CP317A Software Engineering

Low-level design – part 2 – week 4-2

Shaun Gao, Ph.D., P.Eng.

Agenda

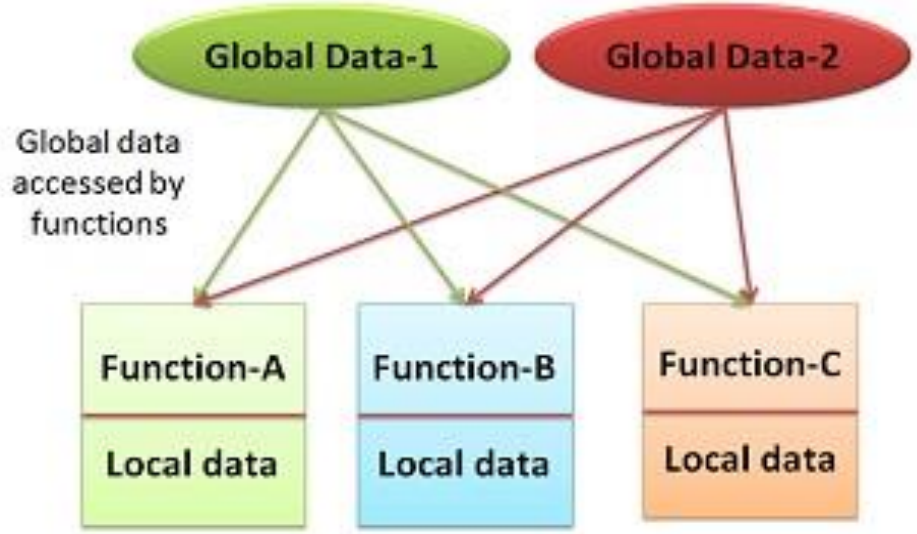
- Review week 4-1
- Introduction
  - Procedural-oriented design
- Object-oriented design
  - Abstraction
  - Inheritance
  - Polymorphism
  - Encapsulation
- Procedural-oriented design vs. object-oriented design
- Summary

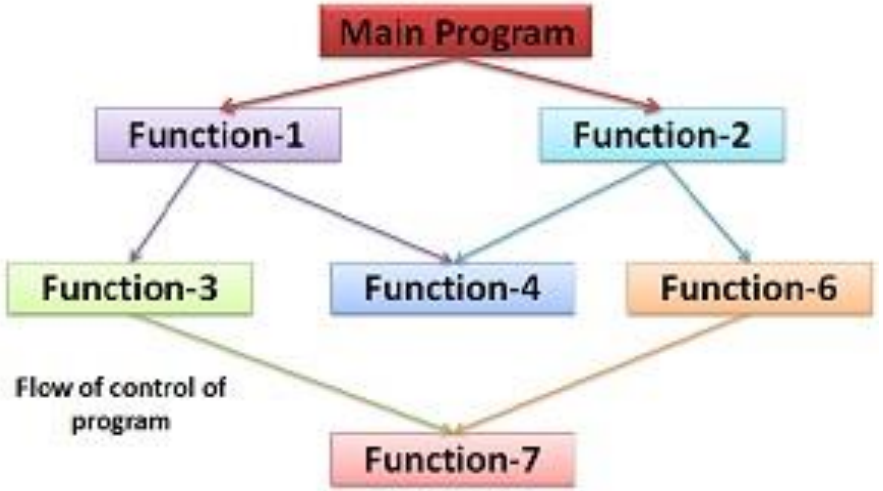
Review week 4-1

- Low-level (detailed) design
  - Concept
- Key tasks in detailed design
  - Understanding requirements, architecture and systems
  - Creating detailed design
    - **Interface design (internal and external)**
      - **Shared memory, inter-process communication (IPC)**
    - **Graphical User interface (GUI) design**
      - **ASCII table/code**
    - **Internal component design (structure and behavioral)**
    - **Data design, data structure design**
  - Documenting detailed design
  - Evaluating detailed design

Introduction

- Procedural-oriented design
  - the primary focus is on functions. Procedural-oriented design creates a step by step software that guides the application through a sequence of instructions. Each instruction is executed in a certain order.





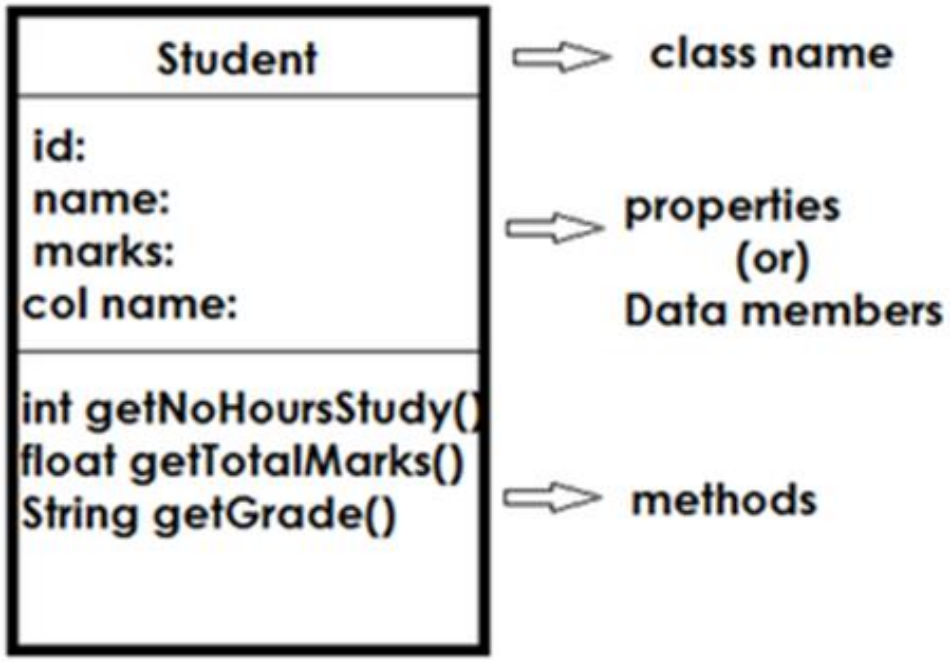
Structure of procedure oriented program

Introduction – cont.

- The disadvantages of procedural-oriented design
  - The data is not protected
  - If new data is added, all the functions need to be modified for accessing the data
  - It does not model real world problems very well
  - When software size becomes larger and more complex, the software becomes problematical such that
    - Difficult to understand and maintain
    - Difficult to modify and extend – lacks scalability
    - Easy to break (vulnerable)

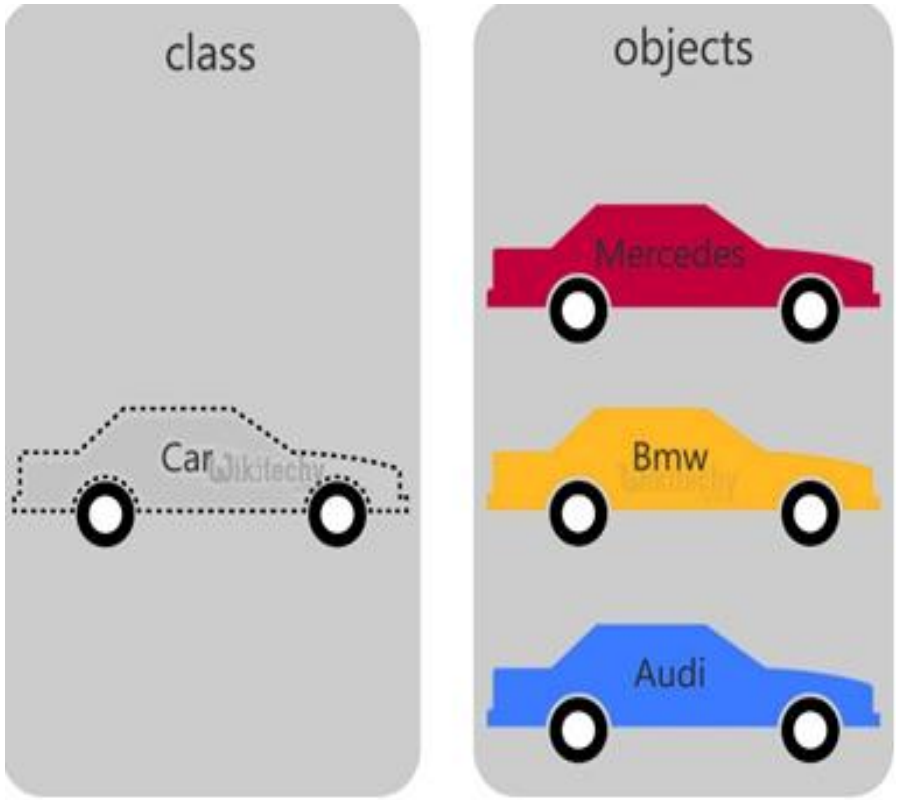
Object-oriented design

- Classes
  - A class is a syntactic unit used to define objects. A class usually contains properties, methods.
  - Constructor
    - A constructor is a special method of a class, which initializes an object.
    - Default constructor
  - Destructor
    - A destructor is a special method of a class, which is automatically invoked when an object is destroyed.
    - Default destructor



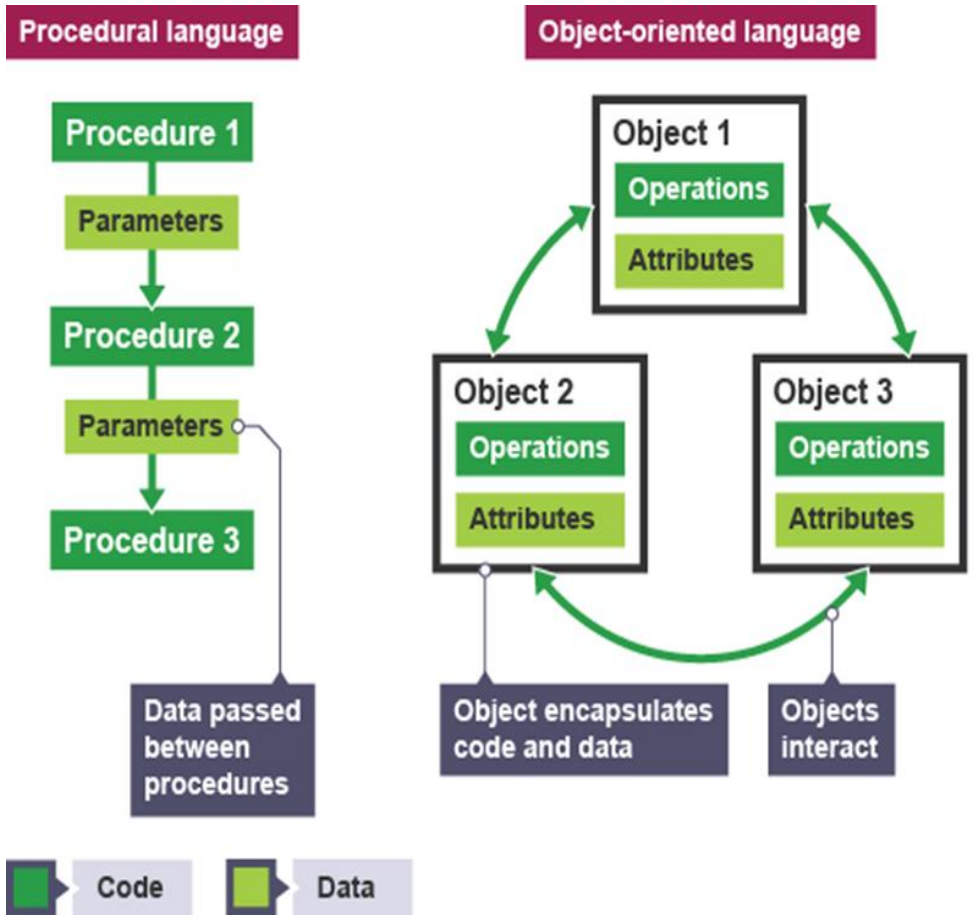
Object-oriented design – cont.

- Attributes
  - An attribute is the property of a class, which is a thing that can be measured or observed.
- Objects
  - An object is created from a class, by using or invoking the constructor of the class with matching parameter types.
- Relationship between a class and objects



Object-oriented design – cont.

- History: data ⇔ methods
- Object orientation combines data and methods together into a cohesive whole class.
- Object-oriented design (OOD)
  - OOD is the process of planning a system of interacting objects for the purpose of solving a software problem.
- Procedural-oriented design vs. object-oriented design



Object-oriented design – cont.

- The features of object-oriented design
  - Abstraction (Data hiding)
  - Inheritance
  - Polymorphism
  - Encapsulation
- The benefits of OOD
  - Code reusability – new objects can be derived from old objects
  - Code modularity – object based
  - Easier maintenance
  - Design stability

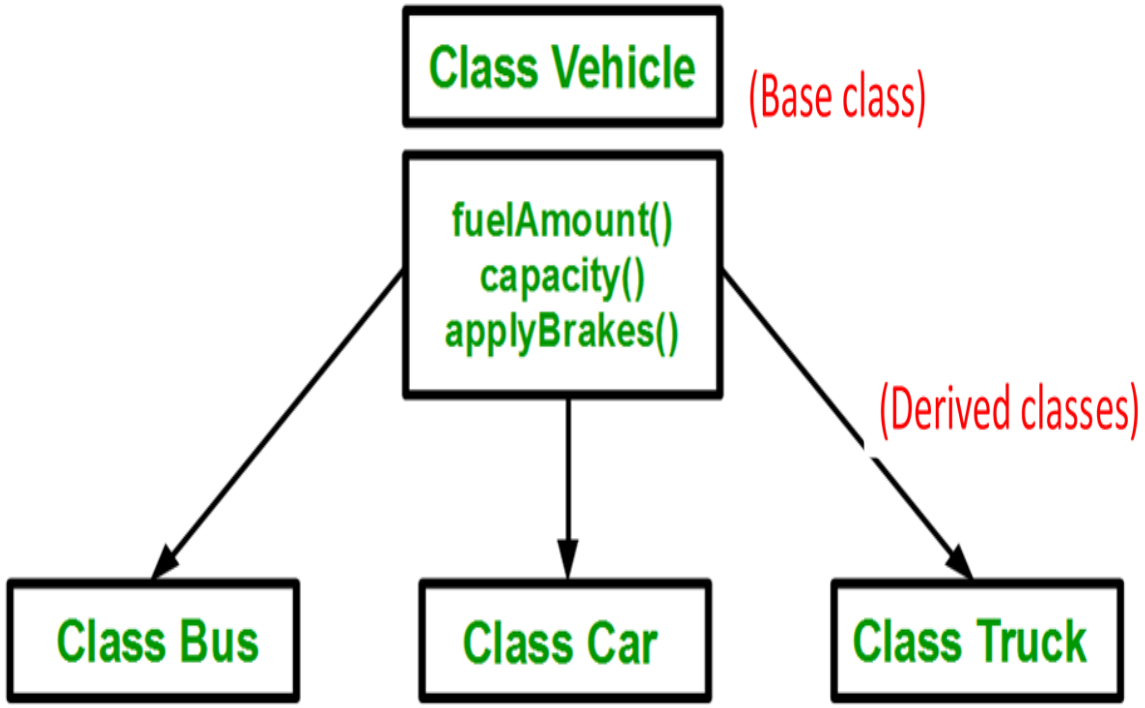
Object-oriented design – cont.

- Abstraction
  - Abstraction is a concept to hide unnecessary details and only show the essential features of the object. There is no implementation here its just a concept.
  - Benefits of abstraction:
    - Reduce the complexity
    - Improve the maintainability
- Examples
  - Abstract classes
    - You can create an abstract class by declaring at least one pure virtual member function in C++

```
using namespace std;
class Sample_Class {
  int a = 5;
  public:
  virtual void sample_func() = 0;
  void print_func() {
     cout << a;
  }
};
class Derived_Class : public Sample_Class {
public:
 void sample_func() {
   cout << "pure virtual function is implemented";
 }
};
```

Object-oriented design – cont.

- Inheritance
  - Inheritance is a mechanism in which derived classes acquire the property of a base class.
- Benefits of inheritance
  - Increased productivity – code reuse
  - Easier for maintenance



Object-oriented design – cont.

- Polymorphism
  - History: *the name of a method is unique*
  - Polymorphism refers to a programming language's ability to process objects differently depending on their data types or class.
  - Polymorphism = many forms
  - Benefits of polymorphism
    - Code reuse – reusability
    - Improves flexibility
    - Help reducing coupling between different components

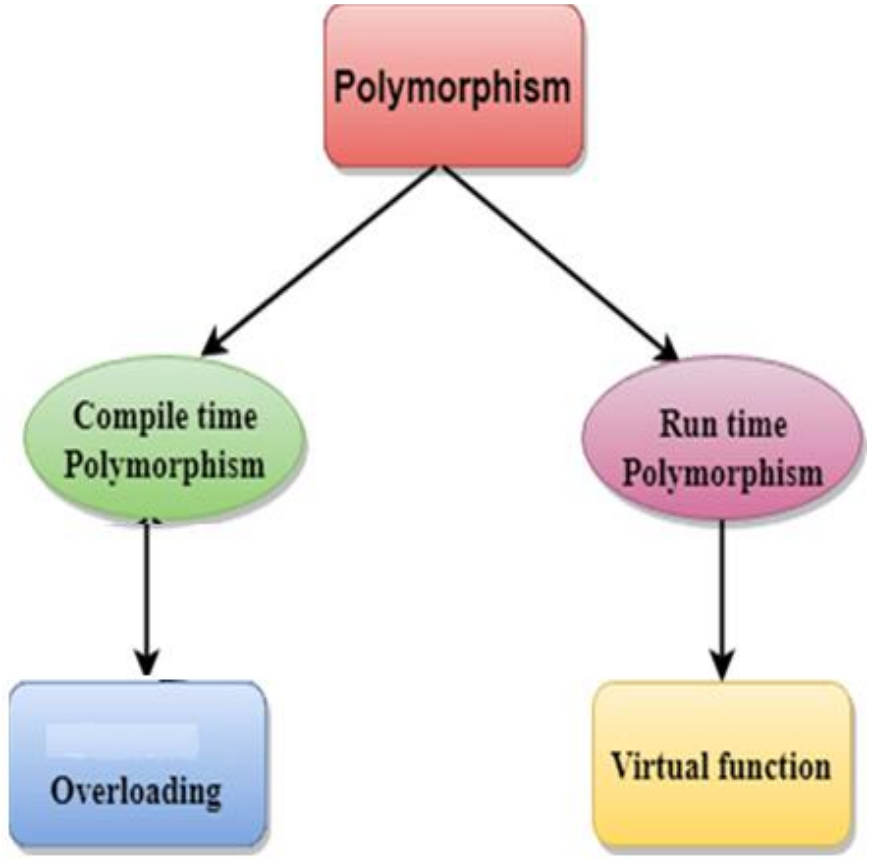
```
class Addition {
        public:
        void sum(int a, int b) {
                cout<<"a+b :"<<a+b;
        }
        void sum(int a, int b, int c) {
                cout<<"a+b+c :"<<a+b+c;
        }
};
int  main() {
        Addition obj;
        obj.sum(10, 20);
        cout<<endl;
        obj.sum(10, 20, 30);
}
```

Object-oriented design – cont.

- Two types of polymorphism
  - **Compile time polymorphism**
  - **Run time polymorphism**
- Compile time polymorphism
  - **Compile time binding or static binding**
  - In C++, all non-virtual functions are bound at compile time.

```
Class TestA{
        Public:
                functX(int num){ …… }
                functX(float dec, int num){……}
                functX( ){……}
}
```

- Overloading: overloading is a technique that there are multiple functions or operators with the same name, but their parameters are different within a class.
- Why is overloading needed?



Object-oriented design – cont.

- Run-time polymorphism = Run-time binding or dynamic binding
  - Run-time binding is to associate a function's name with the entry point at run time.
  - C++ supports run-time binding through virtual function.
  - Keyword virtual is used to declare a function is a virtual function.
  - Derived classes have their own implementation of the virtual function, base class virtual functions are overridden in derived classes.
  - Overriding: If a derived class defines the same function as defined in its base class, it is known as function **overriding**.
- Method overriding allows a derived class to provide a specific implementation of a method that is already provided by one of its parent classes.

Object-oriented design – cont.

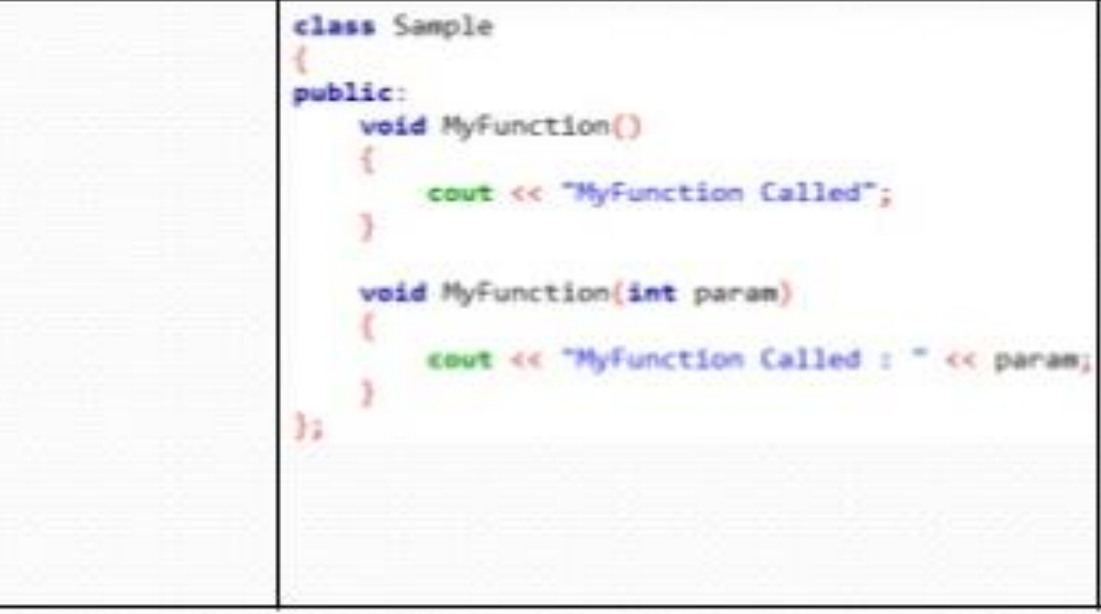
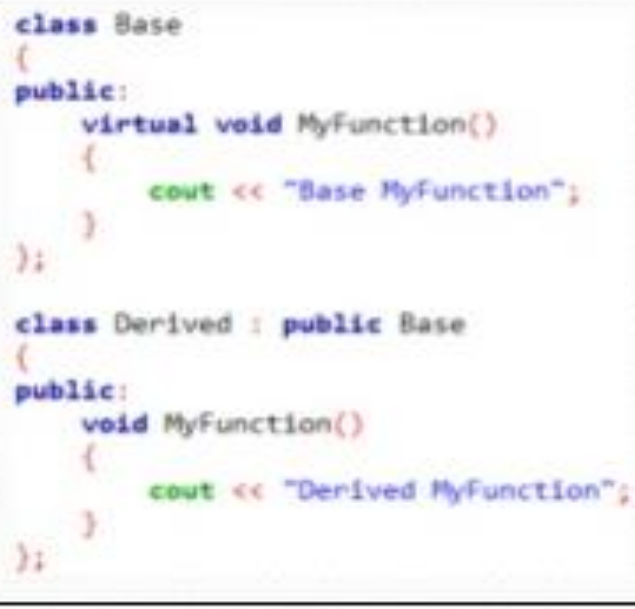
- An example of Function overloading

```
using namespace std;
class Geeks
{
    public:
    // function with 1 int parameter
    void func(int x)
    {
        cout << "value of x is " << x << endl;
    }
    // function with same name but 1 double
parameter
    void func(double x)
    {
        cout << "value of x is " << x << endl;
    }
    // function with same name and 2 int parameters
    void func(int x, int y)
    {
        cout << "value of x and y is " << x << ", "
<< y << endl;
    }
};
```

```
int main() {
   Geeks obj1;
   // Which function is called will depend on the parameters
passed
   // The first 'func' is called
   obj1.func(7);
   // The second 'func' is called
   obj1.func(9.132);
   // The third 'func' is called
   obj1.func(85,64);
   return 0;
}
Output:
value of x is 7
value of x is 9.132
value of x and y is 85, 64
```

Object-oriented design – cont.

Method Overloading vs. Overriding?

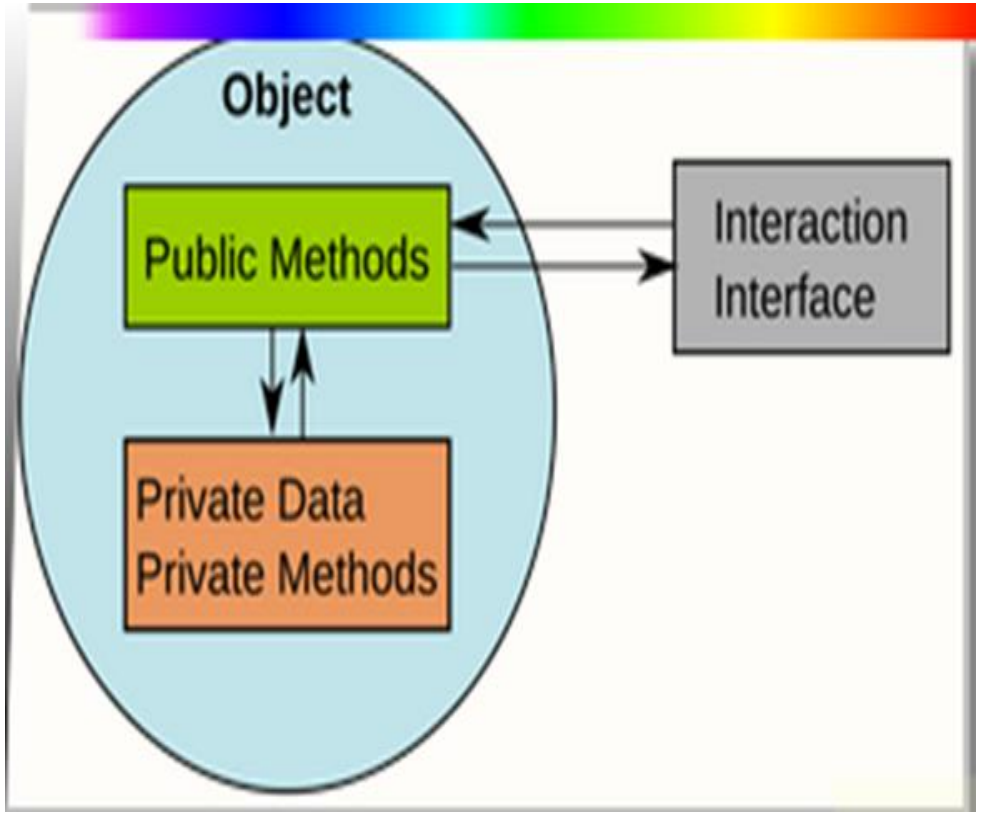
	Overloading	Overriding
Method Relationship	Relationship is between methods of same class.	Relationship is between methods of super class and sub class.
No of Classes	Does not require more than one class for overloading.	Requires at least 2 classes for overriding.
Example		

Object-oriented design – cont.

	Method Overloading	Method Overriding
1)	Method overloading is used *to increase the readability* of the program. **Flexibility**	Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class.
2)	It is performed *within class*.	It occurs *in two classes* that have inheritance relationship.
3)	*parameter must be different.*	*parameter must be same.*
4)	It is the example of *compile time polymorphism*.	It is the example of *run time polymorphism*.

Object-oriented design – cont.

- Encapsulation
  - Encapsulation is the hiding of information.
    It prevents users from seeing the internal working of an object
- Examples:
  - Setter(…) and getter()
- Benefits of encapsulation
  - Improves software reliability
  - Easier for maintenance
  - Reusability



Object-oriented design – cont.

Example of Encapsulation in C++

```
class sum {
        private: int a,b,c;
        public:
        void add() {
                cout<<"Enter any two numbers: ";
                cin>>a>>b;
                c=a+b;
                cout<<"Sum: "<<c;
        }
};
int main() {
        sum s;
        s.add();
}
```

In this example all data and function are bind inside class sum.

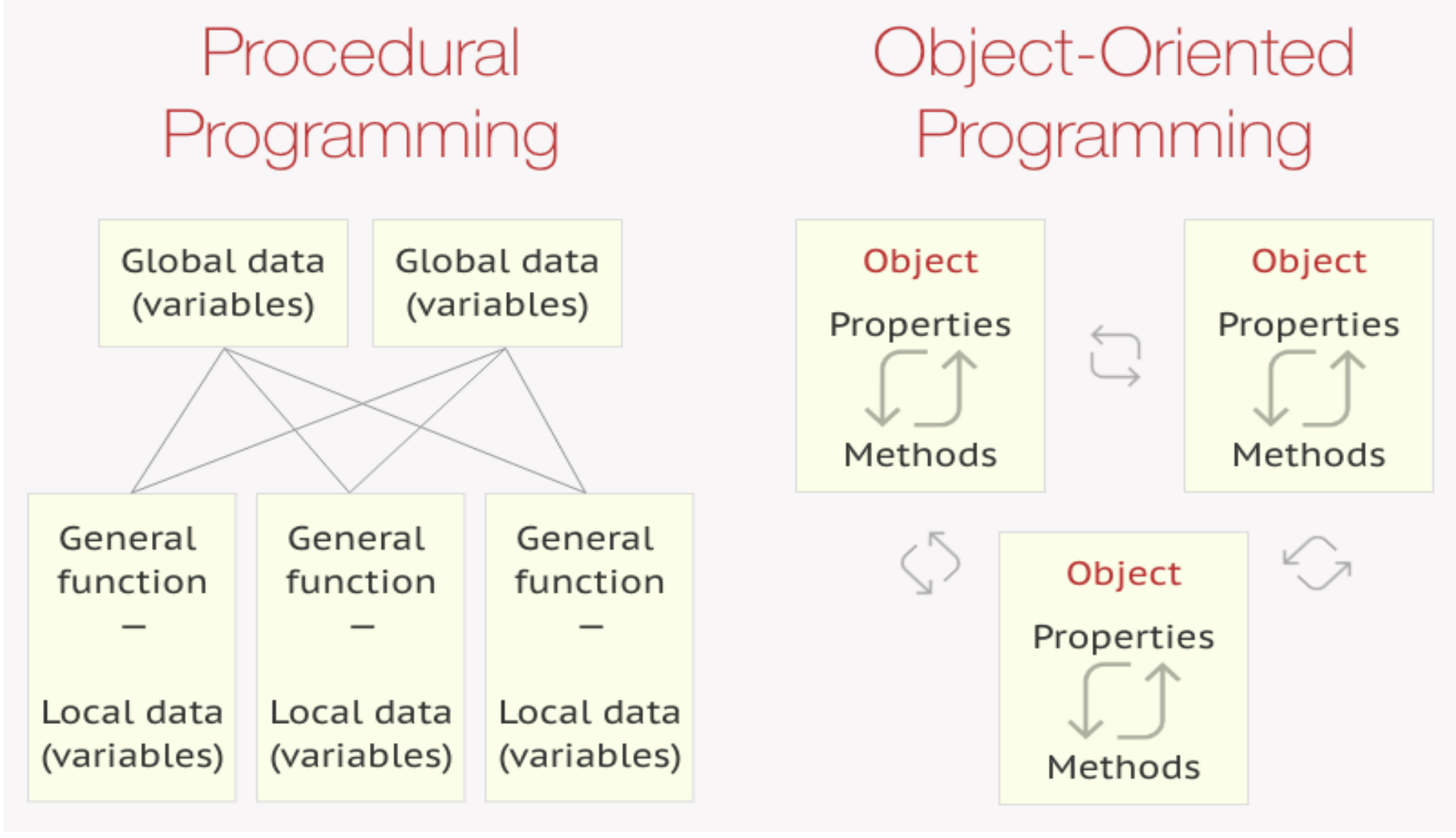
```
Enter any two number:
5
5
Sum: 10
```

Abstraction vs. Encapsulation

- Abstraction means- hiding implementation using abstract class and interfaces etc.
- Encapsulation means-hiding data like using getter and setter etc.

Abstraction	Encapsulation
Abstraction is a general concept formed by extracting common features from specific examples or The act of withdrawing or removing something **unnecessary**.	Encapsulation is the mechanism that binds together code and the data it manipulates, and keeps both **safe from outside interference** and **misuse**.
You can use abstraction using **Interface** and **Abstract** Class	You can implement encapsulation using **Access Modifiers** (Public, Protected & Private)
Abstraction solves the problem in **Design** Level	Encapsulation solves the problem in **Implementation** Level
For simplicity, abstraction means hiding implementation using Abstract class and Interface	For simplicity, encapsulation means hiding data using getters and setters

POD vs. OOD


Procedural Programming
Global data (variables)
Global data (variables)
General function –
Local data (variables)
General function –
Local data (variables)
General function –
Local data (variables)
Object-Oriented Programming
Object
Properties
Methods
Object
Properties
Methods
Object
Properties
Methods

POD vs. OOD

Advantages

POP (Procedure Oriented Programming)

- Provides an ability to reuse the same code at various places.
- Facilitates in tracking the program flow.
- Capable of constructing modules.

OOP (Object Oriented Programming)

- Objects help in task partitioning in the project.
- Secure programs can be built using data hiding.
- It can potentially map the objects.
- Enables the categorization of the objects into various classes.
- Object-oriented systems can be upgraded effortlessly.
- Redundant codes can be eliminated using inheritance.
- Codes can be extended using reusability.
- Greater modularity can be achieved.
- Data abstraction increases reliability.
- Flexible due to the dynamic binding concept.
- Decouples the essential specification from its implementation by using information hiding.

Disadvantages

POP (Procedure Oriented Programming

- Global data are vulnerable.
- Data can move freely within a program
- It is tough to verify the data position.
- Functions are action-oriented.
- Functions are not capable of relating to the elements of the problem.
- Real-world problems cannot be modelled.
- Parts of code are interdependent.
- One application code cannot be used in other application.
- Data is transferred by using the functions.

OOP (Object Oriented Programming)

- It requires more resources.
- Dynamic behaviour of objects requires RAM storage.
- Detection and debugging is harder in complex applications when the message passing is performed.
- Inheritance makes their classes tightly coupled, which affects the reusability of objects.

Think – Pair – Share

- For group project, which OOP features can be used?

Summary

- Procedural-oriented design
- Object-oriented design
  - Abstraction
    - Concept
  - Inheritance
    - Concept
  - Polymorphism
    - Concept
  - Encapsulation
    - Concept
- Differences between POD and OOD

Announcement

- 75% of you have a group. Please find a group ASAP. Please let me know if you need help

- Oct. 8 (Tuesday) test 1 (60 minutes, cover week 1 - week 4). Please bring your laptop
  - Locations
    - BA208 (the first letter of family name from A-H (42))
    - BA211 (the first letter of family name from I-P (31))
    - BA112 (the first letter of family name from Q-Z (28))