

C++ OOP

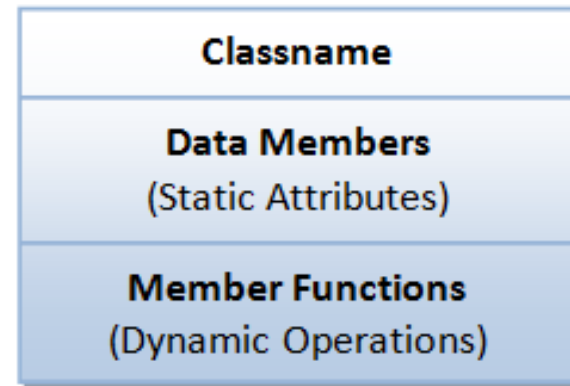
ครั้งที่ 2

ทบทวน

ความรู้เรื่อง คลาส วัตถุ แอททริบิวต์และเมธอด

คลาส (Class)

- โครงสร้างของ Class ประกอบด้วย
 - Class name
 - Data members (attributes)
 - Member functions (methods/operations)



A class is a 3-compartment box encapsulating data and functions

Classname
(Identifier)

Data Member
(Static attributes)

Member Functions
(Dynamic Operations)

Student	Circle
name grade	radius color
getName() printGrade()	getRadius() getArea()

SoccerPlayer	Car
name number xLocation yLocation	plateNumber xLocation yLocation speed
run() jump() kickBall()	move() park() accelerate()

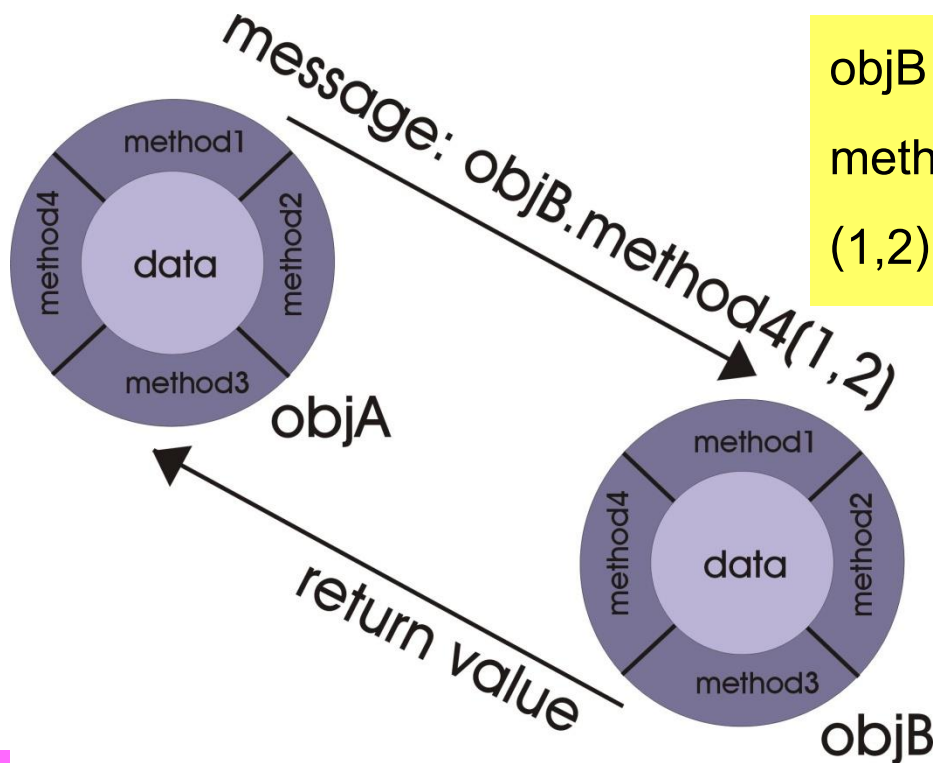
Examples of classes

Classname	<u>paul:Student</u>	<u>peter:Student</u>
Data Members	name="Paul Lee" grade=3.5	name="Peter Tan" grade=3.9
Member Functions	getName() printGrade()	getName() printGrade()

Two instances of the **Student** class

การสื่อสารระหว่างวัตถุ

- การสื่อสารระหว่างกันของวัตถุทำได้โดยการผ่านข่าวสาร (message)



objB = ชื่อวัตถุ
method4 = ชื่อเมธอด
(1,2) = อาร์กิวเมนต์(argument)

Access Modifier ในภาษาจาวา

Modifier	คำอธิบาย
public (สาธารณะ)	คลาสอื่นๆ สามารถเข้าใช้งานแอททริบิวต์ และเมธอด ที่ถูกกำหนด public ได้อย่างอิสระ
protected (ถูกปกป้อง)	ฟังก์ชันที่ประกาศภายใน Class และ Sub-class เท่านั้นที่สามารถอ้างถึงหรือเรียกใช้ข้อมูลหรือฟังก์ชันในส่วน protected ได้
private (ส่วนบุคคล)	ปิดกั้นไม่ให้คลาสอื่นๆ สามารถเข้าใช้งาน แอททริบิวต์ และเมธอด ได้ยกเว้นคลาสของตัวเอง

การแทนค่าคลาส, แอททริบิวต์ และเมธอดโดยใช้ UML

คลาส

1.class Employee {

2. public:

3. char name[50];

4. char id[10];

5. float salary;

6. void show_employee(float salary) {

7. cout <<"Name: " <<name<<endl;

8. cout <<"Id: " <<id<<endl;

9. cout <<"Salary: " <<salary<<endl;

10. };

11.};

12.Employee worker, boss;

แอททริบิวต์

เมธอด

วัตถุ

การแทนค่าคลาส,แอททริบิวท์ และเมธอด โดยใช้ UML

```
1.class Employee {  
2.    public:  
3.        char name[50];  
4.        char id[10];  
5.        float salary;  
6.        void show_employee(float salary) {  
7.            cout <<"Name: " <<name<<endl;  
8.            cout <<"Id: " <<id<<endl;  
9.            cout <<"Salary: " <<salary<<endl;  
10.        };  
11.};  
12.Employee worker, boss;
```

Employee

+name[50]: char

+id[10]: char

+salary : float

+ void show_employee(salary : float)

หัวข้อ

- ขอบเขตของตัวแปรในภาษา C++
(Scope resolution operator)
- Constructor และ Destructor
- Constructor และ Overloading
- Static เมธอดและแอททริบิวท์

ขอบเขตของตัวแปรในภาษา C++

(Scope resolution operator)

- ภาษา C++ สามารถเรียกใช้งานตัวแปรทั้งแบบ local variable และ global variable
- ถ้ามีการตั้งชื่อตัวแปรซ้ำกัน และต้องการเรียกใช้งานค่า global variable สามารถทำได้โดยใช้เครื่องหมาย :: (Scope resolution operator)

- รูปแบบ **:: ชื่อของ** global variable

<https://www.programmingsimplified.com/cpp/source-code/scope-resolution-operator>

<https://www.thaicyberpoint.com/ford/blog/id/147/>

```

1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
4. class Student {
5.     public:
6.         float score;
7.         void showStudent() {
8.             cout << fixed << setprecision(3);
9.             score = 90;
10.            cout << "Score-class = " << score << endl;
11.        };
12. };
13. float score = 100.00;
14. main( ) {
15.     Student stu1;
16.     float score = 80;
17.     cout << "Score-local = " << score << endl;
18.     stu1.showStudent();
19.     cout << "Score-local = " << score << endl;
20.     cout << "Score-global = " << ::score << endl;
21. }

```

E:\Jirawan drive\Teacher\สอน สอน สอน 256

```

Score-local = 80
Score-class = 90.000
Score-local = 80.000
Score-gloabal = 100.000

```

ขอบเขตของตัวแปรในภาษา C++

(Scope resolution operator)

- สามารถใช้ “::” (global resolution operator) เพื่อสร้างเมธอดไว้ภายนอก Class ด้วย
- แต่ทั้งนี้ต้องมีการประกาศชื่อเมธอดที่ต้องการสร้างเพิ่มไว้ที่คลาสิก่อนแล้ว

- รูปแบบ

```
return_type ชื่อคลาส::ชื่อเมธอด  
([อาร์กิวเมนต์]){  
    //คำสั่งที่ใช้ในเมธอด  
    [return];  
};
```

```

1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
4. class Student {
5.     public:
6.         float score;
7.         void showStudent(float score) {
8.             cout <<"Score = "<<score<<endl;
9.         };
10.        void showNameGPA(char name[10],float gpa);
11. };
12. void Student::showNameGPA(char name[10],float gpa){
13.     cout <<"Name = "<<name<<endl;
14.     cout <<"GPA = "<<gpa<<endl;
15. }
16. main( ) {
17.     Student stu1;
18.     cout <<fixed<<setprecision(3);
19.     stu1.showNameGPA("Jirawan",4.00);
20.     stu1.showStudent(90);
21. }

```



E:\Jirawan drive\Teache

```

Name = Jirawan
GPA = 4.000
Score = 90.000

```

Constructor และ Destructor

- Constructor เป็น Class method ที่มีชื่อเหมือนกับ Class และจะ ถูกเรียกอัตโนมัติเมื่อมีการสร้าง Object
- จะใช้ Constructor เพื่อกำหนดค่าเริ่มต้นให้กับ attribute
- มีการใช้ “::” (global resolution operator) เพื่อสร้างเมธอดไว้ภายนอก Class ด้วย
- ถ้าไม่มีกำหนดหรือสร้าง Constructor ของคลาส Compiler จะให้ Constructor ที่เรียกว่า Default constructor

Constructor

- ตัวอย่าง Constructor ไม่มี argument
- https://www.tutorialspoint.com/cplusplus/cpp_constructor_destructor.htm

```

1. #include <iostream>
2. using namespace std;
3. class Line {
4.     public:
5.         void setLength( double len );
6.         double getLength( void );
7.         Line(); // This is the constructor
8.     private:
9.         double length;
10. };
11.
12. // Member functions definitions including constructor
13. Line::Line(void) {
14.     cout << "Object is being created" << endl;
15. }
16. void Line::setLength( double len ) {
17.     length = len;
18. }
19. double Line::getLength( void ) {
20.     return length;
21. }
22. main() {
23.     Line line;
24.     // set line length
25.     line.setLength(5.0);
26.     cout << "Length of line : " << line.getLength()
27.         << endl;
28. }

```

 E:\Jirawan drive\Teacher\สอน สอน สอน 2561\

```

Object is being created
Length of line : 5

```


Constructor

- ตัวอย่าง Constructor ที่มี argument
- https://www.tutorialspoint.com/cplusplus/cpp_constructor_destructor.htm

```

1. #include <iostream>
2. using namespace std;
3. class Line {
4.     public:
5.         void setLength( double len );
6.         double getLength( void );
7.         Line(double len); //constructor
8.     private:
9.         double length;
10. };
11.
12. // Member functions definitions include
13. Line::Line( double len) {
14.     length = len;
15.     cout << "Object is being created, length = " << len << endl;
16. }
17. void Line::setLength( double len ) {
18.     length = len;
19. }
20. double Line::getLength( void ) {
21.     return length;
22. }
23. // Main function for the program
24. main() {
25.     Line line(10.0);
26.
27.     // get initially set length.
28.     cout << "Length of line : " << line.getLength() << endl;
29.
30.     // set line length again
31.     line.setLength(5.0);
32.     cout << "Length of line : " << line.getLength() << endl;
33. }

```

E:\Jirawan drive\Teacher\now now now 2561\lecture-12-c++ const

```

Object is being created, length = 10
Length of line : 10
Length of line : 5

```

```

1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
4. class Student {
5.     public:
6.         float score,sMid,sFi,sLab;
7.         Student(float midterm, float final, float lab);
8.         void showStudent() {
9.             float tScore = sMid+sFi+sLab;
10.             cout <<"Score = "<<tScore<<endl;
11.         };
12. };
13. Student::Student(float midterm, float final, float lab){
14.     sMid = midterm ;
15.     sFi=final ;
16.     sLab=lab ;
17. }
18. main( ) {
19.     Student stu1(30,30,10),stu2(40,40,10);
20.     cout <<fixed<<setprecision(3);
21.     stu1.showStudent();
22.     stu2.showStudent();
23. }

```

E:\Jirawan drive\Teacher\สอน

```

Score = 70.000
Score = 90.000

```

Constructor

- จะเห็นว่า Constructor ไม่มีการ return ค่าและไม่ต้องใส่ void ไว้หน้า Constructor

คำสั่ง Constructor ของคลาส Student

```
Student::Student(float midterm, float final, float lab){  
    sMid = midterm ;  
    sFi=final ;  
    sLab=lab ;  
}
```

เรียกใช้

```
Student stu1(30,30,10),stu2(40,40,10);
```

Constructor

- การสร้าง Object ของ Class Student ในชื่อ stu1 และ stu2 พร้อมกำหนดค่าเริ่มต้นให้กับ attribute ชื่อ sMid,sFi,sLab ของ method ชื่อ showStudent แบบอัตโนมัติ

```
void showStudent() {  
    float tScore = sMid+sFi+sLab;  
    cout <<"Score = "<<tScore<<endl;  
};
```

- Constructor ของ Class Student ถูกเรียกอัตโนมัติ

```
1. #include <iostream>
```

```
2. #include <iomanip>
```

```
3. using namespace std;
```

```
4. class Student {
```

```
5.     public:
```

```
6.         float score,sMid,sFi,sLab;
```

```
7.         Student(float midterm=20, float final=20, float lab=10);
```

```
8.         void showStudent() {
```

```
9.             float tScore = sMid+sFi+sLab;
```

```
10.             cout <<"Score = "<<tScore<<endl;
```

```
11.         };
```

```
12. };
```

```
13. Student::Student(float midterm, float final, float lab){
```

```
14.     sMid = midterm ;
```

```
15.     sFi=final ;
```

```
16.     sLab=lab ;
```

```
17. }
```

```
18. main( ) {
```

```
19.     Student stu1(10,20,30),stu2;
```

```
20.     Student stu3(10,10);
```

```
21.     Student stu4(50);
```

```
22.     cout <<fixed<<setprecision(3);
```

E:\Jirawan drive\Teacher\sta

```
Score = 60.000  
Score = 50.000  
Score = 30.000  
Score = 80.000
```

```
23.     stu1.showStudent();
```

```
24.     stu2.showStudent();
```

```
25.     stu3.showStudent();
```

```
26.     stu4.showStudent();
```

```
27. }
```

Constructor

- การทดสอบการส่งค่าของ argument ใน constructor ของคลาส Student
 - Student stu1(10,20,30),stu2;
 - Student stu3(10,10);
 - Student stu4(50);
- <https://preecha11th.wordpress.com/2011/11/21/constructor-%E0%B9%81%E0%B8%A5%E0%B8%B0-destructor-%E0%B9%83%E0%B8%99-c/>

Constructor และ Overloading

- การ Overloading constructor คือ การที่สร้าง Constructor หลายอันโดยกำหนดให้ Constructor แต่ละอัน มี argument แตกต่างกันไป เช่น

```
1. Student(float midterm, float final, float lab){  
2.     sMid = midterm ;  
3.     sFi=final ;  
4.     sLab=lab ;  
5. }
```

```
Student stu1(10,20,30);
```

```
1. Student(float midterm, float final){  
2.     sMid = midterm ;  
3.     sFi=final ;  
4. }
```

```
Student stu2(10,10);
```



```

1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
4. class Student {
5.     public:
6.         float score,sMid,sFi,sLab;
7.         Student(float midterm, float final, float lab){
8.             sMid = midterm ;
9.             sFi=final ;
10.            sLab=lab ; }
11.        Student(float midterm, float final){
12.            sMid = midterm ;
13.            sFi=final ;}
14.        void showStudent() {
15.            float tScore = sMid+sFi+sLab;
16.            cout <<"Score = "<<tScore<<endl;    };
17. };

```

```

18. main( ) {
19.     Student stu1(10,20,30);
20.     Student stu2(10,10);
21.     cout <<fixed<<setprecision(3);
22.
23.     stu1.showStudent();
24.     stu2.showStudent();
25. }

```

Default Copy Constructor

```
....  
int main()  
{  
    Area A1, A2(2, 1);  
  
    // Copies the content of A2 to A3  
    Area A3(A2);  
    OR,  
    Area A3 = A2;  
}
```

<https://www.programiz.com/cpp-programming/constructors>

```
Score = 60.000  
Score = 20.000  
Score = 60.000  
Score = 20.000
```

```
18. main( ) {  
19.     Student stu1(10,20,30);  
20.     Student stu2(10,10);  
21.     cout <<fixed<<setprecision(3);  
22.     Student stu3(stu1);  
23.     Student stu4 = stu2;  
24.  
25.     stu1.showStudent();  
26.     stu2.showStudent();  
27.     stu3.showStudent();  
28.     stu4.showStudent();  
29. }
```

Destructor

- Destructor function จะมีไว้สำหรับการทำลาย(free up/clean) Object และจะถูกเรียกโดยอัตโนมัติเมื่อมีการทำลาย Object
- Destructur function จะมีชื่อเหมือน Class แต่มีเครื่องหมาย ~(tilde) นำหน้า
- ตัวอย่าง Destructor

```
~class_name(void){  
  
    //function statement  
  
}
```

Constructor และ Destructor

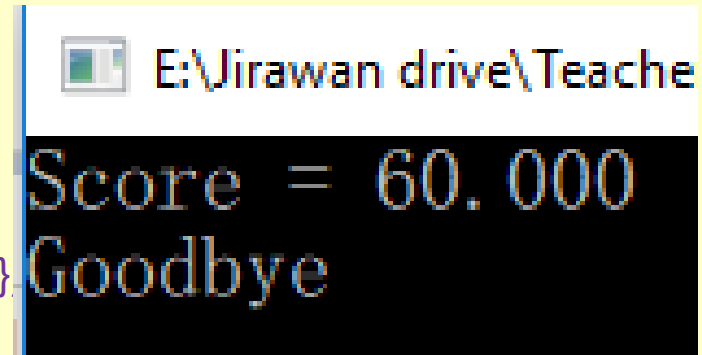
```
Student::~~Student(void){  
    cout<< "Goodbye" <<endl;  
}
```

หรือ

```
~Student(void){  
    cout<< "Goodbye" <<endl;  
}
```

- เมื่อ run โปรแกรมตอนจบ(ออกจาก main function) Destructor จะถูกเรียก
- โดยทั่วไปถ้าไม่มีการ Dynamic allocate memory ใน Object ก็ไม่ต้องมี Code เพื่อปลดปล่อย(release) Memory

```
1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
4. class Student {
5.     public:
6.         float score,sMid,sFi,sLab;
7.         Student(float midterm, float final, float lab){
8.             sMid = midterm ;
9.             sFi=final ;      sLab=lab ; }
10.        void showStudent() {
11.            float tScore = sMid+sFi+sLab;
12.            cout <<"Score = "<<tScore<<endl; }
13.        ~Student(){
14.            cout<< "Goodbye" <<endl; }
15. };
16. main( ) {
17.     Student stu1(10,20,30);
18.     cout <<fixed<<setprecision(3);
19.     stu1.showStudent();
20.
21. }
```



```
E:\Jirawan drive\Teache
Score = 60.000
Goodbye
```

Static เมธอดและแอททริบิวต์

- โดยปกติเมื่อสร้าง Object Object แต่ละตัวที่ถูกสร้างขึ้นก็จะมีค่าหรือสถานะเก็บใน attribute เป็นของตัวเอง ในบางครั้ง
- ถ้าต้องการให้มีการใช้ค่าร่วมกัน(share)ในระหว่าง Object สามารถทำได้โดยการเพิ่มคำว่า static ไว้หน้า type ของ Data member

private:

```
static int shared_valued;
```

- และเมื่อต้องการอ้างถึงเพื่อกำหนดค่า สามารถทำได้โดย

```
int class_name::shared_valued = ค่าที่กำหนด;
```

Static function และ data members

ประกาศ static attribute

```
public: static int objectCount;
```

กำหนดค่าเริ่มต้นจาก class Box

```
int Box::objectCount = 0;
```

เรียกใช้งาน static attribute

```
cout << "Total objects: " << Box::objectCount << endl;
```

```
1. #include <iostream>
2. using namespace std;
3. class Box {
4.     public:
5.         static int objectCount;
6.
7.         // Constructor definition
8.         Box(float l = 2.0, float b = 2.0, float h = 2.0) {
9.             cout <<"Constructor called." << endl;
10.            length = l;
11.            breadth = b;
12.            height = h;
13.
14.            // Increase every time object is created
15.            objectCount++;
16.        }
17.        float Volume() {
18.            return length * breadth * height;
19.        }
```

https://www.tutorialspoint.com/cplusplus/cpp_static_members.htm


```

20. private:
21.     float length;    // Length of a box
22.     float breadth;   // Breadth of a box
23.     float height;    // Height of a box
24. };

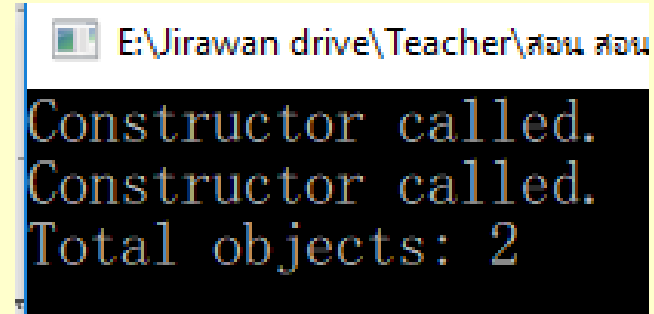
25. // Initialize static member of class Box
26. int Box::objectCount = 0;

27. main(void) {
28.     Box Box1(3.3, 1.2, 1.5);    // Declare box1
29.     Box Box2(8.5, 6.0, 2.0);    // Declare box2

30.     // Print total number of objects.
31.     cout << "Total objects: " << Box::objectCount << endl;

32. }

```



```

E:\Jirawan drive\Teacher\สอนสอน
Constructor called.
Constructor called.
Total objects: 2

```

Static function และ data members

ประกาศ static attribute

```
public: static int objectCount;
```

ประกาศ static method

```
static int getCount() {  
    return objectCount;  
}
```

กำหนดค่าเริ่มต้นจาก class Box

```
int Box::objectCount = 0;
```

เรียกใช้งาน static method

```
cout << "Initial Stage Count: " << Box::getCount() << endl;
```

```
1. #include <iostream>
2. using namespace std;
3. class Box {
4.     public:
5.         static int objectCount;
6.         // Constructor definition
7.         Box(float l = 2.0, float b = 2.0, float h = 2.0) {
8.             cout << "Constructor called." << endl;
9.             length = l;
10.            breadth = b;
11.            height = h;

12.            // Increase every time object is created
13.            objectCount++;
14.        }
15.        float Volume() {
16.            return length * breadth * height;
17.        }
18.        static int getCount() {
19.            return objectCount;
20.        }
```

```

21. private:
22.     float length;    // Length of a box
23.     float breadth;   // Breadth of a box
24.     float height;    // Height of a box
25. };

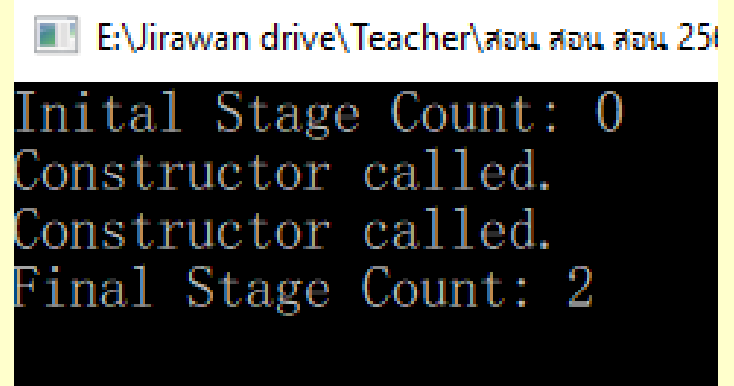
26. // Initialize static member of class Box
27. int Box::objectCount = 0;

28. main( ) {
29.     // Print total number of objects before creating object.
30.     cout << "Initial Stage Count: " << Box::getCount() << endl;

31.     Box Box1(3.3, 1.2, 1.5);    // Declare box1
32.     Box Box2(8.5, 6.0, 2.0);    // Declare box2

33.     // Print total number of objects after creating object.
34.     cout << "Final Stage Count: " << Box::getCount() << endl;
35.
36. }

```



```

E:\Jirawan drive\Teacher\สอน สอน สอน 2561
Initial Stage Count: 0
Constructor called.
Constructor called.
Final Stage Count: 2

```

Static เมธอดและแอททริบิวต์

- จะเห็นได้ว่าเมื่อประกาศให้ `objectCount` เป็น static int โปรแกรมจะทำให้ `objectCount` เป็นเสมือน Global variable เมื่อโปรแกรมเปลี่ยนค่าของ `objectCount` ทุกตัวจะเห็นค่าที่เปลี่ยน
- การประกาศแอททริบิวต์หรือเมธอดเป็น static ทำให้เราสามารถเรียกใช้ Method หรือ initial ค่าโดยไม่ต้องสร้าง Object เนื่องจากการประกาศ Member แบบ static เปรียบเสมือนการประกาศ Member ให้กับ Class ไม่ใช่ให้กับ Object

คำถาม



Quiz 12

1) โปรแกรมคิดเงินค่าหอพักนิสิต

- 1) มีการสร้าง Constructor
- 2) มีการใช้สร้าง Overloading constructor 2 รูปแบบ
- 3) มีการใช้คำสั่ง Default Copy Constructor
- 4) มีการใช้คำสั่ง Destructor

Quiz 12

1) โปรแกรมธนาคารมีการให้บริการฝาก-ถอน เงินให้ลูกค้า โดยมีข้อกำหนดให้ต่อไปนี้

- 1) มีการสร้าง Constructor
- 2) มีการใช้สร้าง Overloading constructor 2 รูปแบบ
- 3) มีการใช้คำสั่ง Default Copy Constructor
- 4) มีการใช้คำสั่ง Destructor