Computer Programming (b) - E1124

(Spring 2021-2022)

Lecture 10

C++ Constructors

INSTRUCTOR

Dr / Ayman Soliman

➢ Contents

➢ Constructors' definition

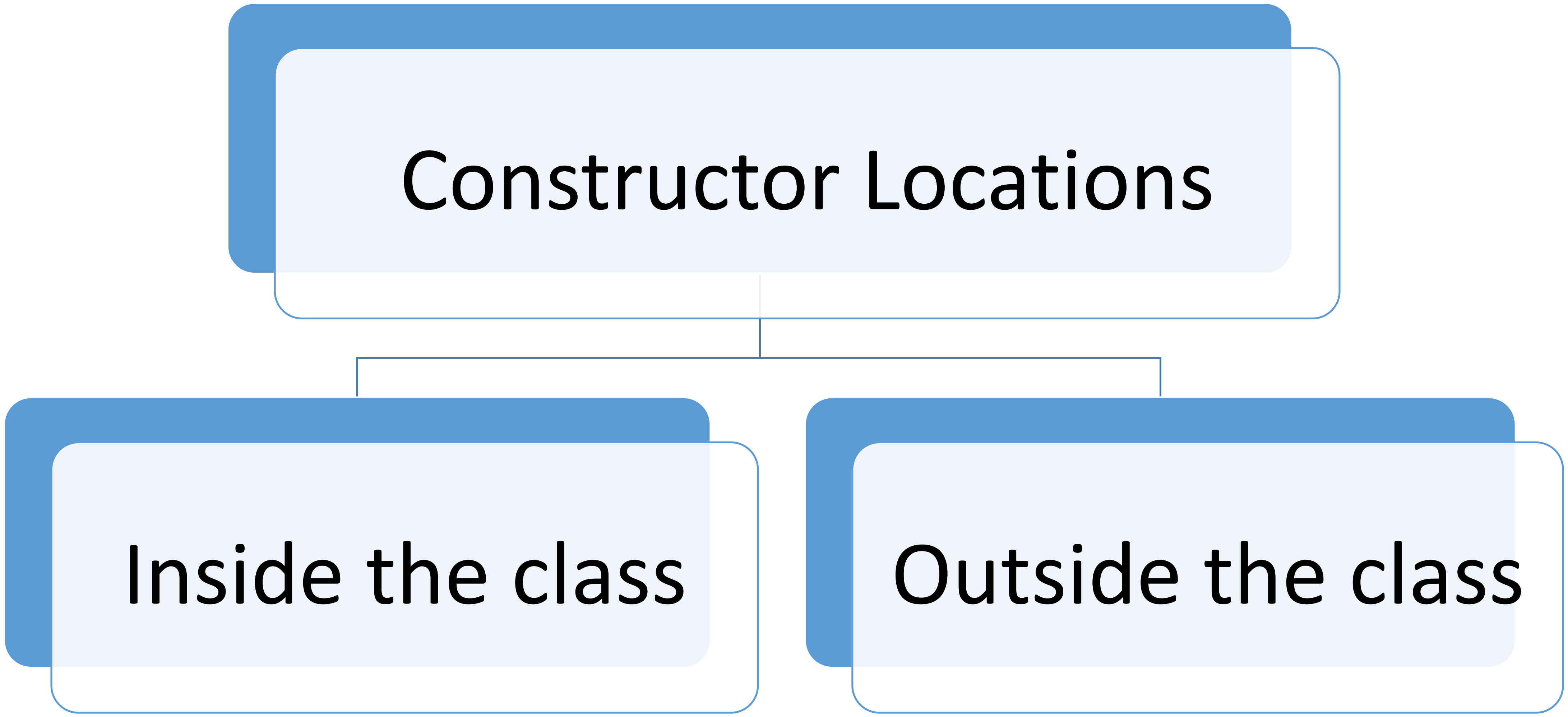
- ➢ A constructor in C++ is a special method that is automatically called when an object of a class is created.

- ➢ To create a constructor, use the same name as the class, followed by parentheses ():

- ➢ Note: The constructor has the same name as the class, it is always public, and it does not have any return value.

➢ Example

```
class MyClass {                    // The class
  public:                          // Access specifier
    MyClass() {                    // Constructor
      cout << "Hello World!";
    }
};

int main() {
  MyClass myObj;                   // Create an object of MyClass (this will call
                                      the constructor)
  return 0;
}
```


Constructor Locations
Inside the class
Outside the class

Inside the class

➢ Constructor Parameters

- ➢ Constructors can also take parameters (just like regular functions), which can be useful for setting initial values for attributes.
- ➢ The following class have brand, model and year attributes, and a constructor with different parameters.
- ➢ Inside the constructor we set the attributes equal to the constructor parameters (brand=x, etc).
- ➢ When we call the constructor (by creating an object of the class), we pass parameters to the constructor, which will set the value of the corresponding attributes to the same:

➢ Example

```
class Car {        // The class
  public:          // Access specifier
    string brand;  // Attribute
    string model;  // Attribute
    int year;      // Attribute
    Car(string x, string y, int z) {
   // Constructor with parameters
      brand = x;
      model = y;
      year = z;
    }
};
```

```
int main() {
  // Create Car objects and call the constructor with
  // different values
  Car carObj1("BMW", "X5", 1999);
  Car carObj2("Ford", "Mustang", 1969);

  // Print values
  cout << carObj1.brand << " " << carObj1.model
          << " " << carObj1.year << "\n";
  cout << carObj2.brand << " " << carObj2.model
          << " " << carObj2.year << "\n";
  return 0;
}
```

Outside the class

➢ Outside the class

- ➢ Just like functions, constructors can also be defined outside the class.

- ➢ First, declare the constructor inside the class, and then define it outside of the class by specifying the name of the class, followed by the scope resolution :: operator, followed by the name of the constructor (which is the same as the class):

➢ Example

```
class Car {        // The class
  public:           // Access specifier
    string brand;  // Attribute
    string model;  // Attribute
    int year;       // Attribute
    Car(string x, string y, int z) ;
// Constructor declaration
};
// Constructor definition outside the class
Car::Car(string x, string y, int z) {
  brand = x;
  model = y;
  year = z;
}
```

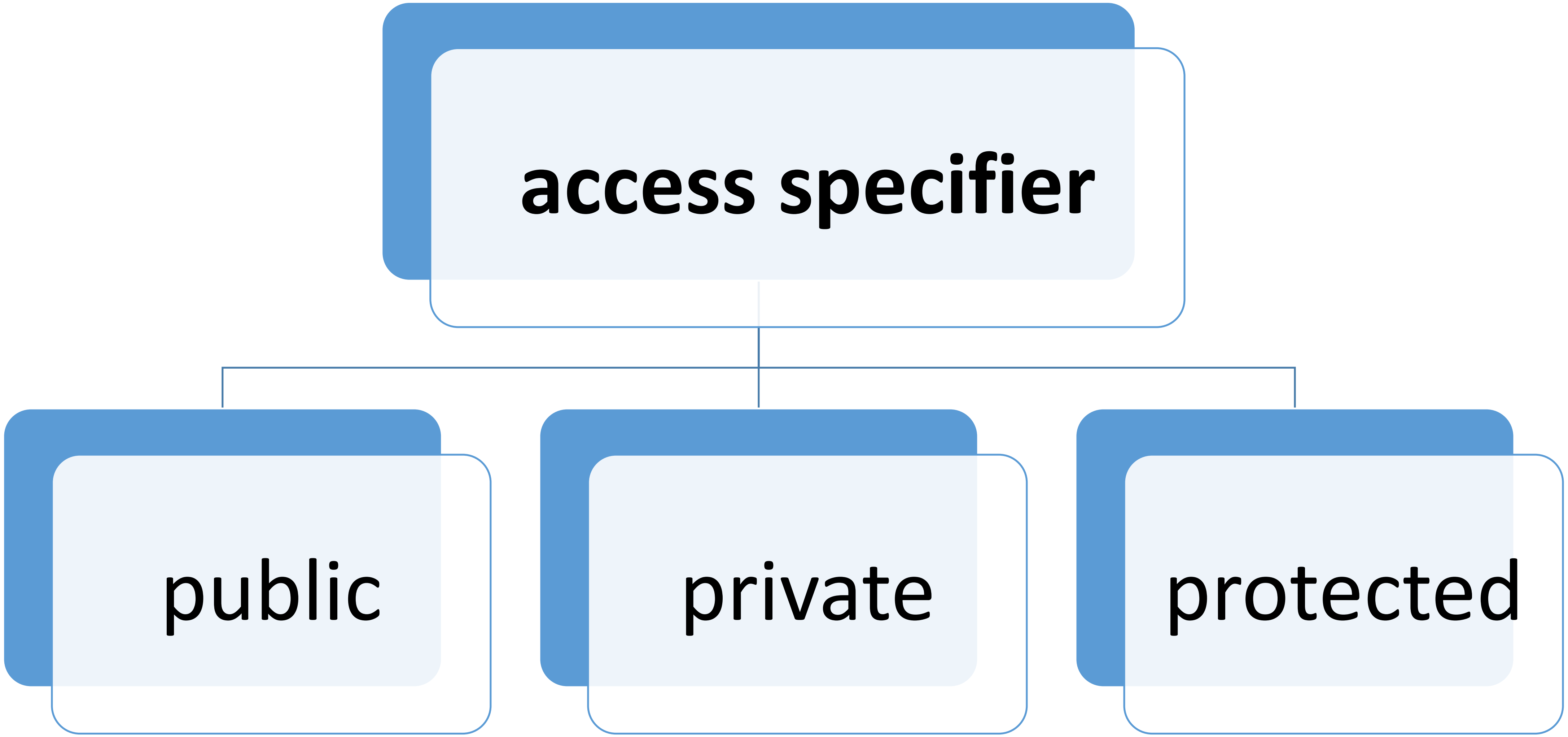
```
int main() {
  // Create Car objects and call the constructor with
  // different values
  Car carObj1("BMW", "X5", 1999);
  Car carObj2("Ford", "Mustang", 1969);

  // Print values
  cout << carObj1.brand << " " << carObj1.model
        << " " << carObj1.year << "\n";
  cout << carObj2.brand << " " << carObj2.model
        << " " << carObj2.year << "\n";
  return 0;
}
```

C++ Access Specifiers

- By now, you are quite familiar with the public keyword that appears in all our class examples:
- Example

```
class MyClass {          // The class
  public:                // Access specifier
                        // class members goes here
};
```

The public keyword is an access specifier. Access specifiers define how the members (attributes and methods) of a class can be accessed. In the example above, the members are public - which means that they can be accessed and modified from outside the code.


access specifier
public
private
protected

➢ C++ Access Specifiers

- ➢ public - members are accessible from outside the class
- ➢ private - members cannot be accessed (or viewed) from outside the class
- ➢ protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes. You will learn more about Inheritance later.

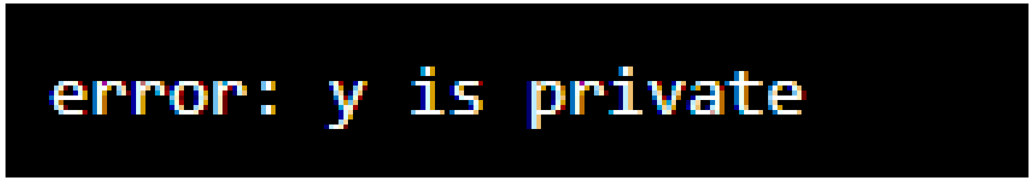
- ➢ Note: By default, all members of a class are private if you don't specify an access specifier:

Example

```
class MyClass {
  int x;                  // Private attribute
  int y;                  // Private attribute
};
```

➢ Example

```
class MyClass {
  public:                    // Public access specifier
    int x;                   // Public attribute
  private:                   // Private access specifier
    int y;                    // Private attribute
};

int main() {
  MyClass myObj;
  myObj.x = 25;          // Allowed (public)

  myObj.y = 50;          // Not allowed (private)
  return 0;
}
```



```
error: y is private
```

Thank you