بسم الله الرّحمٰن الرّحيم

CS217 OBJECT ORIENTED PROGRAMMING

Spring 2020

INSTRUCTOR

Muhammad Danish Khan

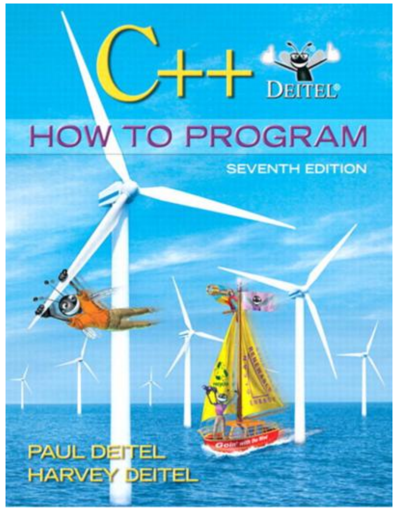
Lecturer, Department of Computer Science

M.Danish@nu.edu.pk

Room#16, Computer Science Block

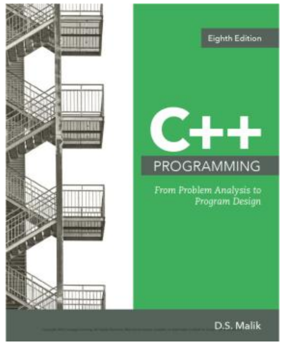
FAST NUCES (Karachi)

TEXT BOOK(S)

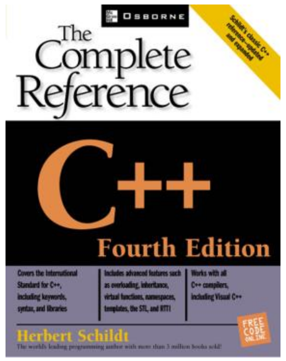


Recommended Text:

C++: How to Program
Paul Deitel
Harvey Deitel



C++ Programming
From Problem Analysis
to Program Design
D.S Malik



C++: The Complete Reference
Herbert Schildt

MARKS DISTRIBUTION

Mid Exam I	10%
Mid Exam II	10%
Quizzes (3 – 4)	10%
Assignment (>=10 Tasks)	05%
Semester Project	15 - 20%
Final	40 - 45%

TEACHING PLAN

Mid I

- Introduction to OO paradigm, Comparison from sequential & procedural paradigms, Data Abstraction, Encapsulation, Introduction to Objects in real world, Introduction to classes and objects, Access Control, Constructors, Destructors, Implicit and Explicit Casting, Member initialization list & constants, Static data and member functions, Inline functions
- Quiz I
- Tasks

Mid II:

- Semester Projects' Proposals
- Inheritance and its Types, Data and Code Hiding, Polymorphism: Function Overloading and Function Overriding, Friend Functions, Operator Overriding, Multiple Inheritance, Virtual Inheritance, Virtual Functions, Abstract Class, Filing,
- Quiz II
- Tasks
- Semester Project (Evaluation 1)

Finals

- Generics, Exception Handling, Intro. To C#, Properties in C#, GUI, Linking Window Form, Filing in C#, Exception Handling in C#
- Quiz III
- Tasks
- Semester Assignment (All Tasks)
- Semester Projects (Evaluation 2 + VIVA + Report)

COMPUTERS TODAY

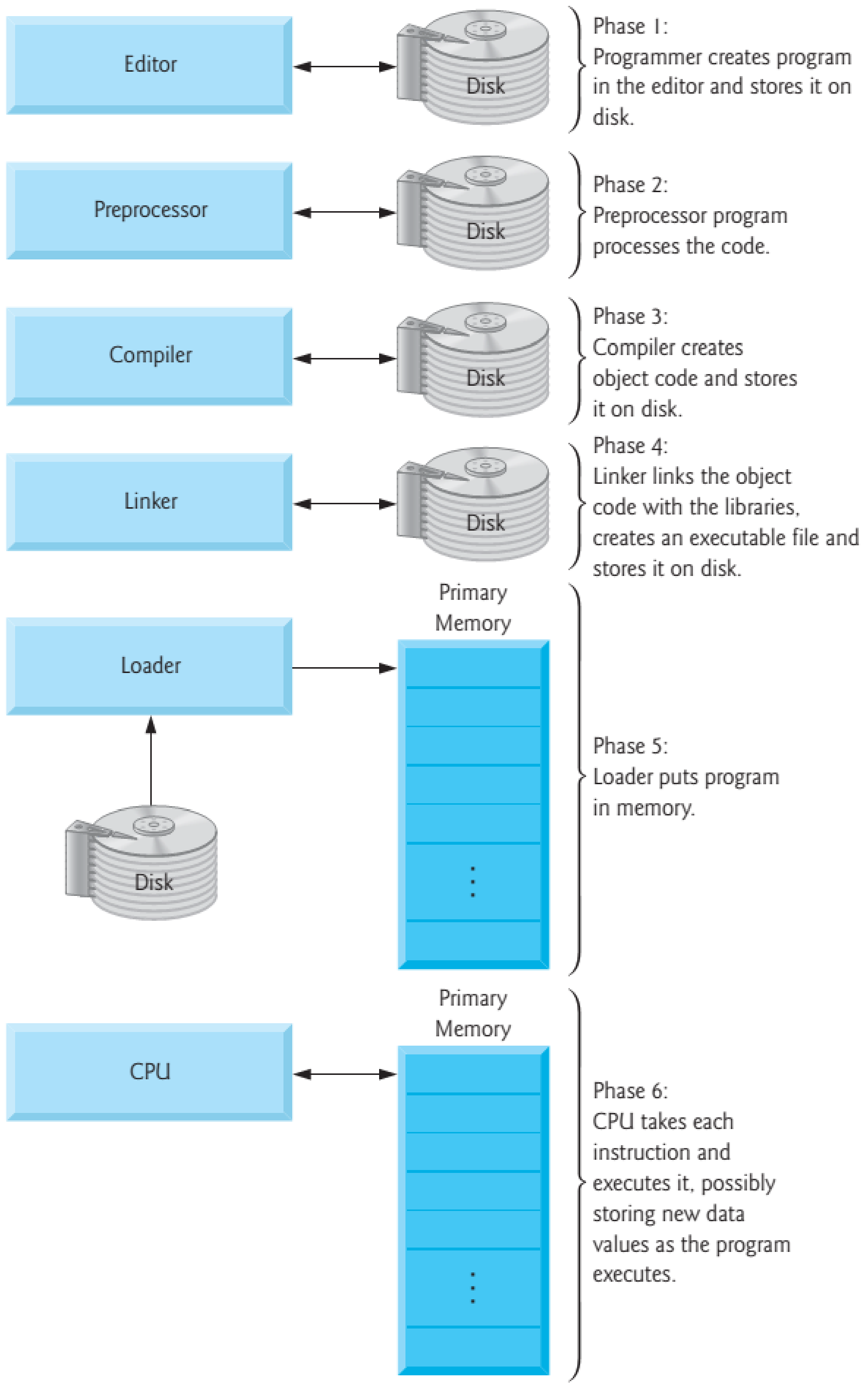
- Super Computers
- Mainframes
- Spreadsheets
- Microcomputers
- Embedded Computers
- ….

All controlled by **Computer Programs** (System SW, Application SW, Firmwares).

- **Computer Organization**: I/O Units, Memory Unit (Primary Memory), Microprocessor/CPU (ALU)

- **Languages**: Machine Language, Assembly Language, High-Level Language**.**

- **Computer Program**: Sequence of instructions ***compiled***, ***assembled***, ***linked***, and ***loaded*** for ***execution***.
- Programs are designed on some **algorithm**.
-


Editor
Disk
Phase 1:
Programmer creates program in the editor and stores it on disk.
Preprocessor
Disk
Phase 2:
Preprocessor program processes the code.
Compiler
Disk
Phase 3:
Compiler creates object code and stores it on disk.
Linker
Disk
Phase 4:
Linker links the object code with the libraries, creates an executable file and stores it on disk.
Primary Memory
Loader
Disk
Phase 5:
Loader puts program in memory.
Primary Memory
CPU
Phase 6:
CPU takes each instruction and executes it, possibly storing new data values as the program executes.

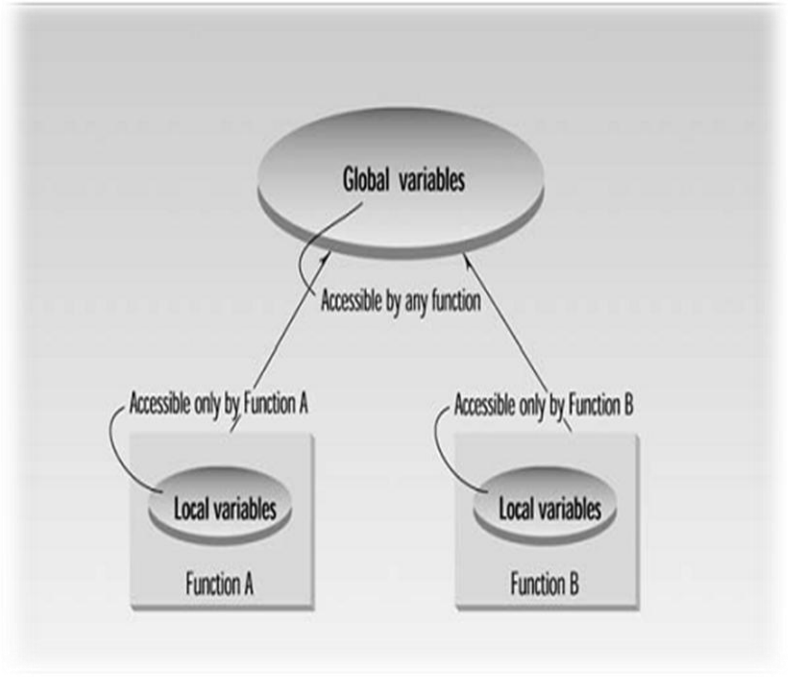
- Why do we need Object Oriented Programming (OOP)?

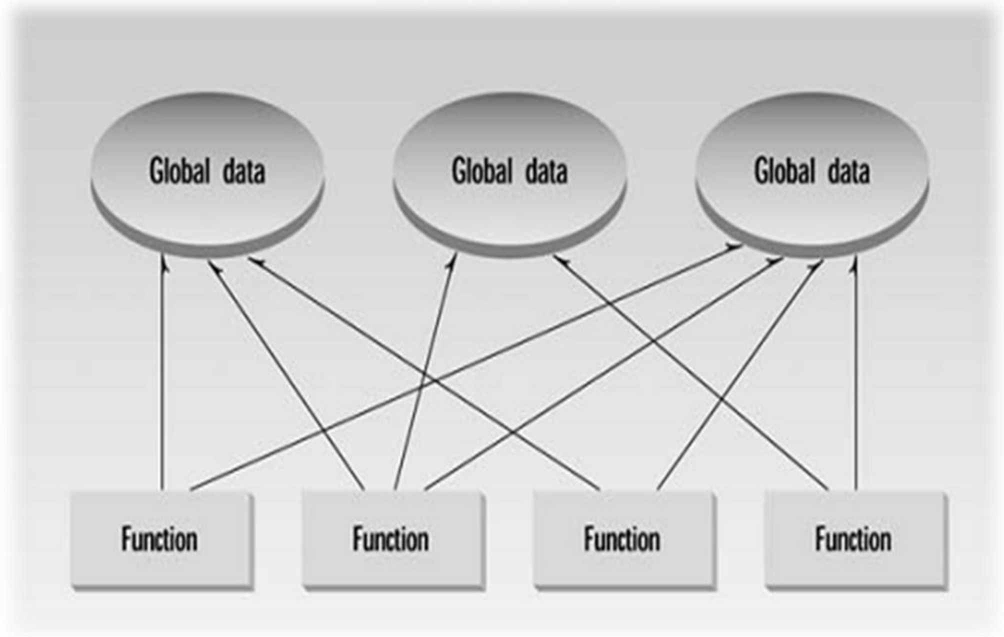
- Characteristics of Object Oriented Language

WHY OOP?

- Procedural Languages
  - Each statement in the procedural language tells the computer to do something, that is a list of instructions. No other *paradigm* is needed.

- Unrestricted Access
  - **Local Data**: Data Accessible to its originating functions only.
  - **Global Data**: making the data global leads to an even larger number of potential connections between functions and data


Global variables
Accessible by any function
Accessible only by Function A
Accessible only by Function B
Local variables
Function A
Local variables
Function B

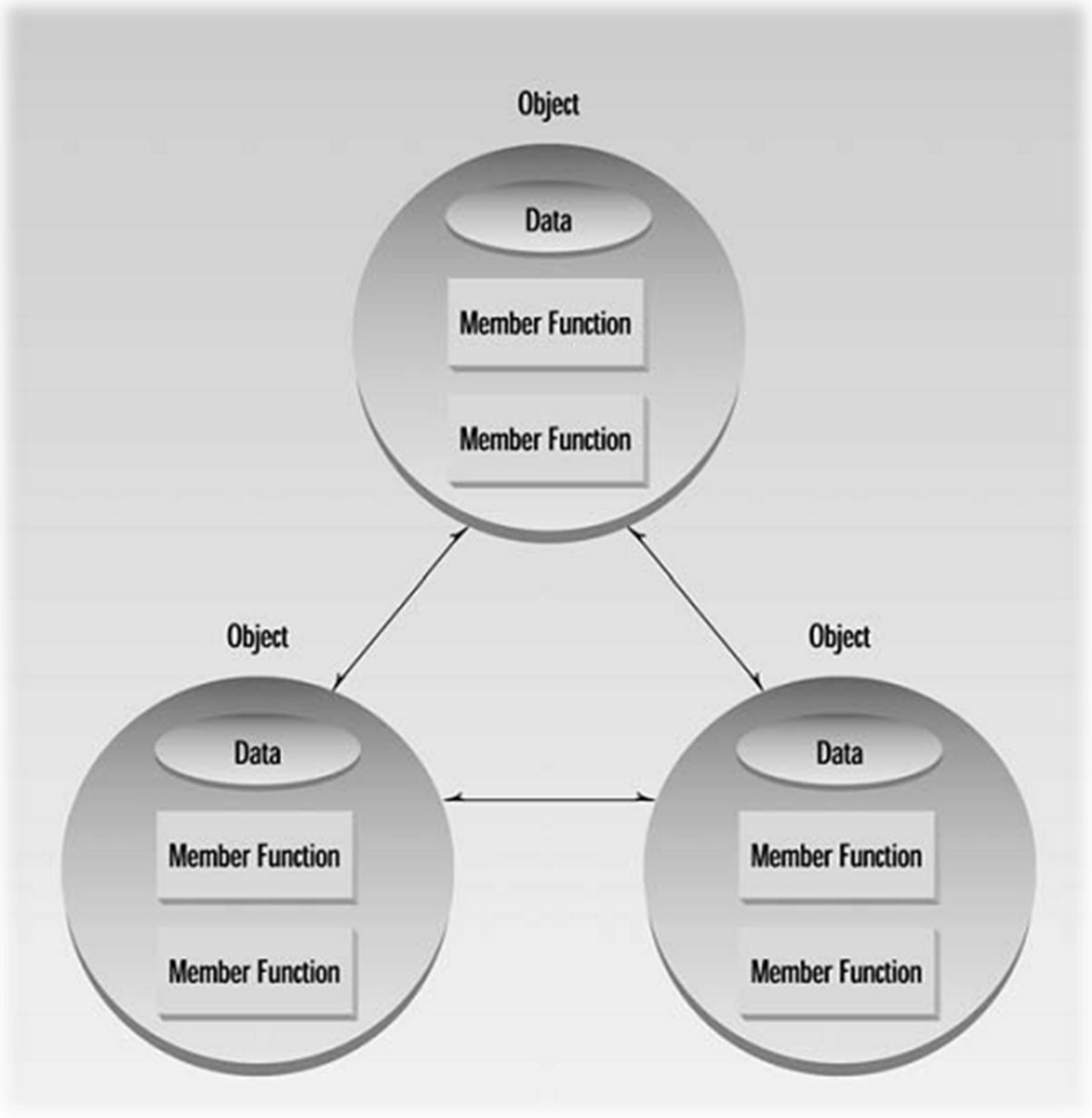

Global data
Global data
Global data
Function
Function
Function
Function

THE OBJECT ORIENTED APPROACH

"combine into a single unit both <u>data</u> and the <u>functions</u> that operate on that data"

- Such a unit is called an *<u>Object</u>*.
- An object's functions, called <u>member functions</u>, typically provide the only way to access its data.
  - If you want to read a data item in an object, you call a member function in the object.
  - You can't access the data directly.
- The data is <u>hidden</u>, so it is safe from accidental alteration. Data and its functions are said to be <u>encapsulated</u> into a single entity.

THE OBJECT ORIENTED PARADIGM



CHARACTERISTICS OF OBJECT-ORIENTED LANGUAGES

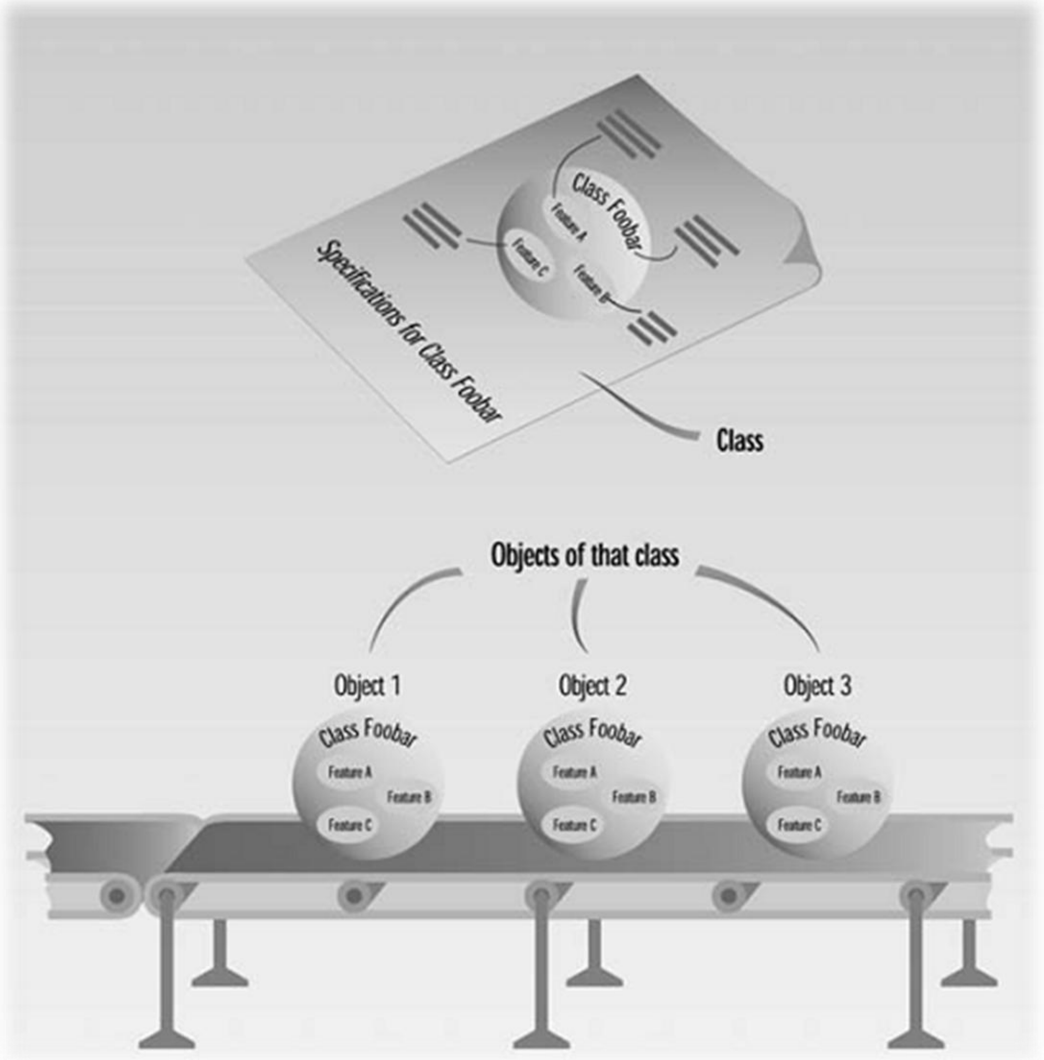
- **Classes**
- **Objects**
- **Encapsulation**
- **Inheritance**
- **Polymorphism and Overloading**

CLASSES AND OBJECTS

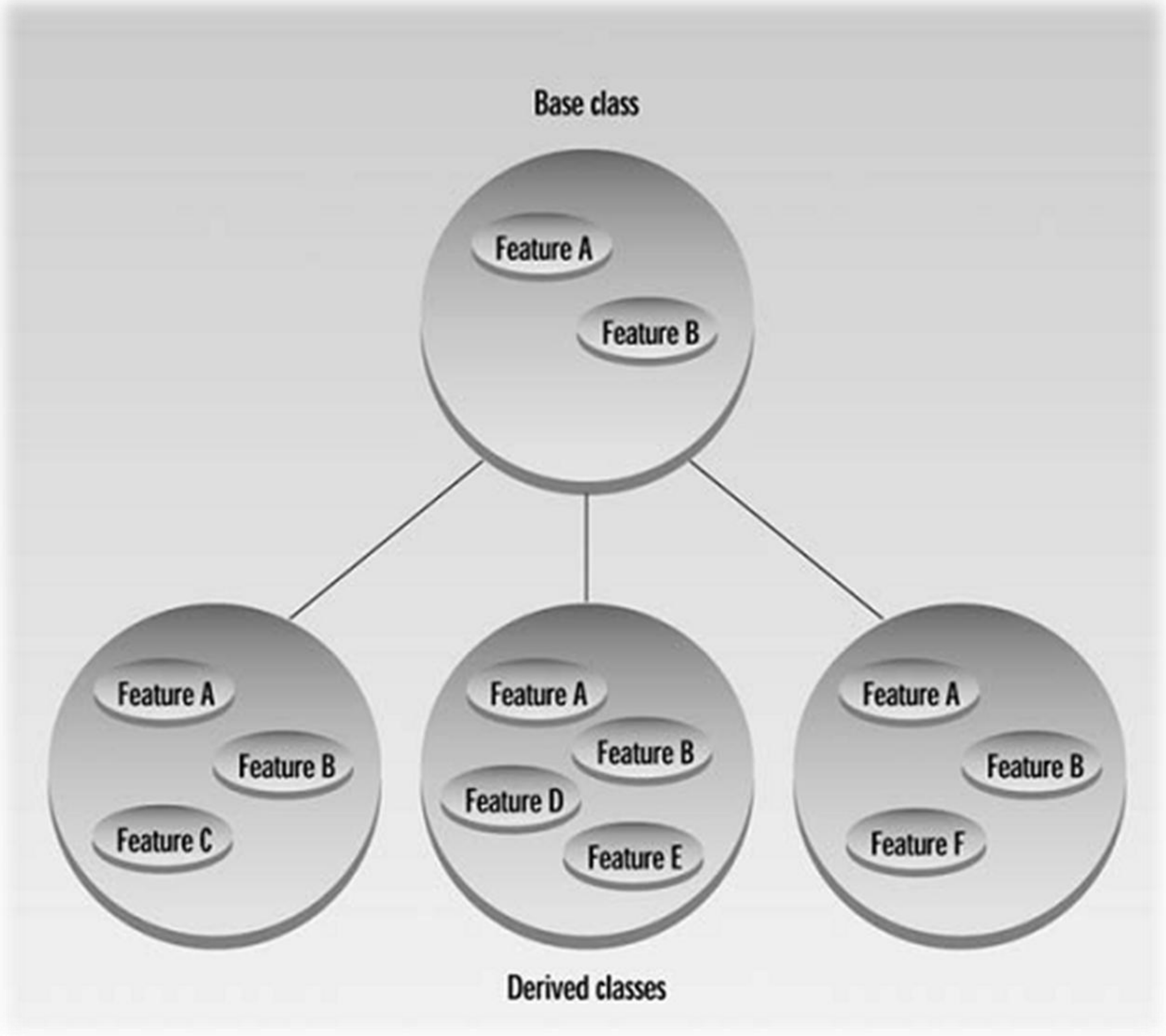
- Recall:
  - **Structures**, which provide a way to group data elements.

  - **Functions**, which organize program actions into named entities.

A *CLASS* AND ITS *OBJECTS*



INHERITANCE

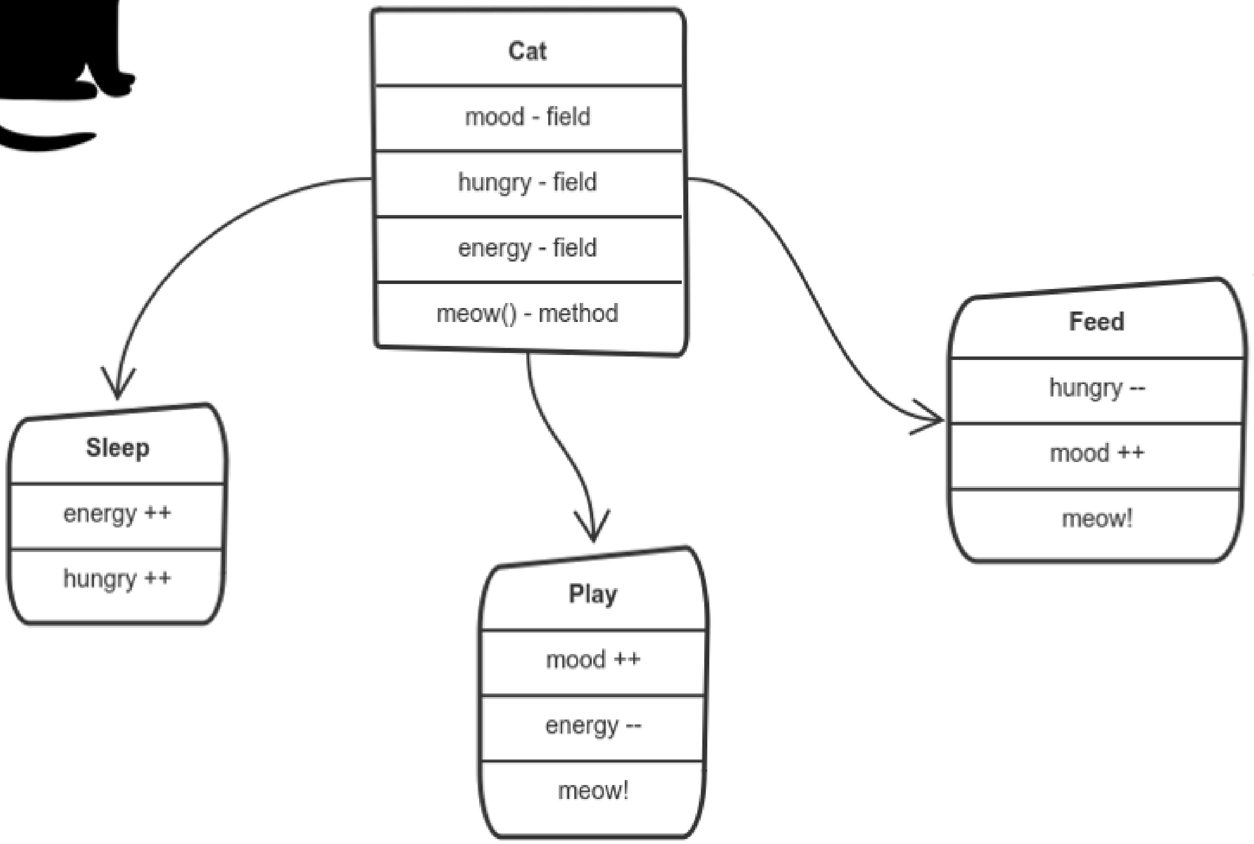


WHAT IS OBJECT ORIENTATION????

WHAT IS AN OBJECT?

- An object is:
  - Anything for which we want to save information
  - Something tangible (e.g. Cat, Car)
  - Something that can be captured intellectually

- An object has:
  - State/Attribute/Properties/Data
  - Well-defined Behavior/Methods/Functions

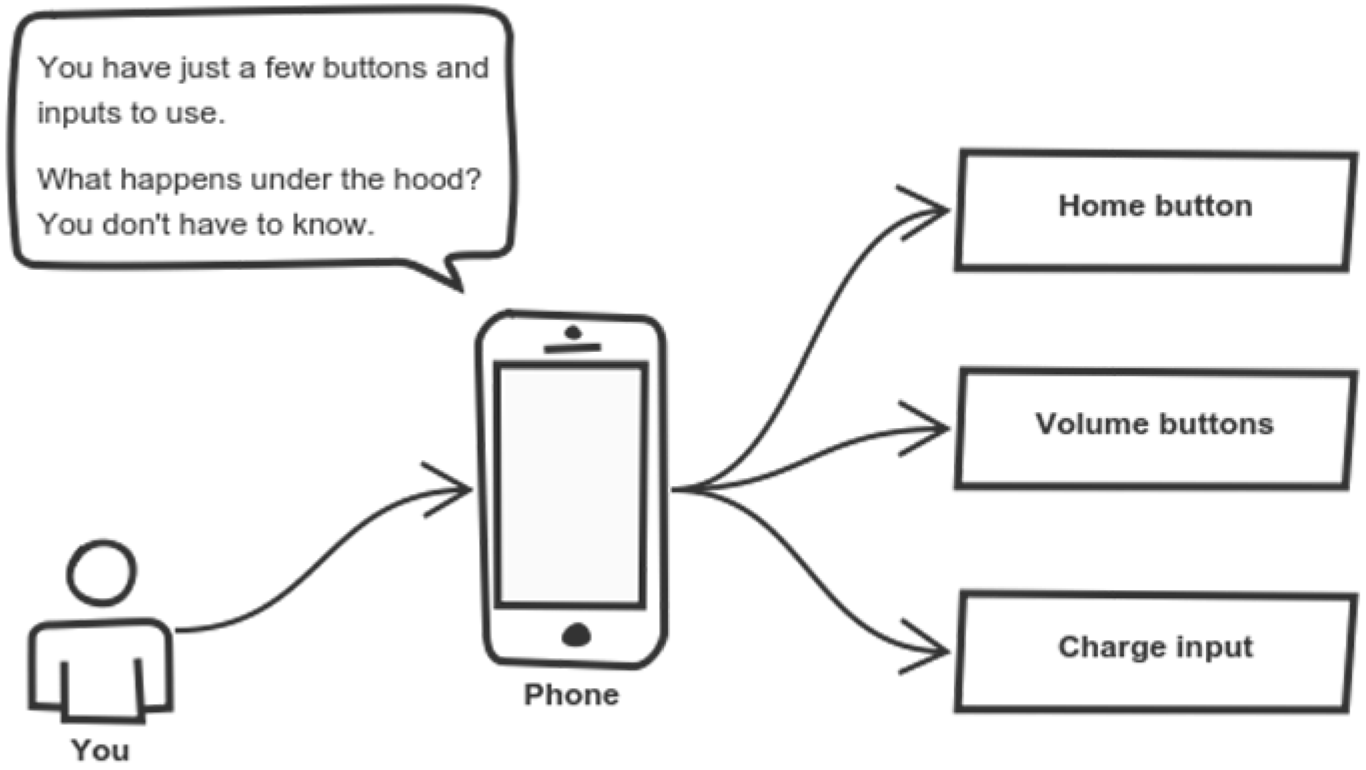

Cat
mood - field
hungry - field
energy - field
meow() - method
Sleep
energy ++
hungry ++
Play
mood ++
energy --
meow!
Feed
hungry --
mood ++
meow!

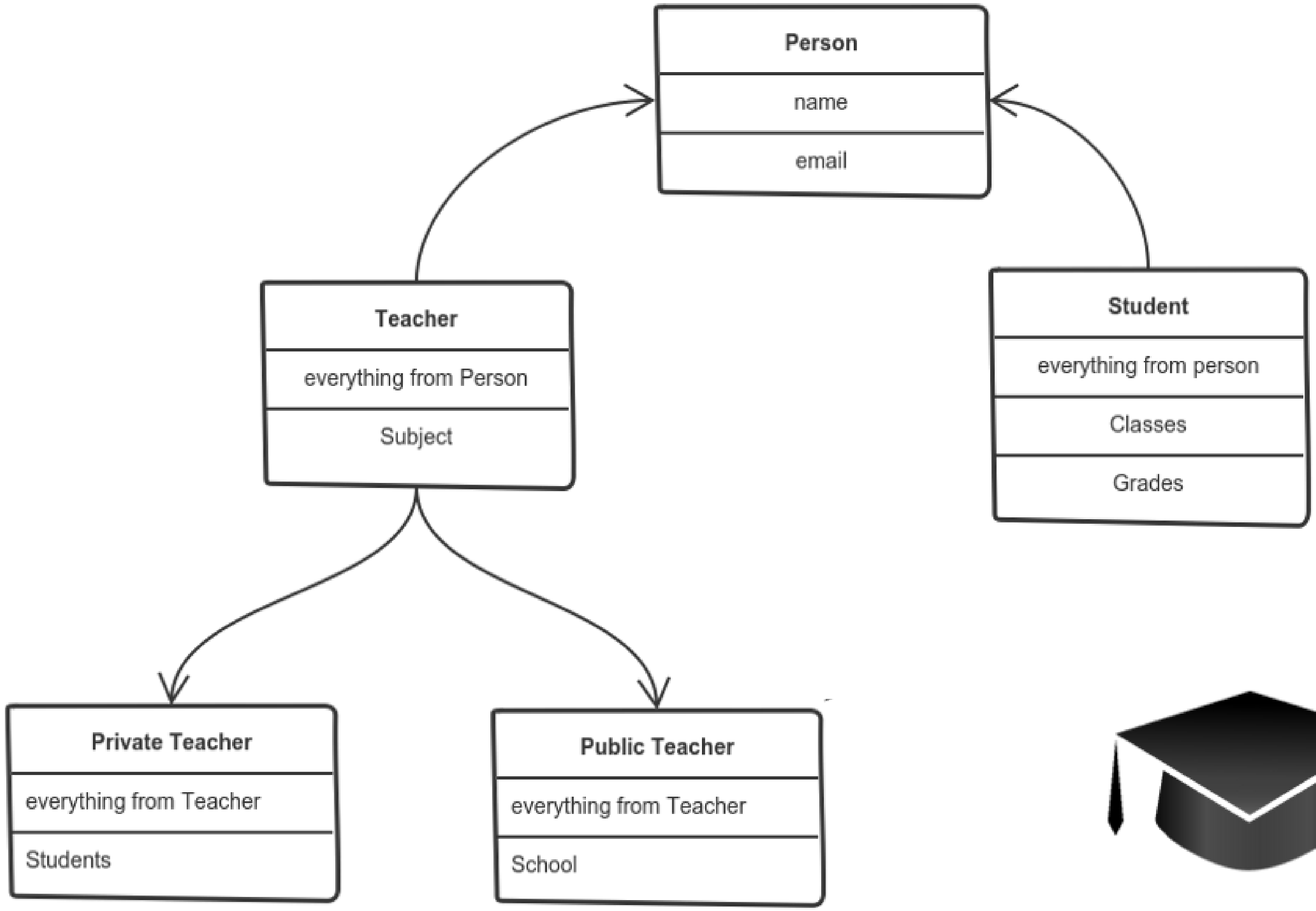
EXERCISE

- Are there any ‘objects’ in this classroom?

EXERCISE

- Are there any ‘objects’ in this classroom
  - **Chair**
  - **Multimedia**
  - **Student**
  - **Teacher**


You have just a few buttons and inputs to use.
What happens under the hood? You don't have to know.
Home button
Volume buttons
Charge input
Phone
You


Person
name
email
Teacher
everything from Person
Subject
Student
everything from person
Classes
Grades
Private Teacher
everything from Teacher
Students
Public Teacher
everything from Teacher
School

EXERCISE

- Possible Objects in this university?

INFORMATION HIDING

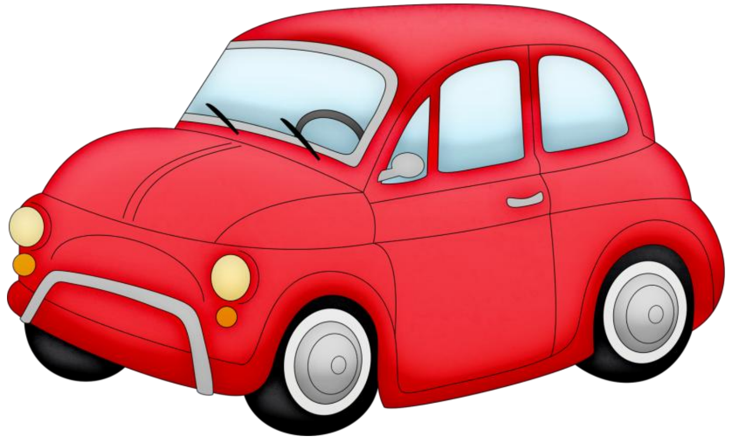
- Implementation details are hidden from the outside world
- Information is stored within the object
- It can only be *changed* by the object itself

ADVANTAGES OF INFORMATION HIDING

- Simplifies the model by hiding implementation details
- Prevents accidental access
- Prevents invalid access or manipulation

CLASSES

- Accelerator Pedal
- Brake Pedal
- Steering Wheel



- User-friendly “interfaces” to control the car
- Their mechanism is *housed* inside the engineering drawings or blueprints

CLASSES

```
class Car
{
        void accelerate()
        { \\ logic for acceleration }

        void brake()
        { \\ logic for brakes }
};
```

CLASSES

```
class Car
{
        string model;
        int numOfDoors;
        string color;

        void accelerate()
        { \\ logic for acceleration }

        void brake()
        { \\ logic for brakes }

};
```

HOW TO USE THE CAR?

```
int main()
{
        Car mycar;
        car.accelerate();
}
```

CLASS VS STRUCT

- Members of a class are **private** by default and members of struct are **public** by default
- Classes can be inherited whereas structures cannot
- Struct are value-type whereas classes are reference-type
- A structure can't be abstract, a class can be.

PROBLEM

```
int main()
{
        Car mycar;
        car.accelerate();
}
```

The function accelerate() cannot be accessed!

ACCESS MODIFIERS

- Access modifiers are used to control access to class members
- **public**, **private** & **protected** are three of the access modifiers available in C++
- In C++, class members are considered **private** when no access modifier is used

EXAMPLE

```
class BankAccount
{
        int accountNo;
        private:
                int PIN;
        public:
                int accountType;
}
```

GETTER/SETTER FUNCTIONS

- Getter functions (or accessor functions) are used to read value of a private member of some class

- Setter functions (or mutator functions) are used to modify the value of a private member of some class

EXAMPLE

```
class BankAccount
{
   int PIN;                    //private variable

   int get_PIN() const
   {
           return PIN;
   }
}
```

EXAMPLE

```
class BankAccount
{
 int accountNo;          //private variable

 void set_accountNo(int num)
 {
         accountNo = num;
 }
}
```

CASE STUDY 1

A bank wants a simple application module to manage the accounts of its customers. For every new customer, the app must let us fill in the details including his Name, Age, NIC#, Address, Opening Balance, Current Balance, Contact# & PIN. These details may be modified later except for the PIN. At any given time, the customer can check his balance.

Also, tax must be calculated (Tax is 0.15% of the current balance for customers aged 60 or above and 0.25% for all other customers).

CASE STUDY 2

A student wants to write an application that calculates his CGPA & finds if he is eligible for a Thesis.

Eligibility for Thesis requires 26 credit hours passed & CPGA of 3.0 or higher.

CASE STUDY 3

You need to write a function that encrypts a given variable by taking its reference and increasing its value by 5.

But you do not know beforehand the type of variable (it can either be an integer, character or float).