m, int y) {
    day = d;
    month = m;
    year = y;
}
```

date4.h

```
int main() {
    Date d;
    d.setDate(10,”Aug”,2021);
    d.display();
}
```

Internal data and processing

date2.h

dateApp2.cpp

```
class Date {
  private:
     int day;
     int month;
     int year;
     string mapping(int);
  public:
    void setDate(int,int,int);
    void display();
    void showDate();
};
```

Why do we set it to be private?

```
void Date::display() {
    cout << "The date is ";
    cout << day << " ";
    cout << mapping(month-1);
    cout << " " << year << endl;
}

void Date::showdate() {
    cout << "The date is ";
    cout << setfill('0')
                << setw(2) << day << '/'
                << setw(2) << month << '/'
                << setw(2) << year % 100
                << endl;
}
```

```
string Date::mapping(int month) {
    string map[12] ={"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
                     "Sep", "Oct", "Nov", "Dec"};
    return map[month];
}
```

Struct vs Class in C++

```
struct  className {
   private:
        data items
        member functions
   protected:
        data items
        member functions
   public:
        data items
        member functions
};
```

Be aware of the differences among C, C++ and Java

The difference between class and struct is that if leave access specifiers out, the default access for class is private while the default access for struct is public. There is no other difference between them.

Classes and Objects: a summary

A class contains:

a) data items – *attributes/properties/characteristics*,

b) methods – *member functions/behaviours/capabilities*.

An object is an instance of a class, which have:

a) *specified attributes, properties, characteristics, containing data.*

b) *abilities to use the member functions to operate the data of the object.*

Classes and Objects: a summary

```
void main() {
    Date d1, d2, d3;
    d1.setDate(20, 8 ,2021);
    d2.setDate(10,9,2021);
    d3.setDate(19,10,2021);

   d1.showdate("Today is ");
   d2.showdate("The due date for assignment 1 is ");
   d3.showdate("The due date for assignment 2 is ");
}
```

objects

date5.h

dateApp5.cpp

Be aware of the difference between C++ and Java

multipleClass.cpp

Principles of Object-Oriented Programming

- All data types are classes, including *int*, *double*, *bool*, *array*, *char*.
- All data values are objects (variables are objects).
- Any *non-built-in* object is made up of other objects.
- A class defines the data types of its data members and their behaviour.
- Computer programs are designed by making them out of objects that interact with one another.

From procedural to OO

```
int main() {
    cout << "Welcome to OOP class.";
    return 0;
}
```

Procedural style

ProceduralStyle.cpp

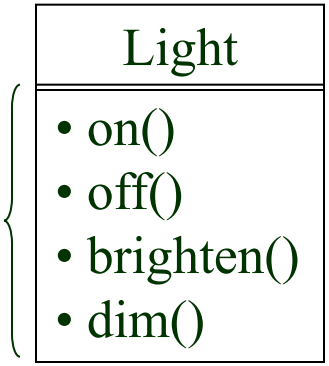
```
class Easy {
public:
    void run() {
       cout << "Welcome to OOP class.";
    }
};

int main() {
    Easy e;
    e.run();
    return 0;
}
```

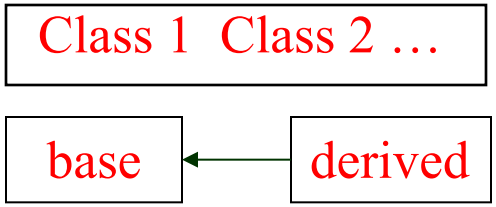
OO style

OOStyle.cpp

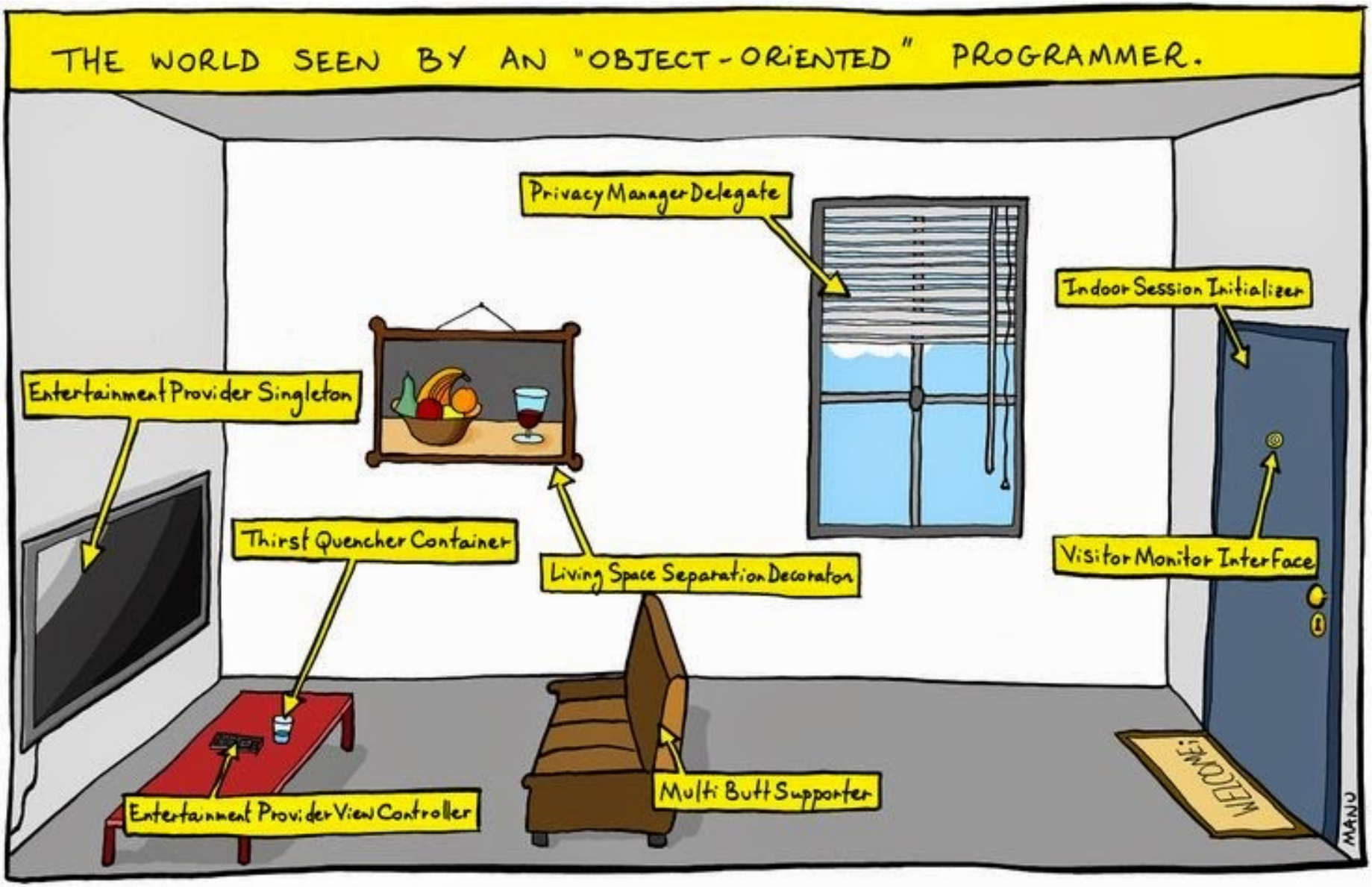
Class Implementation and Reusability

- Abstraction: a class is an abstract of the same type of objects
- Interface: public members of a class -- the entries to an object.
- Hidden implementation:
  - Private
  - Protected
- Reusing the implementation:
  - Composition
  - Inheritance
- Polymorphism





Abstraction



Homework

- Read textbook: Chapter 6
- Complete and attend practical 3 this week
- Complete the online tutorial 1 via vUWS. It will be open for one week. You can do it anywhere within the specified time period. Meanwhile practical 4 is expected to be completed within Weeks 5&6