Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Access Control and Introduction to Classes

Quick Recap of Relevant Topics

IIT Bombay

- Structures representing objects
  - Groups of related variables, arrays, other structures
  - Member functions as interfaces for interaction
  - Accessing members (data and functions) of structures

No restrictions on accessing members of a structure from anywhere in a program

Overview of This Lecture

IIT Bombay

- Access control of members in structures
  - private and public members
- Classes in C++ programs

Acknowledgment

- Much of this lecture is motivated by the treatment in

  An Introduction to Programming Through C++

  by Abhiram G. Ranade

  McGraw Hill Education 2014

Recap: Object-Oriented Programming Overview

- Identify **entities** or **objects** involved in the working of the system
- Think of system functionality in terms of **operations on and interactions between objects**
  - Member functions are interfaces for these operations
- **Abstract** away (hide) details of object not necessary to be exposed
  - Data hiding or encapsulation
  - More generally controlling access to information/interface of objects

Focus of this lecture

Recap: Struct V3

IIT Bombay

```
struct V3 {
    double x, y, z;
    double length() { return sqrt(x*x + y*y + z*z); }
    V3 sum (V3 const &b) {
        V3 v;
        v.x = x + b.x; v.y = y + b.y; v.z = z = b.z;  return v;
    }
    V3 scale (double const factor) {
        V3 v;
        v.x = x*factor; v.y = y*factor;  v.z = z*factor;  return v;
    }
};
```

Accessing Data Members of V3

IIT Bombay

```
int main()
{  V3 vel, acc, pos; // initial velocity, acceleration, initial position
   … Some more declarations …
   cout << “Give x, y and z components of initial velocity: “ << endl;
   cin >> vel.x >> vel.y >> vel.z;
   cout << “Give x, y and z components of acceleration: “ << endl;
   cin >> acc.x >> acc.y >> acc.z;
   … Rest of code …
}
```

Accessing Member Functions of V3

IIT Bombay

```
int main()
{  V3 vel, acc, pos; // initial velocity, acceleration, initial position
   V3 currDispl, currPos; // current displacement & position
   double t = 0.0, deltaT, totalT;  //  t: time elapsed so far
   … Reading in and validating values …
   while (t < totalT) {
      // Calculate current displacement using vel*t + (0.5)*acc*t^2
      currDispl = (vel.scale(t)).sum(acc.scale(0.5*t*t));
      currPos = currDispl.sum(pos);
      cout << "Time " << t << " ";  currPos.print();  t = t + deltaT;
   }
   return 0;
}
```

No restrictions on accessing members of a structure from anywhere in a program

Controlling Access to Members

- C++ allows three types of access control for every member (data or function)
  - **private**: Member can be accessed only from member functions of same structure
  - **public**: Member can be accessed from anywhere in program
  - **protected**: Outside scope of current discussion …
- Crucial for data hiding or encapsulation
- **private**, **public**, **protected**: C++ keywords

Controlling Access to Members

IIT Bombay

All members of a structure are public by default

```
struct V3 {
    double x, y, z;

    double length() { return sqrt(x*x + y*y + z*z); }
    V3 sum (V3 const &b) {
        V3 v;  v.x = x + b.x; v.y = y + b.y; v.z = z = b.z;  return v;
    }
    V3 scale (double const factor) {
        V3 v;  v.x = x*factor; v.y = y*factor;  v.z = z*factor;  return v;
    }
};
```

Controlling Access to Members

IIT Bombay

C++ allows access-control of groups of members

```
struct V3 {
  private:
     double x, y, z;
  public:
    double length() { return sqrt(x*x + y*y + z*z); }
    V3 sum (V3 const &b) {
       V3 v;  v.x = x + b.x; v.y = y + b.y; v.z = z = b.z;  return v;
    }
    V3 scale (double const factor) {
       V3 v;  v.x = x*factor; v.y = y*factor;  v.z = z*factor;  return v;
    }
};
```

Controlling Access to Members

IIT Bombay

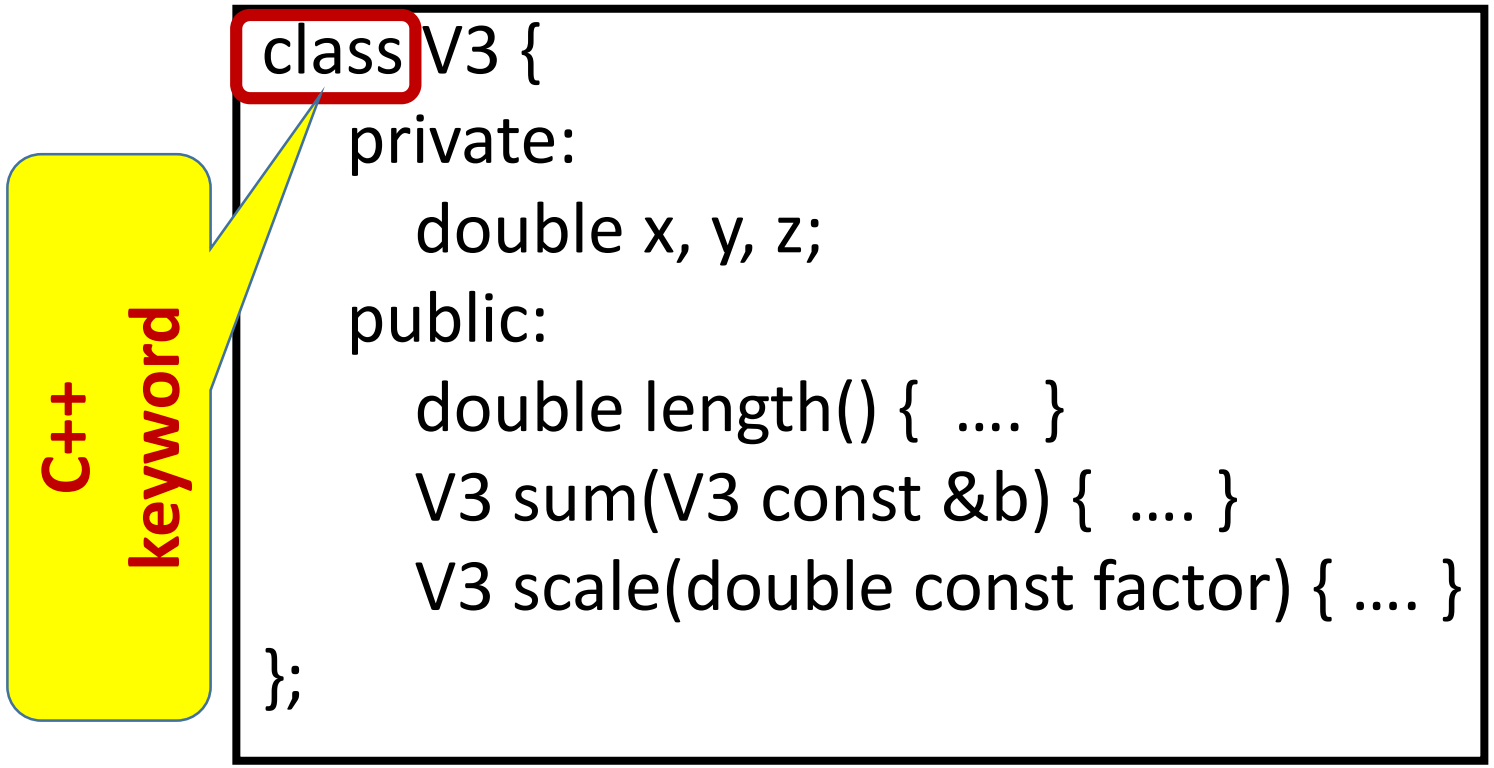
```
struct V3 {
  private:
    double x, y, z;
  public:
    V3 sum (V3 const &b) {
      V3 v;  v.x = x + b.x; v.y = y + b.y; v.z = z = b.z;  return v;
    }
    V3 scale (double const factor) {
      V3 v;  v.x = x*factor; v.y = y*factor;  v.z = z*factor;  return v;
    }
  private:
    double length() { return sqrt(x*x + y*y + z*z); }
};
```

C++ allows access-control of groups of members

Classes in C++

IIT Bombay

- A **class** is like a structure, except that all members are private by default.
  - More commonly used than structures in C++ programs



Effect of Access Control

IIT Bombay

```
int main()
{  V3 vel, acc, pos; // initial velocity, acceleration, initial position
   … Some more declarations …
   cout << "Give x, y and z components of initial velocity: " << endl;
   cin >> vel.x >> vel.y >> vel.z;
   cout << "Give x, y and z components of acceleration: " << endl;
   cin >> acc.x >> acc.y >> acc.z;
   … Rest of code …
}
```

So How Do We Read/Write Data Members of V3?

- Make all data members public
  - Not preferred, defeats purpose of data encapsulation
  - Breaks modularity of code by exposing internal details
    - E.g., we chose Cartesian coordinates to represent 3-D vectors
    - What if we later decide to use cylindrical coordinates ?

```
class V3 {
   public:
    //  double x, y, z;
        double rho, phi, z;
    … Member functions …
};
```

```
int main() {
  V3 vel, acc, pos;
  // cin >> vel.x >> vel.y >> vel.z;
   cin >> vel.rho >> vel.phi >> vel.z;
  … Rest of code …
}
```

Should All Data Members Always Be Private?

IIT Bombay

- Not necessarily
- Expose and allow access to only those members that other functions need access to
- Hide and prevent access to

  book-keeping data members, implementation-specific data members, internal state recording data members, …
- Choice of what should be private/public affects quality and modularity of code
  - Relevant for data members and member functions
  - Careful thought process essential – more of an art!

So How Do We Read/Write Data Members of V3?

IIT Bombay

Accessor functions

Member functions that return values of only those data members that other functions are allowed to read

```
class V3 {
  private: double x, y, z;
  public:
      double getX() {return x;}
      double getY() {return y;}
      double getZ() {return z;}
      … Other member functions …
};
```

So How Do We Read/Write Data Members of V3?

Mutator functions

Member functions that update values of data members that other functions are allowed to update

```
class V3 {
  private: double x, y, z;
  public:
      void setXYZ(double vx, double vy, double vz)
        { x = vx; y = vy; z = vz; return;}
      … Other member functions …
};
```

So How Do We Read/Write Data Members of V3?

IIT Bombay

- Changing internal representation of a class requires changing only accessor/mutator function definitions

```
class V3 {
  private: double rho, phi, z;
  public:
      double getX() {return (rho* cos(phi));}
      double getY() {return (rho* sin(phi));}
      double getZ() {return z;}
      void setXYZ(double vx, double vy, double vz)
      { rho = sqrt(vx*vx + vy*vy); phi = arctan(vy/vx); z = vz; return;}
      … Other member functions …
};
```

Summary

- Controlling access to members in structures through “public” and “private”
- A brief introduction to C++ classes

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Constructor and Destructor Functions

Quick Recap of Relevant Topics

IIT Bombay

- Structures and classes
- Data members and member functions
- Accessing members
- Access control of members
  - public and private members

Overview of This Lecture

IIT Bombay

- Special member functions
  - Constructor functions
  - Destructor functions

Acknowledgment

- Much of this lecture is motivated by the treatment in

  An Introduction to Programming Through C++

  by Abhiram G. Ranade

  McGraw Hill Education 2014

Recap: Member Functions and Their Usage

IIT Bombay

- A class can have public or private member functions

```
class V3 {
  // 3-dimensional vector with printLength()
  private: double x, y, z;
  public:
    … Other member functions …
    void printLength() {
       cout << length() << endl; return;
    }
  private:
    double length() {return sqrt(x*x + y*y + z*z);}
};
```

```
int main() {
  V3 a, * ptr;
  … Some code here …
  a.printLength();
  ptr = new V3;
  if (ptr == NULL) return -1;
  … Some code here …
  ptr->printLength();
  delete ptr;
  return 0;
}
```

Two Special Member Functions of Every Class

IIT Bombay

- **Constructor:** Invoked **automatically** when an object of the class is allocated
  - Convenient way to initialize data members
  - Just like any other member function
    - Accepts optional input parameters
    - Can be used to perform tasks other than initialization too
- **Destructor:** Invoked **automatically** when an object of the class is de-allocated
  - Convenient way to do book-keeping/cleaning-up before de-allocating object
  - Accepts no parameters
  - Can be used to perform other tasks before de-allocating object

Example Constructor of Class V3

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
  public:
    V3 (double vx, double vy, double vz) {
       x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
      … Other member functions of V3 …
};
```

Constructor of class V3

Example Constructor of Class V3

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
  public:
    V3 (double vx, double vy, double vz) {
       x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
      … Other member functions of V3 …
};
```

Constructor of class V3

- A member function
- No return type
- Same name as that of class (i.e. V3)
- Optional input parameters
- Mostly used for initialization

Example Constructor of Class V3

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
  public:
    V3 (double vx, double vy, double vz) {
      x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
      … Other member functions of V3 …
};
```

Another constructor of class V3

Typed list of parameters different from that of the previous constructor

Multiple Constructors of Same Class

IIT Bombay

- A class can have multiple constructors as long as each one has a distinct list of parameter types
  - V3 (double vx, double vy, double vz) and V3()
- When allocating an object of the class, the types of parameters passed to the constructor determine which constructor is invoked
  - V3 myObj1; V3 *myObj2 = new V3(1.0, 2.0. 3.0);
- Allocated object serves as the receiver object for the constructor call

Usage of Constructors

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
  public:
    V3 (double vx, double vy, double vz) {
      x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
   … Other member functions of V3 …
};
```

Note the "public" declaration

```
int main() {
  V3 a (0.0, 0.0, 0.0);
  V3 b;
  V3 *p, *q;
   … Some code here …
  p = new V3 (1.0, 2.0, 3.0);
  q = new V3;
   … Some code here …
  delete p; delete q;
  return 0;
}
```

Usage of Constructors

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
  public:
    V3 (double vx, double vy, double vz) {
      x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
    … Other member functions of V3 …
};
```

Note the "public" declaration

```
int main() {
  V3 a (0.0, 0.0, 0.0);
  V3 b;
  V3 *p, *q;
   … Some code here …
  p = new V3 (1.0, 2.0, 3.0);
  q = new V3;
   … Some code here …
  delete p; delete q;
  return 0;
}
```

Usage of Constructors

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
  public:
    V3 (double vx, double vy, double vz) {
      x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
   … Other member functions of V3 …
};
```

```
int main() {
  V3 a (0.0, 0.0, 0.0);
  V3 b;
  V3 *p, *q;
   … Some code here …
  p = new V3 (1.0, 2.0, 3.0);
  q = new V3;
   … Some code here …
  delete p; delete q;
  return 0;
}
```

Usage of Constructors

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
  public:
    V3 (double vx, double vy, double vz) {
       x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
    … Other member functions of V3 …
};
```

```
int main() {
  V3 a (0.0, 0.0, 0.0);
  V3 b;
  V3 *p, *q;
   … Some code here …
  p = new V3 (1.0, 2.0, 3.0);
  q = new V3;
   … Some code here …
  delete p; delete q;
  return 0;
}
```

Two Special Member Functions of Every Class

- **Constructor:** Invoked **automatically** when an object of the class is allocated
  - Convenient way to initialize data members
  - Just like any other member function
    - Accepts optional input parameters
    - Can be used to perform tasks other than initialization
- **Destructor:** Invoked **automatically** when an object of the class is de-allocated
  - Convenient way to do book-keeping/cleaning-up before de-allocating object
  - Accepts no parameters
  - Can be used to perform other tasks before de-allocating object

Example Destructor of Class V3

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
    double length() { … }
  public:
    … Constructors of class V3 …
    ~V3() { if (length() == 0.0)
                {cout << "Zero vector!!! " << endl;}
              return;
      }
    … Other member functions of class V3 …
};
```

Destructor of class V3

- A member function
- No return type
- Name: ~ followed by name of class (i.e. ~V3)
- No input parameters
- Mostly used for book-keeping/clean-up before de-allocation of objects

Multiple destructors of same class not allowed in C++

Example Destructor of Class V3

IIT Bombay

```
class V3 {
  private:
    double x, y, z;
    double length() { ... }
  public:
    ... Constructors of class V3 ...
    ~V3() { if (length() == 0.0)
              {cout << "Zero vector!!! " << endl;}
            return;
     }
   ... Other member functions of class V3 ...
};
```

Note the "public" declaration

```
int main() {
  V3 a (1.0, 2.0, 3.0);
  { V3 b;
    a = b;
  }
  V3 *p =
       new V3(1.0, 1.0, 1.0);
  a = *p;
  delete p;
  return 0;
}
```

Summary

IIT Bombay

- Constructor and destructor functions of classes
- Simple usage of above special member functions
  - More complex usage coming later …

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: More on Constructors

Quick Recap of Relevant Topics

- Object-oriented programming with structures and classes
- Data members and member functions
- Accessing members and controlling access to members
- Constructor and destructor functions

Overview of This Lecture

- Closer look at constructors
- Example usage in C++ programs

Acknowledgment

- Much of this lecture is motivated by the treatment in

  An Introduction to Programming Through C++

  by Abhiram G. Ranade

  McGraw Hill Education 2014

- Examples taken from this book are indicated in slides by the citation **AGRBook**

Recap: Constructor and Destructor Functions

- **Constructor:** Invoked **automatically** when an object of the class is allocated
  - Object is allocated first, then constructor is invoked on object
  - Convenient way to initialize data members
- **Destructor:** Invoked **automatically** when an object of the class is de-allocated
  - Destructor is invoked on object first, then object is de-allocated
  - Convenient way to do book-keeping/cleaning-up before de-allocating object

Recap: Constructors/Destructor of Class V3

IIT Bombay

```
class V3 { private: double x, y, z;
                    double length() { … }
  public:
    V3 (double vx, double vy, double vz) {
       x = vx; y = vy; z = vz;  return;
    }
    V3 () { x = y = z = 0.0;  return; }
    ~V3() { if (length() == 0.0) {
                cout << "Zero vector!!!";
              return;
     }
    … Other member functions of V3 …
};
```

Constructor functions

Destructor function

Recap: Invoking Member Functions

IIT Bombay

- Member functions of a class can be usually invoked explicitly on a receiver object of the same class

```
class V3 {
  private: double x, y, z;
                double length() { … }
  public:
    V3 scale(double const factor) { … }
    void printLength() { … }
    … Other member functions …
};
```

```
int main() {
   V3 a (1.0, 2.0, 3.0);
   V3 b = a.scale(4.0);
    b.printLength();
    return 0;
}
```

Recap: Invoking Member Functions

IIT Bombay

- Member functions of a class can be usually invoked explicitly on a receiver object of the same class

```
class V3 {
  private: double x, y, z;
           double length() { … }
  public:
   V3 scale(double const factor) { … }
   void printLength() { … }
   … Other member functions …
```

```
int main() {
   V3 a (1.0, 2.0, 3.0);
   V3 b = a.scale(4.0);
    b.printLength();
    return 0;
}
```

Can we do the same with constructors and destructors?

Invoking Constructors/Destructors Explicitly

IIT Bombay

- Usually **an error** to call a destructor explicitly in a program
- **OK** to call a constructor explicitly in a program

```
int main() {
        V3 a(1.0, 1.0, 1.0);
        V3 b = a.sum( V3(2.0, 2.0, 2.0) );
        a.printLength(); b.printLength();
        return 0;
 }
```

Explicit constructor invokation

Looks like normal function call

Name of (constructor) function is name of class

Creates temporary object and invokes constructor on it

Invoking Constructors Explicitly [Ref. AGRBook]

IIT Bombay

- An interesting implementation of “sum” in class V3

```
class V3 {
   private: double x, y, z;
   public:
    … Other members, constructors, destructor …
    V3 sum (V3 const &b) {
        return V3(x+b.x, y+b.y, z+b.z);
    }
};
```

Specifying Default Parameter Values

IIT Bombay

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of the following lead to correct constructor calls

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Specifying Default Parameter Values

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of th

Equivalent to V3 a(0.0, 1.0, 2.0);

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Specifying Default Parameter Values

IIT Bombay

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

Equivalent to V3 b(1.2, 1.0, 2.0);

all of th

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Specifying Default Parameter Values

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of th

Equivalent to V3 c(1.2, 1.3, 2.0);

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Specifying Default Parameter Values

C++ allows default values to be specified for parameters of constructors (and also for other member functions)

With constructor definition

```
V3(double vx = 0.0, double vy = 1.0,  double vz = 2.0) {
   x = vx; y = vy; z = vz; return;
}
```

all of th

Equivalent to V3 d(1.2, 1.3, 1.4);

```
V3 a;  V3 b (1.2);  V3 c (1.2, 1.3);  V3 d (1.2, 1.3, 1.4);
```

Initialization Lists

IIT Bombay

Specifies how different data members of the receiver object are initialized before execution of the constructor begins

```
class V3 {
  private: double x, y, z;
  public:
    V3(double vx, double vy, double vz) : x(vx), y(vy), z(vz) {
       … Rest of constructor code comes here …
    }
    … Other member functions of class V3 …
};
```

Note the :

Initialization List

Initialization Lists

IIT Bombay

Specifies how different data members of the receiver object are initialized before execution of the constructor begins

```
class V3 {
  private: double x, y, z;
  public:
    V3(double vx, double vy, double vz) : x(vx), y(vy), z(vz) {
       … Rest of constructor code comes here …
    }
    … Other member functions of class V3
};
```

Note the :

Initialization List

Can be any expression in vx, vy, vz

An Interesting Example [Ref. AGRBook]

IIT Bombay

```
class Point { private: double x, y;
                public: Point (double p, double q) {x = p; y = q; return;}
};

class Disk { private: Point center; double radius;
                public: Disk(double x, double y, double r) :
                            center(Point(x, y)), radius(r) { return; }
};
```

Summary

IIT Bombay

- Closer look at constructor functions
  - Invoking constructors explicitly
  - Specifying default values of parameters
  - Initialization lists

Bài tập đề xuất

+ Bài 1: Class
+ Bài 2: Box it