Classes

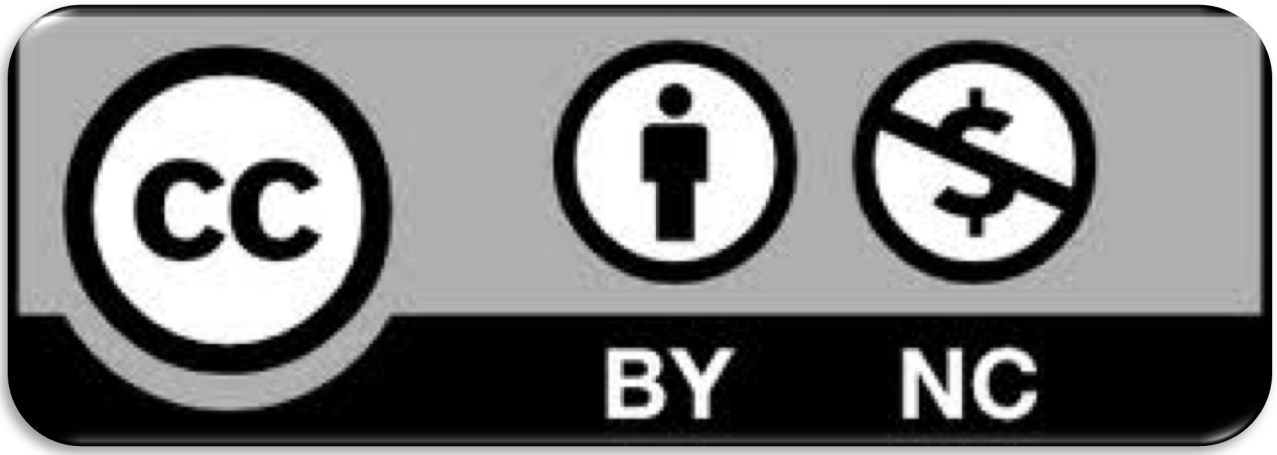
Alessandro Savino
Politecnico di Torino (Italy)

alessandro.savino@polito.it

www.testgroup.polito.it

License Information

This work is licensed under the Creative Commons BY-NC License



To view a copy of the license, visit:
http://creativecommons.org/licenses/by-nc/3.0/legalcode

Disclaimer

- **We disclaim any warranties or representations as to the accuracy or completeness of this material.**
- **Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.**
- **Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.**

Goal

- **This lecture presents a global overview about C++ classes and objects**

Prerequisites

- **C/C++ programming basis**

Homework

- **None**

Definition

- **Classes are an expanded concept of data structures: like data structures, they can contain data members, but they can also contain functions as members.**

- **An object is an instantiation of a class. In terms of variables, a class would be the type, and an object would be the variable.**

Definition

- **Classes:**
  - **Fundamental units for encapsulation**
  - **Allow you to create objects**
  - **Model that defines the appearance of an object**
  - **Set of plans that specify how to build an object**
  - **provide logical abstraction**

Behavior

- **Classes are defined using either keyword class or keyword struct, with the following syntax:**

```
class class-name {
public:
 public functions and data
private:
 private functions and data
};
```


Behavior

- **Classes are defined using either keyword class or keyword struct, with the following syntax:**

```
class class-name { New type for the objects
public:
 public functions and data
private:  ENCAPSULATION
 private functions and data
};
```


Behavior

- **Public and private sections can be ordered as you prefer; best practices say to put public first.**

```
class class-name {
public:
 public functions and data
private:
 private functions and data
};
```


Behavior

- **When not specified, everything is private by default:**

```
class class-name {
   int a, b;
   int somma(int x, int y);
public :
   void stampa();
};
```


Behavior

- **When not specified, everything is private by default:**

```
class class-name {
   int a, b;                          PRIVATE
   int somma(int x, int y);
public :
   void stampa();
};
```


First example

- **Hello world!!**

```
class HelloClass {
public:
 void printGreetings()
 {
     printf(”Hello!!\n”);
 }
};
```


First example

- **Hello world!!**

```
class HelloClass {
public:
 void printGreetings()
 {
     printf(”Hello!!\n”);
 }
};
```

Class definition

Public member functions

First example

- **Hello world!!**

```
class HelloClass {
public:
 void printGreetings()
 {
     printf(”Hello!!\n”);
 }
};
```

This is going to be replaced soon...

First example - 2

- **Hello world!!**

```
int main(int argc, char**argv)
{
  HelloClass salutation;
  salutation.printGreetings();
  return 0;
}
```


First example - 2

- **Hello world!!**

```
int main()
{
  HelloClass salutation;
  salutation.printGreetings();
  return 0;
}
```

Object instantiation or creation

Call object function

Class example

- **Let's model a generic vehicle:**

```
class Vehicle {
public:
 int passengers;
 int fuelcap;
 int kmpl;

};
```


Class example - 2

- **How to create (instantiate) a real vehicle?**

```
class Vehicle {
public:
 int passengers; // number of passengers
 int fuelcap;    // fuel capacity in litters
 int kmpl;        // fuel consumption in
                          km per litter
};
```


Class example - 3

- As for regular variables, class-name object-name

```
Vehicle minivan, station_wagon;
```

- Each of the two objects has its own copy of each variable defined in the class: the two objects do not share memory

- Access to variable defined in the class (class members) via the dot "." operator

```
minivan.passengers = 7;
```


Class example - 4

```
int main() {
  Vehicle minivan; // create a Vehicle object
  int range;
  minivan.passengers = 7;
  minivan.fuelcap = 60;
  minivan.kmpl = 7;

  range = minivan.fuelcap * minivan.kmpl;

  ...
  return 0;
}
```


Class example - 4

```
int main() {
  Vehicle minivan; // create a Vehicle object
  int range;
  minivan.passengers = 7;
  minivan.fuelcap = 60;
  minivan.kmpl = 7;

  range = minivan.fuelcap * minivan.kmpl;

  ...
  return 0;
}
```

Public members: accessible everywhere!

Example

```
int main() {
  Vehicle minivan; // create a Vehicle object
  Vehicle sportscar; // create another object
  int range1, range2;

  minivan.passengers = 7;
  minivan.fuelcap = 60;
  minivan.kmpl = 6;

  sportscar.passengers = 2;
  sportscar.fuelcap = 54;
  sportscar.kmpl = 5;
```


Example

```
  range1 = minivan.fuelcap * minivan.kmpl;
  range2 = sportscar.fuelcap * sportscar.kmpl;

  ...

  return 0;
}
```


Example

```
    range1 = minivan.fuelcap * minivan.kmpl;
    range2 = sportscar.fuelcap * sportscar.kmpl;

    ...

    return 0;
}
```

What is the value of range1 and range2 here?

Methods / Member functions

- **Model and change object behavior and data members**
- **Usually only accessible points for external world: class variables should be private**
- **As regular functions, they have both a prototype and implementation:**
  - **Prototype is declared within the class**
  - **Implementation everywhere with the "::" operator**

Methods / Member functions

- **As before, methods or member functions are accessed through the "." operator:**

```
class_name.function1()
```


Improved example

- **Let's model a generic vehicle:**

```
class Vehicle {
 int passengers;
 int fuelcap;
 int kmpl;
public:
 void set_members(int p, int f, int k);
 int range();
};
```


Improved example - 2

- **Implement the `range` method**

```
// Implement the range member function.
int Vehicle::range() {
  return kmpl * fuelcap;
}
```

- **Access the `range` method**

```
var = minivan.range();
```


File organization

- **`class_name.h` typically used to declare a class**

```
### Vehicle.h
#ifndef _VEHICLE
#define _VEHICLE
class Vehicle {
 int passengers;
 int fuelcap;
 int kmpl;
public:
 void set_members(int p, int f, int k);
 int range();
};
#endif //Vehicle
```


File Organization

- **`class_name.cpp` typically used to instantiate methods**

```
### Vehicle.cpp
int Vehicle::range() {
  return kmpl * fuelcap;
}
```


File Organization

- **`class_name.cpp` typically used to instantiate methods**

```
### Vehicle.cpp
int Vehicle::range() {
  return kmpl * fuelcap;
}
```

Binary scope resolution operator

File Organization - 2

Class implementation programmer

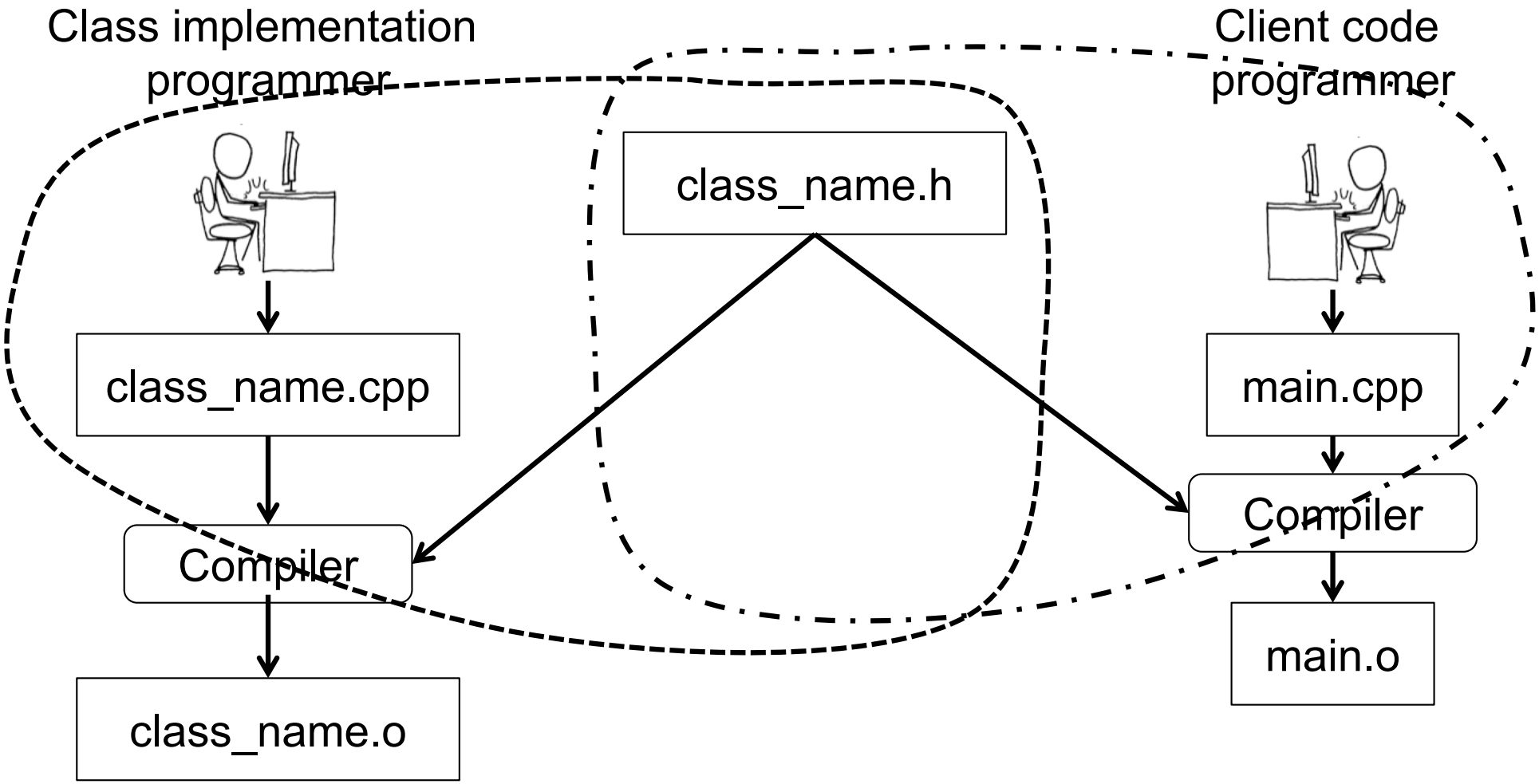
class_name.cpp

class_name.h

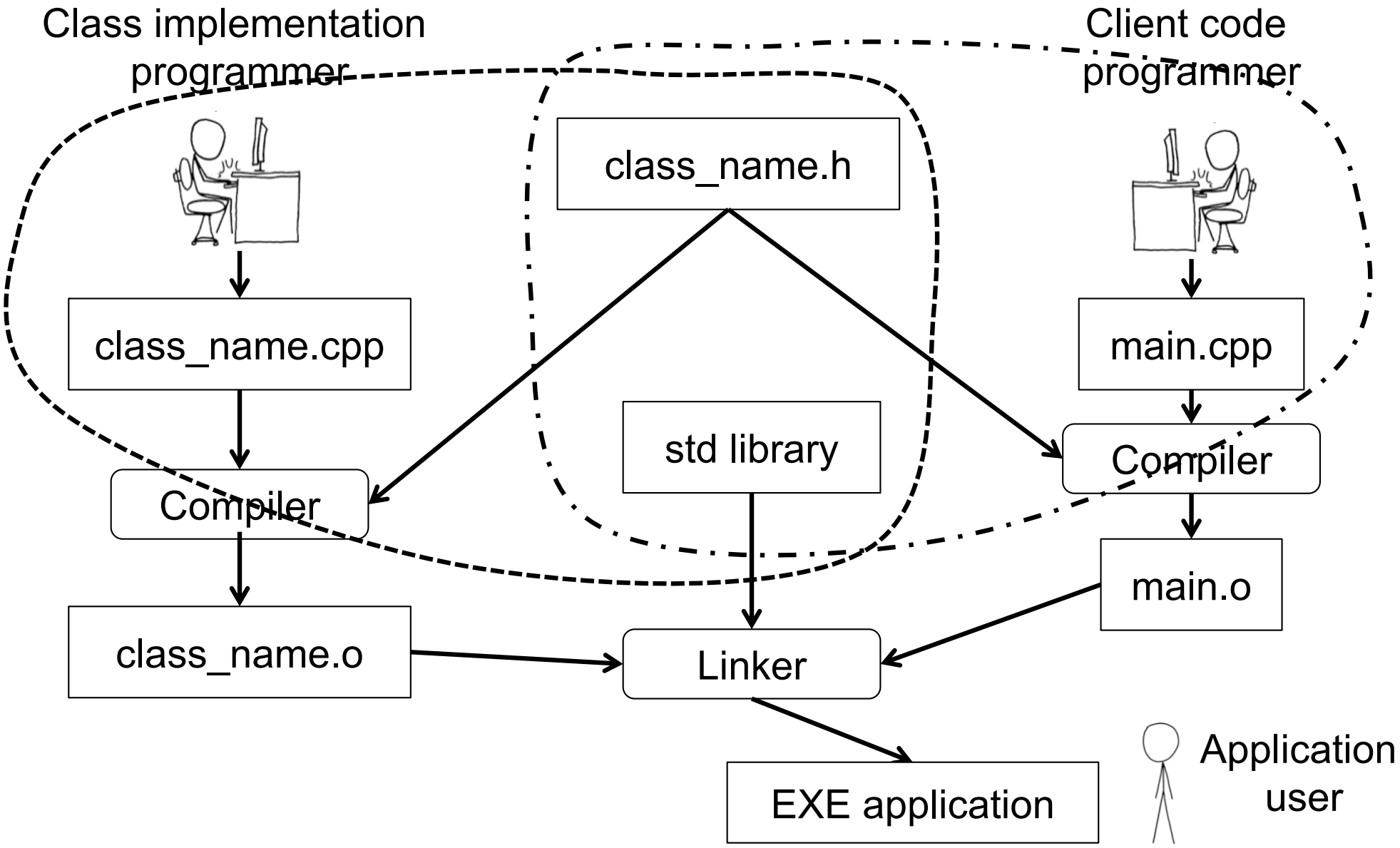
Client code programmer

main.cpp

File Organization - 2



File Organization - 2




Малые Автюхи, Калинковичский район, Республики Беларусь