Qt Programming A Learning Experience

Dheerendra V Purohit
Qt Gardener

Nokia Certified
Qt Specialist

Nokia Certified
Qt Developer

My Profile

- ✓ Nokia Certified Qt Specialist and Nokia Certified Qt Developer

- ✓ B'Tech and M'Tech Computer Science from BITS Pilani
- ✓ Served as S/W Engineer, Lead, Manager in Cisco
- ✓ 14 years of Industry experience Unix, Windows and Networking
- ✓ Two patents
- ✓ IT Gardener with focus on Mobile Technology
- ✓ Volunteer for local community and NGO(Education)

Your Objective

- What is your objective ?
- Why you are here ?

- My objective is to learn C++

Your Objective

PthinkS
"Changing the way we THINK and LEARN"

- Do you have FIRE to learn ?
- You are here because of your parents paid money
- I want Job in TOP Companies
- I want one lakh salary every month!!!!!

Nokia Certified Qt Developer

Nokia Certified Qt Specialist

Objective

- I would like to learn C++
- OOOOPPPPPPPPPPSSS!!!!
- Classes and Objects
- Constructors and Destructors
- Inheritance
- Virtual Functions

How to program ?

- Algorithms
  - Formalize the problem
  - Choose the strategy to solve it
  - Break it into sub-problems
- Computer Language
  - Learning syntax and semantics
  - How to write a program
  - How to debug a program
- Programming Environment
  - Source code store and load
  - Visualize, Optimize, Manage Projects

C++ origin and Features

- An extension of the language C
  - Fast,
  - Portable
  - Widely used
- Written by Bjarne Stroustrup in 1979 at Bell Labs, then called “C with Classes”
- Object oriented language:
  - Encapsulation
  - Inheritance
  - Modularity and polymorphism

First Program

```
#include <iostream>
using namespace std
int main()
{
    cout << “Welcome to JVIT” << endl;
    Return 0;
}
```

First Program

- Declare the main function
  - Int main() …
- Declare your variables
  - Int a , b, char[10], float z
- Printing to the screen
  - cout << “Hello FSMK” << endl;
- Taking input
  - int a
  - cin >> “Enter the value of a “ << endl;
  - Cin >> a

Second Program

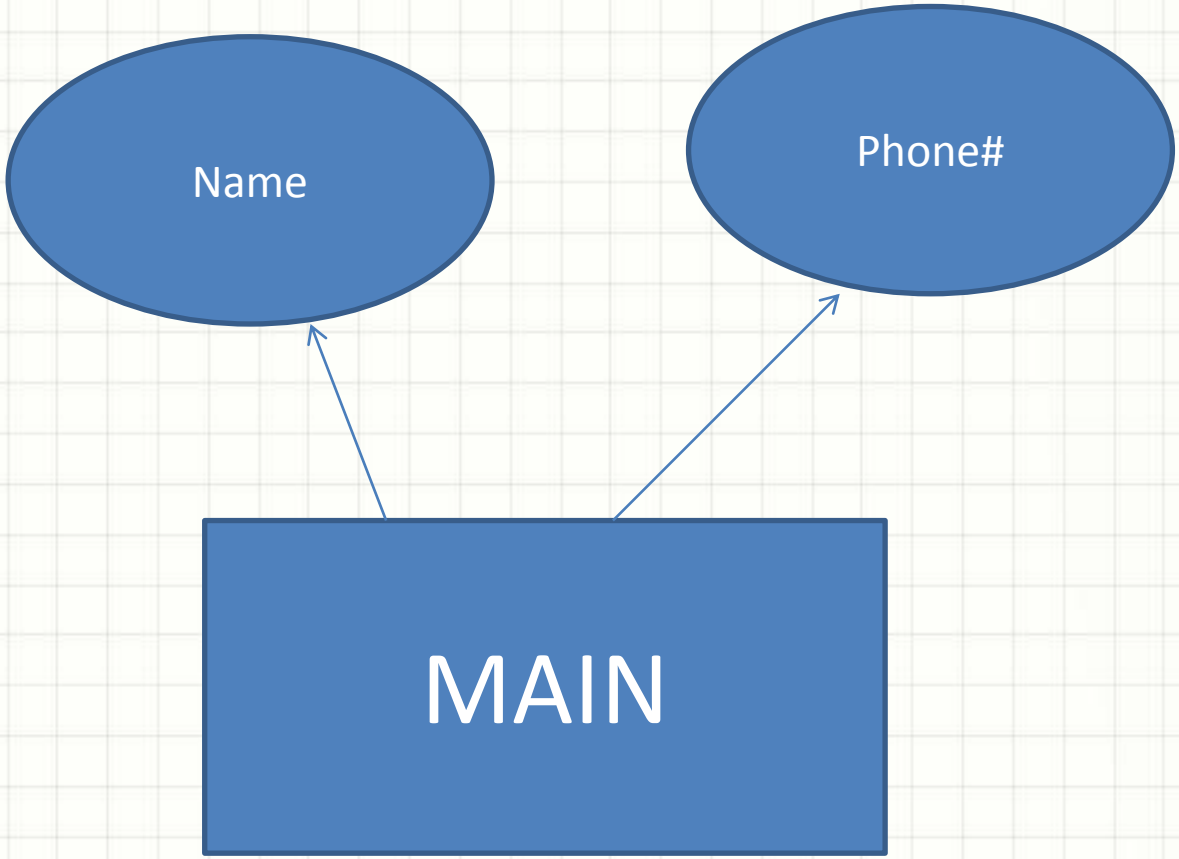
- ***Program to accept the name and phone number and print the same***
- {

```
cout << "Hello world" << endl;
int phoneNo; char name[20];
cout << "Enter the name = " ;
cin >> name;
cout << "Enter the phone# " ;
cin >> phoneNo;
cout << " Name = " << name;
cout << " Phone = " << phoneNo;
```

- }

Second Program

- ***Program to accept the name and phone number and print the same***



Third Program

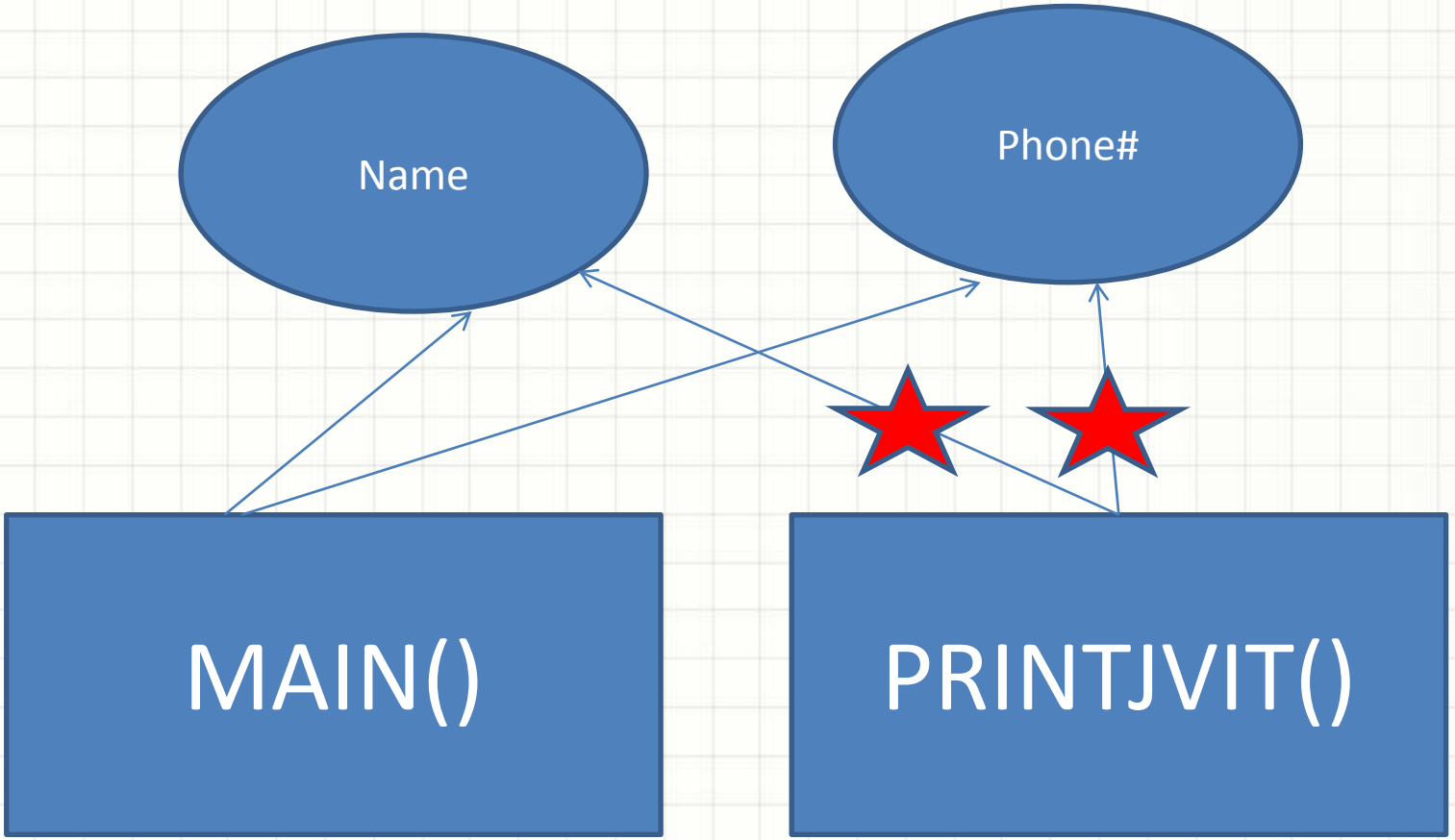
- ***Let us add one function to this program.***
- ***This function should print the "name" and "phone no".***
- Printjvit()
- {

```
//int phoneNo; char name[20];
cout << " Name = " << name;
cout << " Phone = " << phoneNo;
```

- }
- Compilation problem !!!
- Forward declaration
- Scope of Variables

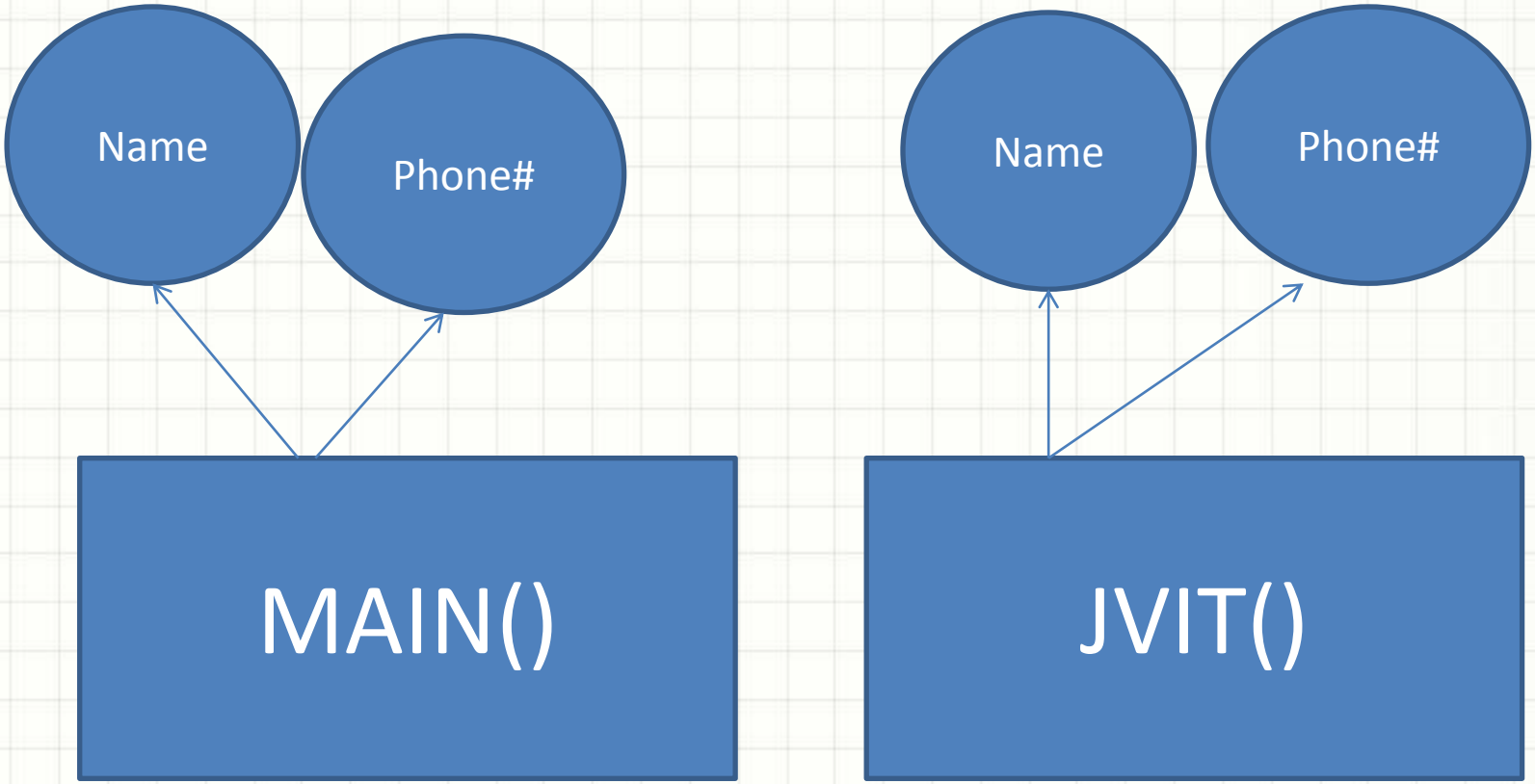
Third Program

- ***Program to accept the name and phone number and print the same***



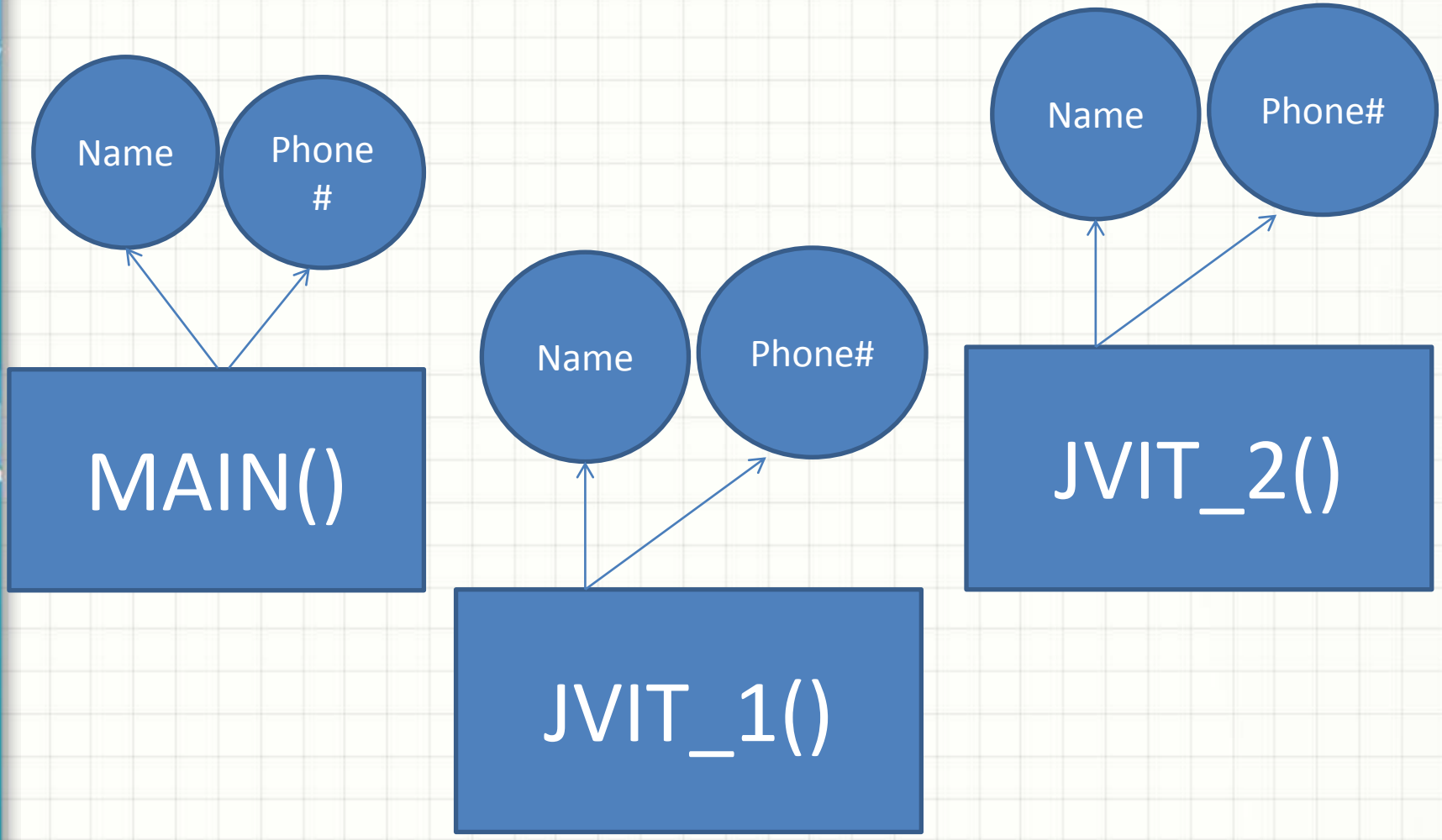
Third Program

- ***Program to accept the name and phone number and print the same***



Fourth Program

- ***Let us add one more function***



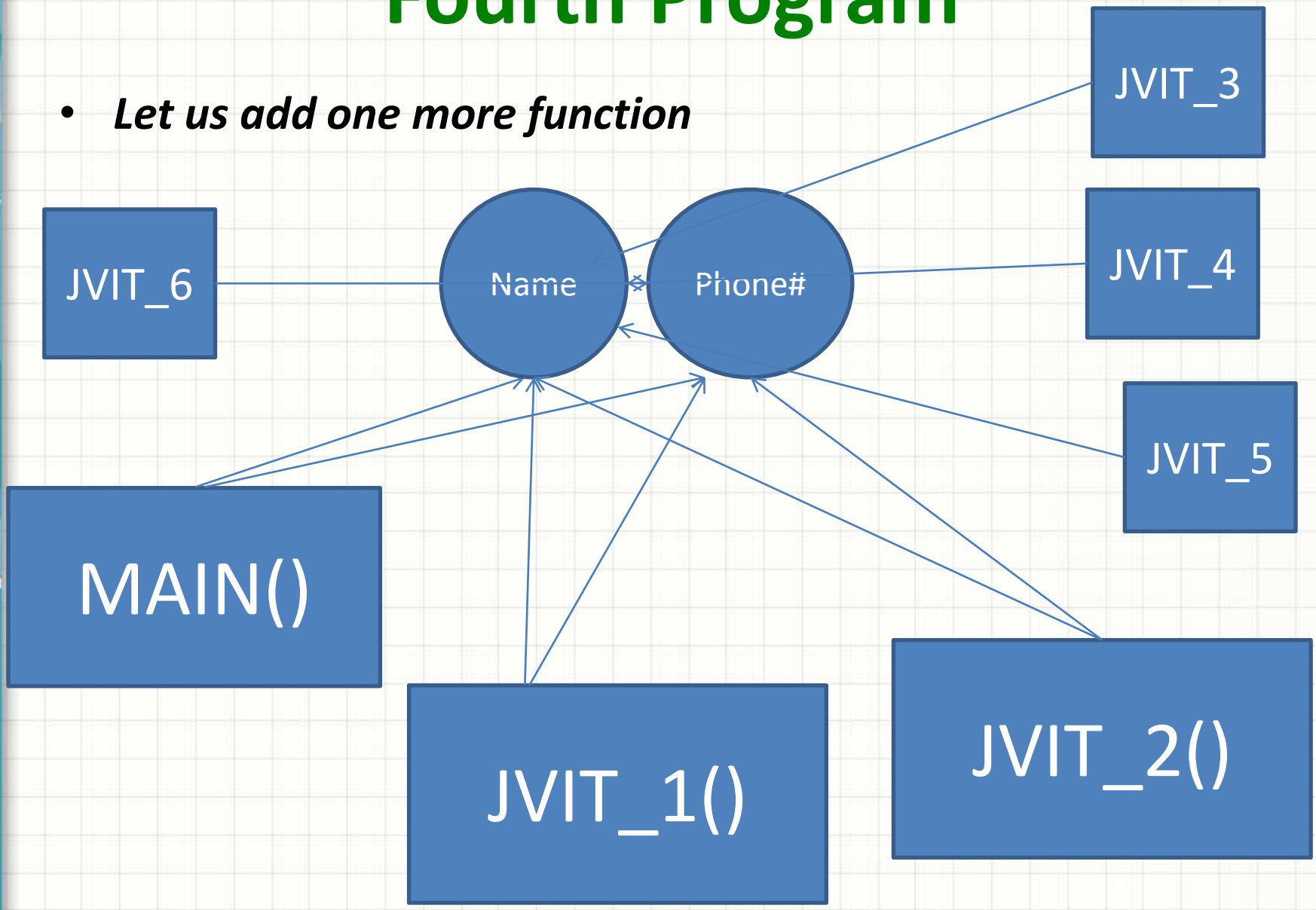
Fourth Program

- Variables are getting duplicated
- Each function is working on its own variable set
- YESSS ????

- I WOULD LIKE make all the functions to work on the same variable …
- i.e…

Fourth Program

- ***Let us add one more function***



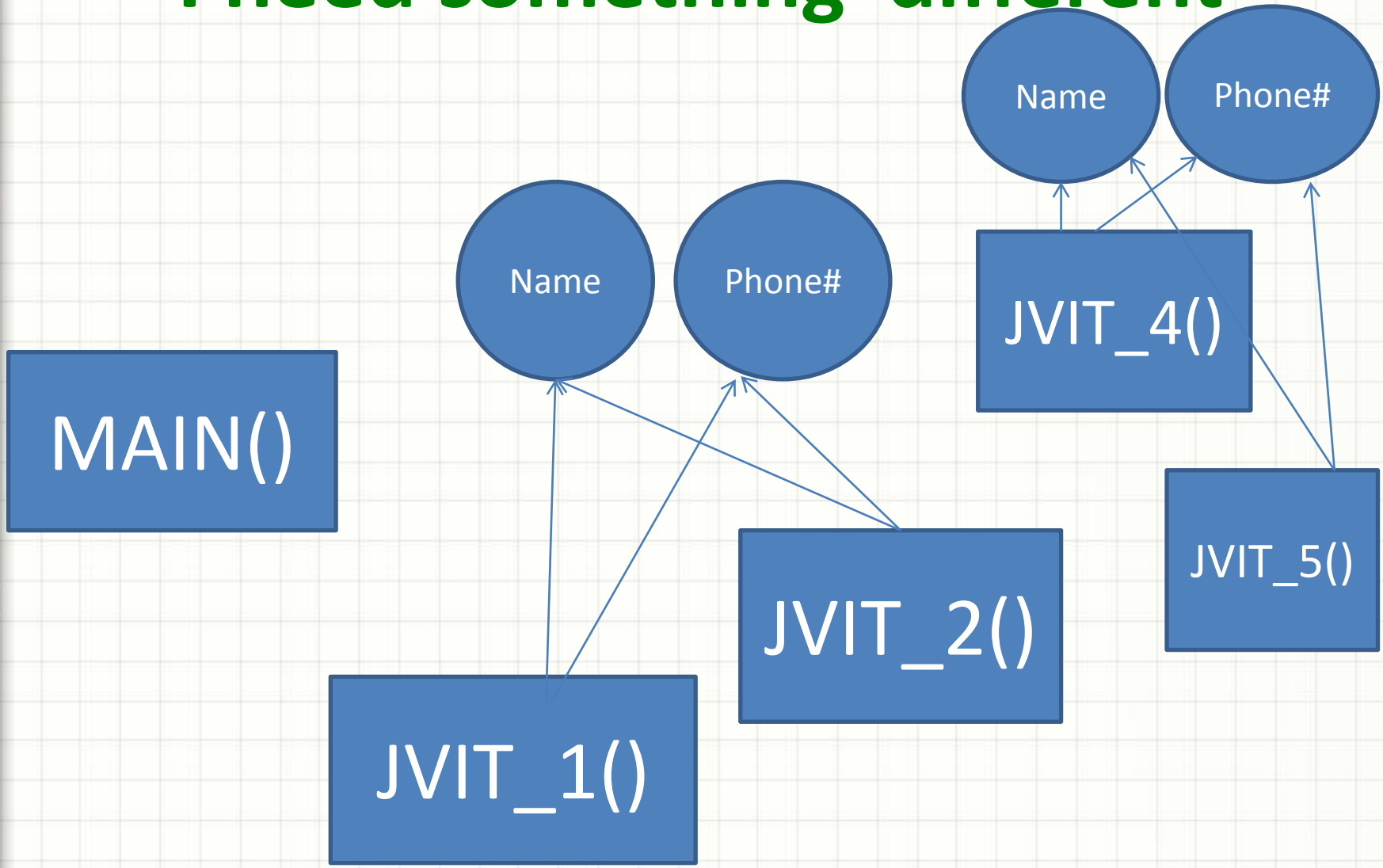
Fourth Program

- Variables are global now
- Each function is working on global variable set
- YESSS ????

- I WOULD LIKE make
- 3 functions to work on the few set of variables …

- Another 3 functions to work on the few set of variables …
- i.e…

I need something different



What do you need ?

- *I need group of functions*
- *I need group of variables only related to these functions*

- *How do I do it ????*

What is that…

Class……

- Yes.. It is C++ Class.
- Class groups the functions and variables on which these functions can work
- In C++ terms –
- Functions = Member Methods
- Variables – Data Members

OOOPSSS Concept

- First Two entry points for OOP Prgoramming..

- Data Encapsulation – Combining of Data & Functions
- Data abstractions – Representing features without worrying about internal implementation
- How this is done in C++ ?

Class

```
Class Person {
   char name[10]
   int phoneNo;

   void jvit_1();
   void jvit_2();
   void jvit_3();
}
```

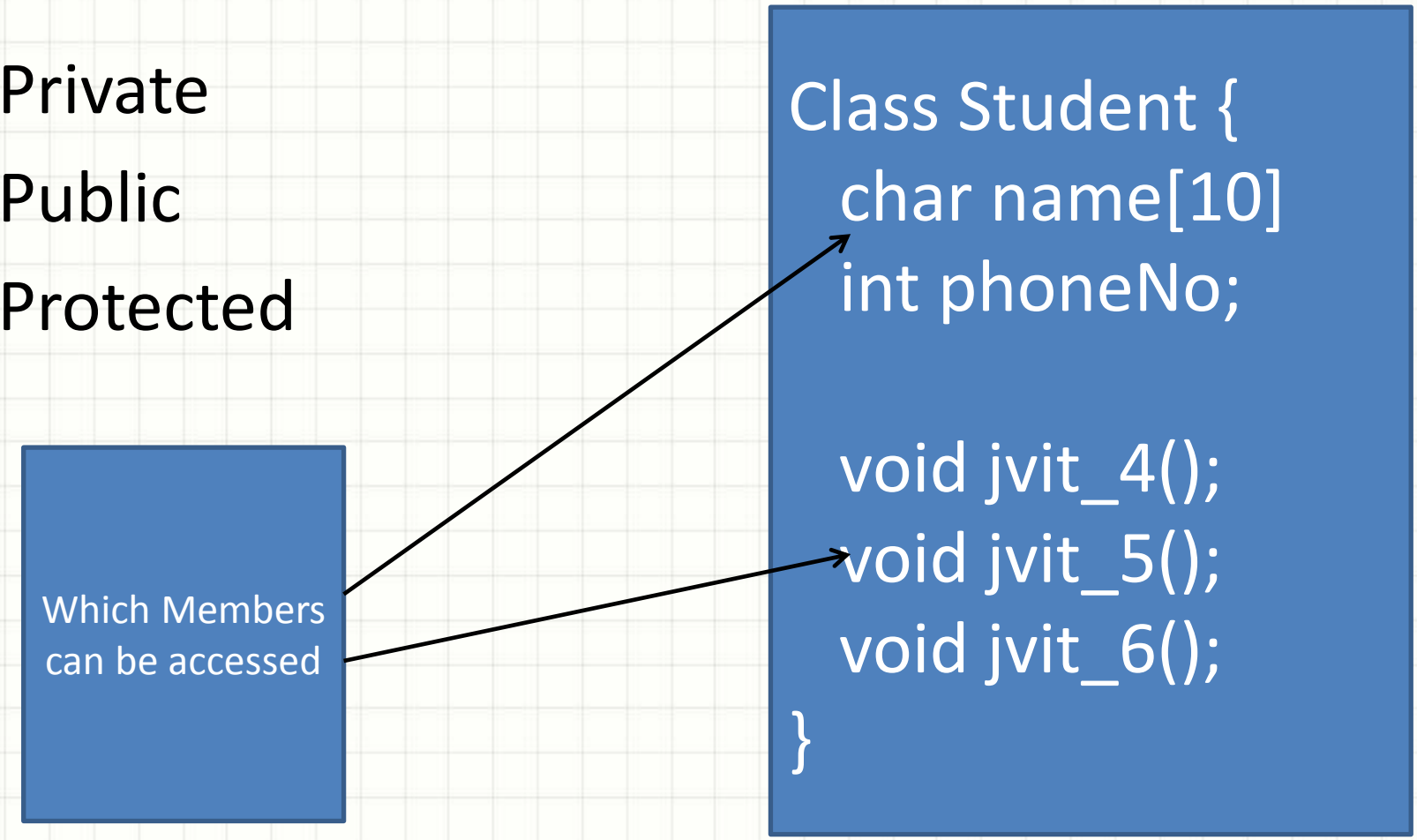
```
Class Student {
   char name[10]
   int phoneNo;

   void jvit_4();
   void jvit_5();
   void jvit_6();
}
```

Let us declare these classes

Accessibility

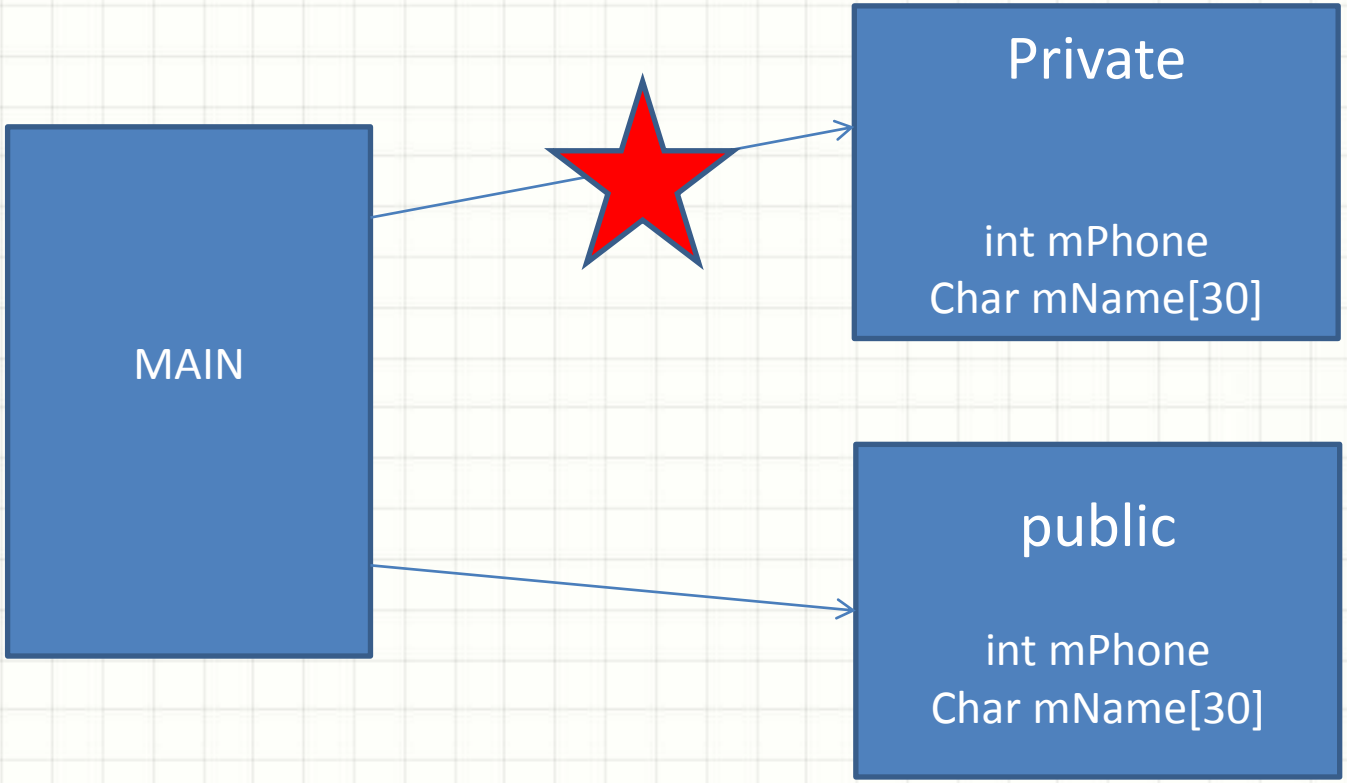
- Private
- Public
- Protected



Accessibility

- Which members can be accessed from outside the class
- Any member can access any member inside the class
- Private cannot be accessed outside the class
- Public can be accessed out

Accessibility



Analogy…

- Int a
- Float b
- Char c

- Person data;

- What is the similarity between int and Person ?
- What is the similarity between a and data ?

Member functions types

- Functions defined with in the class itself definition itself
- Const member functions – Functions which does not modify the data members. It is read only member functions
- Change the getData to const member functions

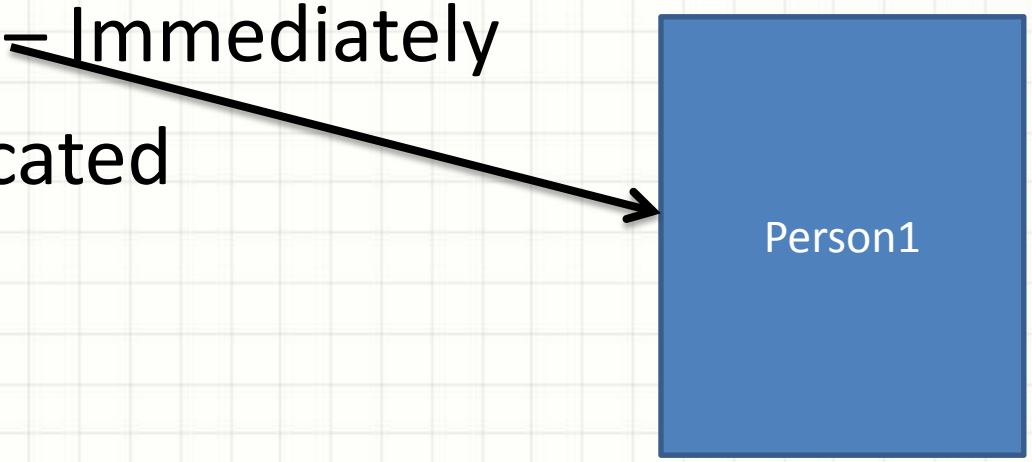
Creation and Accessing

- Person person1
- Person1.getData – Using the ‘dot’ operator

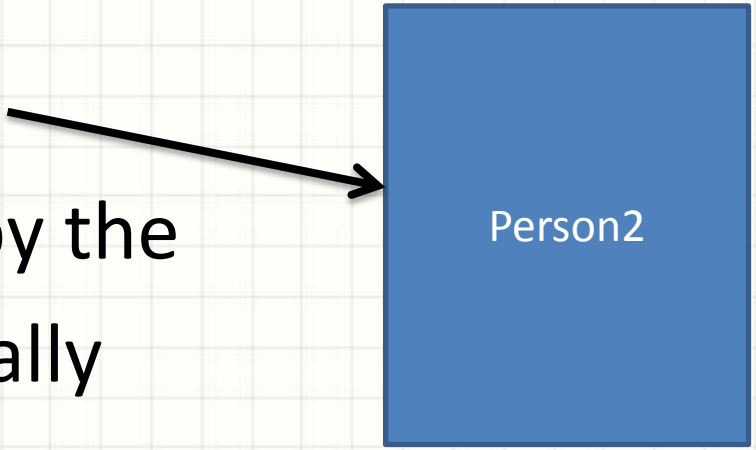
- Person *person2;
- Person2->getData() – Using the ‘pointer’ operator

Memory allocation

- Person person1– Immediately
- Memory is allocated

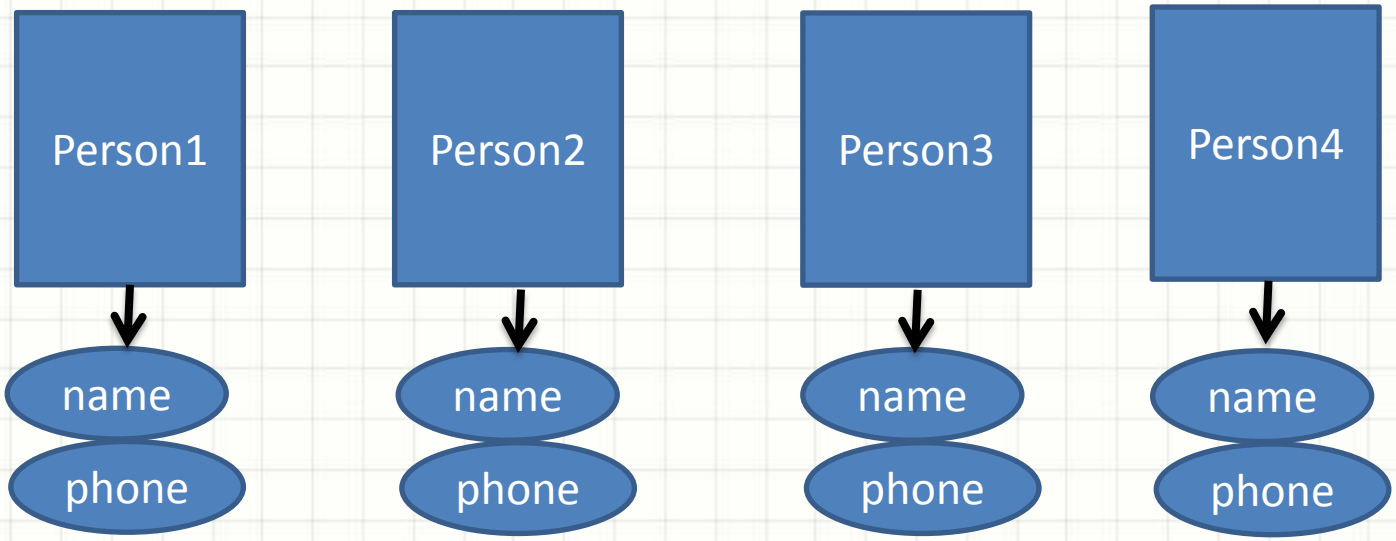


- Person *person2
- person2 = new Person
- Memory is allocated by the
- Programmer dynamically



Data members

- Data members are per object – For each object there will be separate copy of the data members



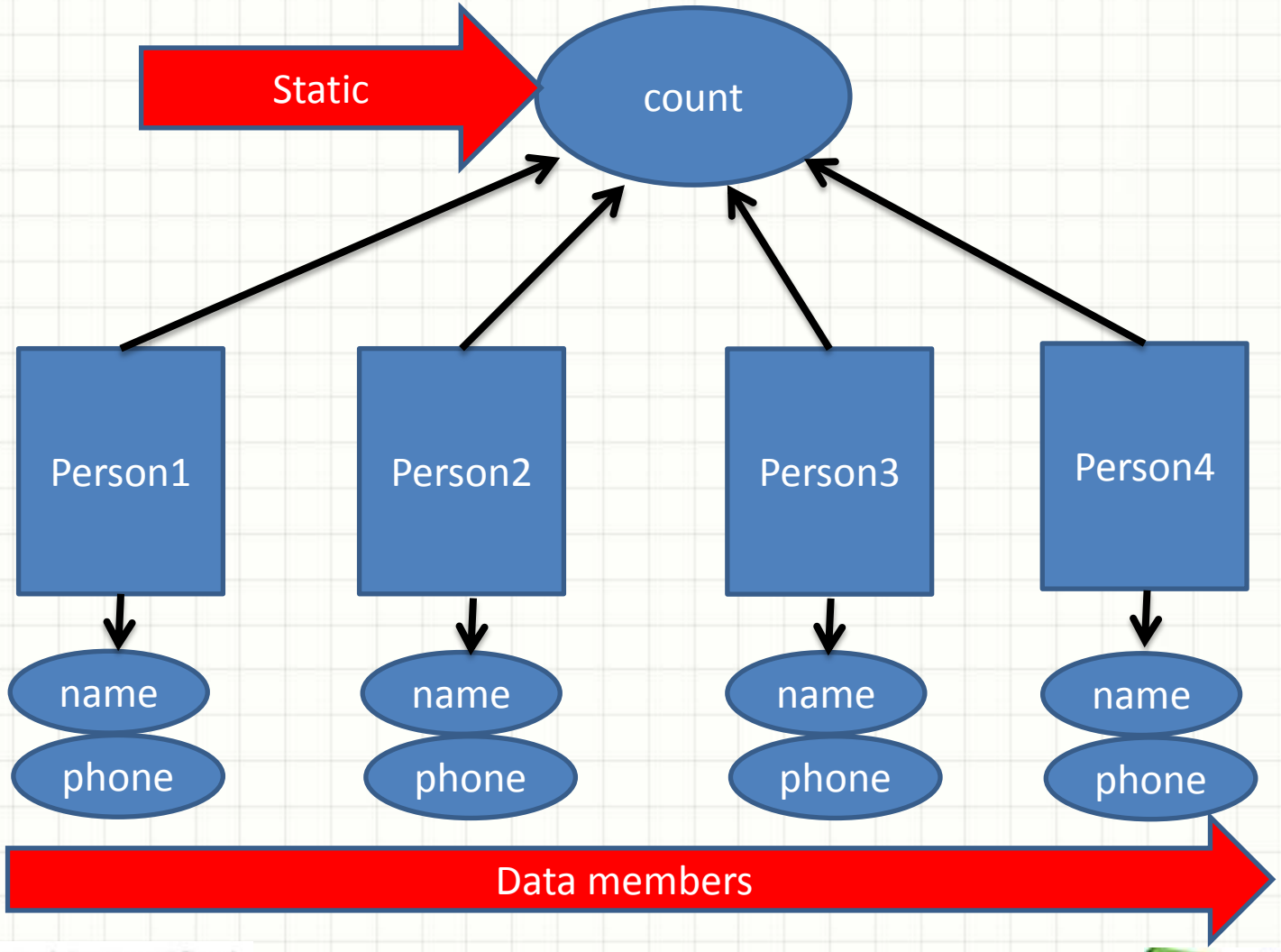
Static class members

- Static member is just like global variable of class
- There is only one instance maintained by class for all the objects of that class
- This member is only visible within the class and life time is entire program
- Declared within the class definition
- Defined outside the class definition

Static class members

- Class person {
- static int count;
- ….
- ….
  };
- Int Person::count = 0;

Static member



Constructors and Destructors

- When you created the person objects what is the value assigned to member variables ?
  - Name ?
  - PhoneNo ?

  - Please not “I asking when the object is created”.
  - There is nothing exist when the class is written by you.

Constructor

- What happens when you declare variable
- INT A
- Similarly what happens when you create variable of type Person
- Person B
- Person is your datatype
- It is a class
- It has data members

Constructor

- Constructor is member method
- It gets automatically called whenever you just create an object
- Please note – If not constructor, member method has to be explicitly called
- Two types of constructors
- Default Constructor – No arguments
- Argument constructor – Takes arguments

Constructor

- Class Person {
- ……..
- Public :
- Void Person()
- Void Person(int m, char name[10]);
- Void Person(int a);
- }

Constructor helps in default initialization of the objects

Destructors

- Required to destroy the object.
- Destructor is also member function,
- Name is same as constructor preceded by ~
- Destructor has no argument
- Compiler generates default destructor

Destructor helps in deletion of the objects

Exercises

- Example – 1
  - Create program to accept the name and phone number
  - Print the name and phone number
- Example -2
  - Add function to print above in function
- Example-3
  - Add function to read the variables
- Example-4
  - Add 10 member functions to print those variables

Exercise

- Example-5
  - Create class called Person with field names Name and phoneNo
  - Add method called getName().. to read the name
  - Add method called getPhone().. to read the phoneNo
  - Add method called printData().. to print the name and phoneNo
  - Create object
  - Create object dynamically

Exercise

- Example-6 – Use class Person
  - Add default constructor and initialize the name and phoneNo
  - Add two argument constructor to initialize the name and phoneNo
  - Add default arguments constructor
  - Please add ‘cout’ statement in every constructor
  - Add destructor

Exercise

- Example-7 – Use class Person
  - Dynamically allocate memory for name and destroy in destructor
  - Change the field name declaration to pointer variable – char *mName
  - Every constructor add statement to dynamically allocate memory

    mName = new char[40]

  -- Inside the destructor add following statement

    delete ~mName

Exercise

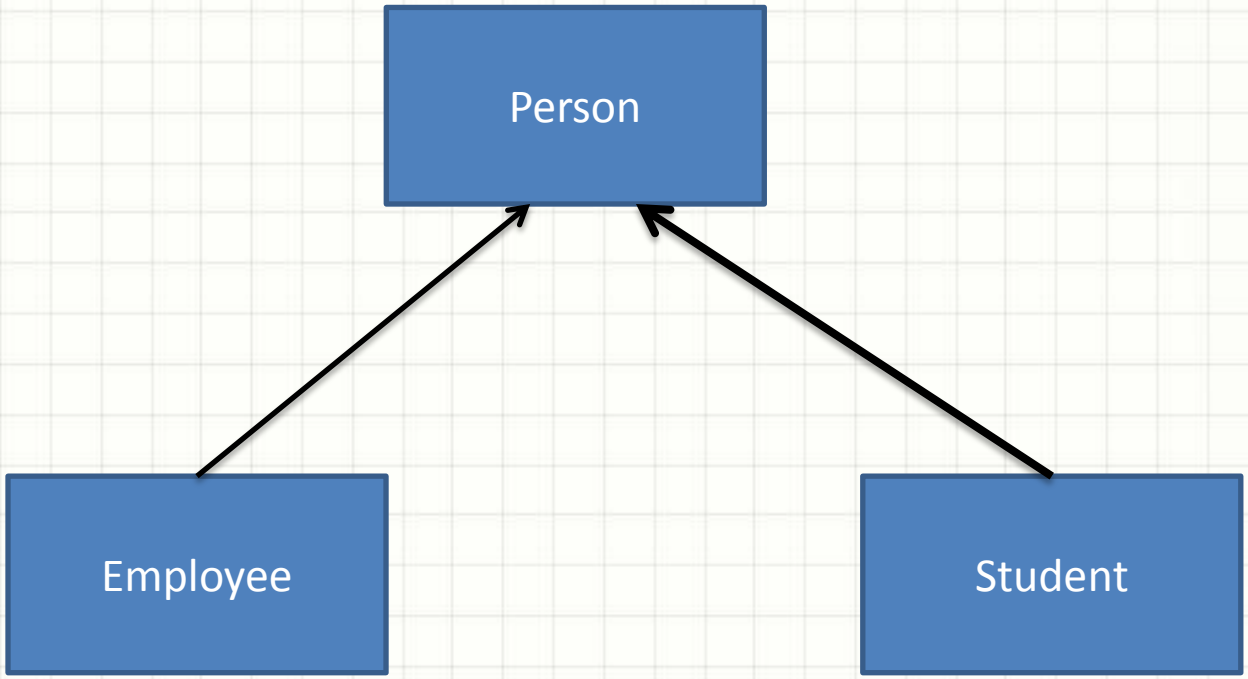
- Example-8 – Static Variable use case
  - Use class Person
  - Declare static variable called count to class person
  - Define and initialize the static variable
  - Create 10 objects of Person class
  - At the end print variable count.
  - This should print value 10. Please print through member method only

Inheritance

- Inheritance is super concept
  - WAW!!!!!! Super Super
- It is the feature of one class inheriting the properties another class
  - WAW!!!!!! Super Super
- It is the way of extending class
  - WAW!!!!!! Super Super
- It helps in software re-usability
  - WAW!!!!!! Super Super
- Give an example…
  - Car inheriting from vehicle

Inheritance

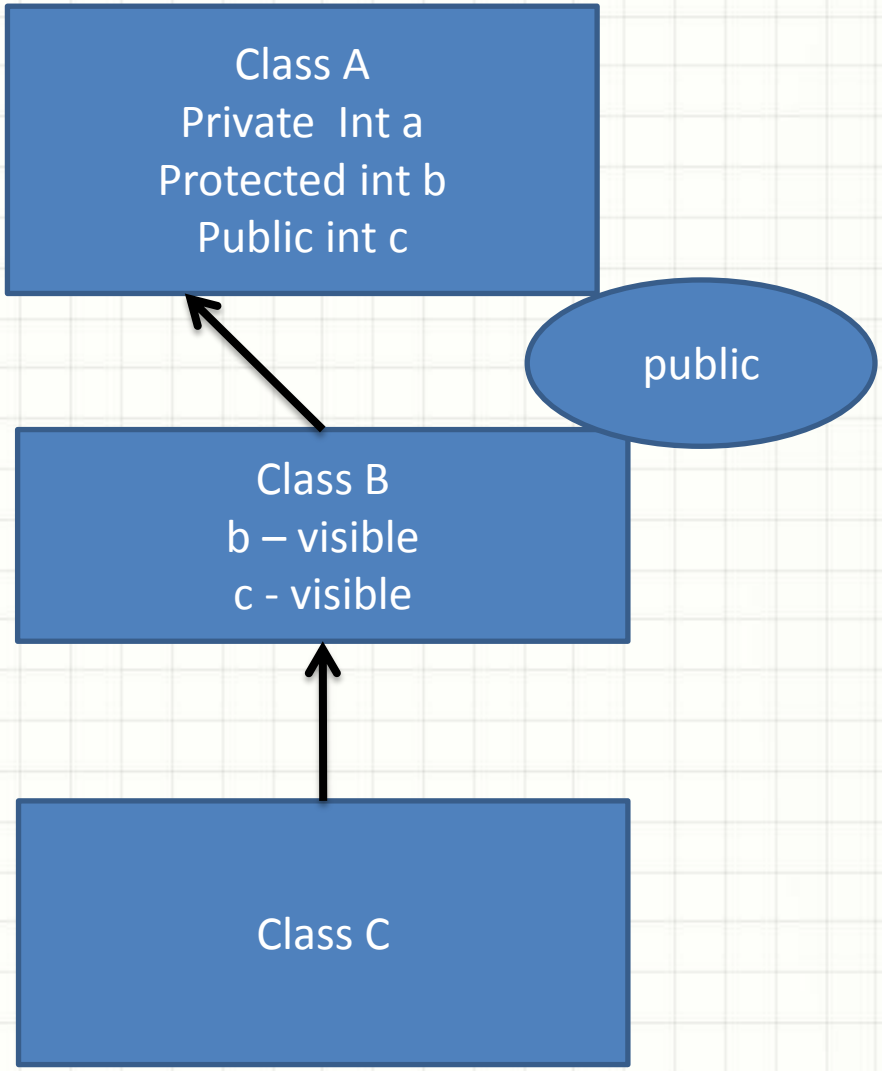
- I would like to create Employee
- I would like to create Student
- How do we do it ?



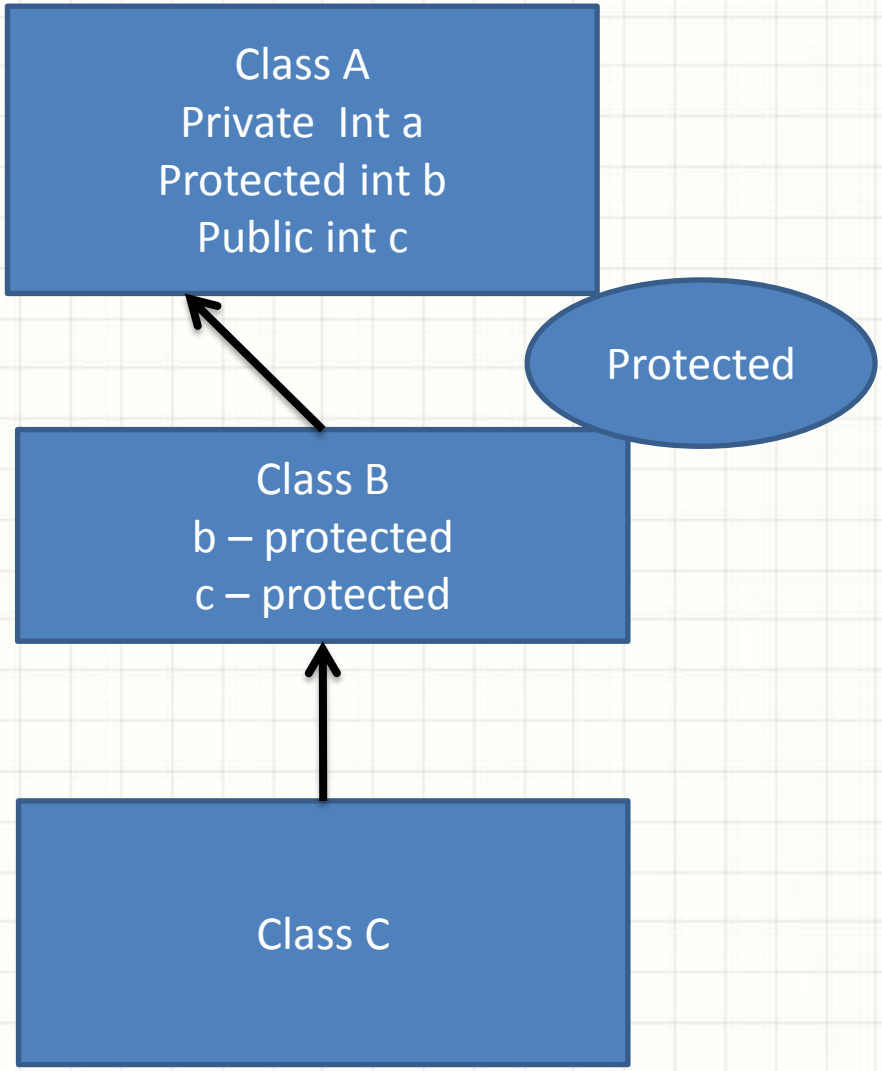
Inheritance

- Class Student : <visibilitymode> Person {
- }
- Private – Inheritance will stop
- Protected – Public will be upgraded to Protected
- Public – Public->public, Private->Private, Protected->Protected.

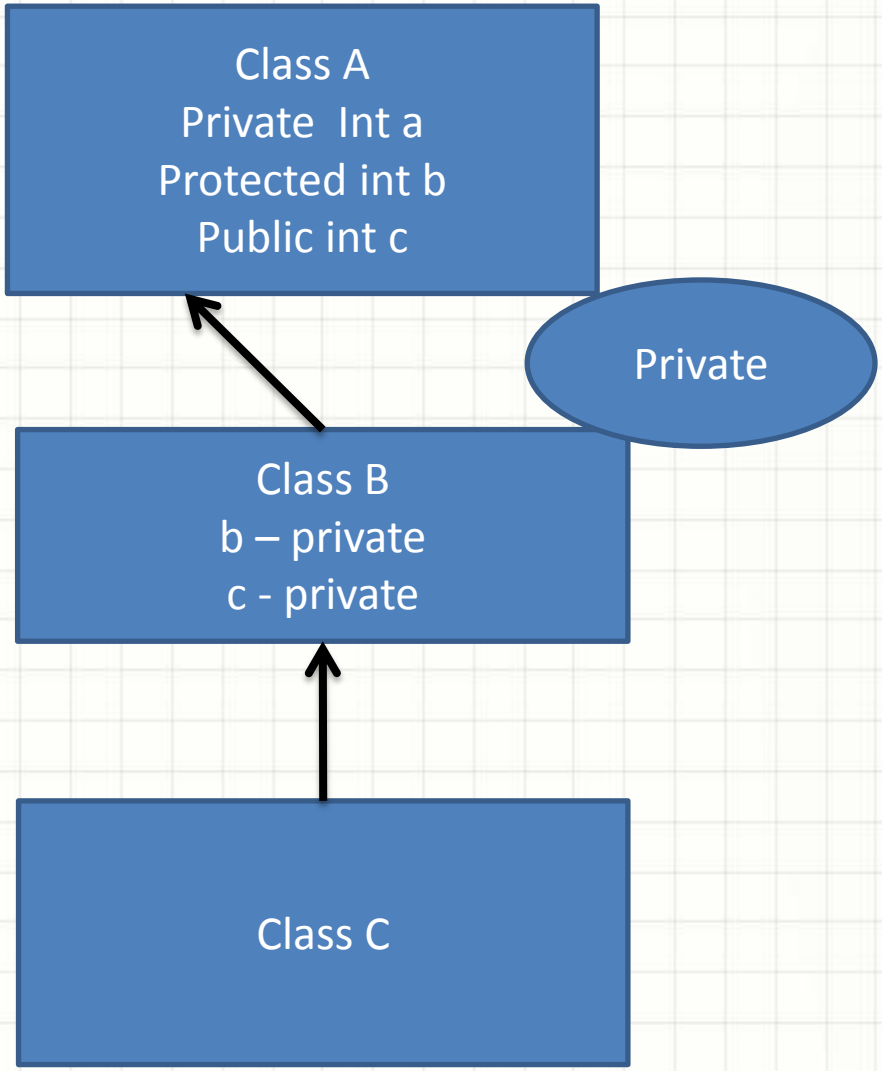
Inheritance



Inheritance



Inheritance



Inheritance

- Inheritance is super concept
  - WAW!!!!!! Super Super
- It is the feature of one class inheriting the properties another class
  - WAW!!!!!! Super Super
- It is the way of extending class
  - WAW!!!!!! Super Super
- It helps in software re-usability
  - WAW!!!!!! Super Super
- Give an example…
  - Car inheriting from vehicle