



# **การพ้องชื่อของเมทอด (Method Overloading)**

**Benjamas Panyangam  
Matinee Kiewkanya  
Computer Science, CMU**

## การพ้องชื่อของเมทอด (Method Overloading)

- ❁ เมทอดพ้องชื่อ (Overloaded Method) คือเมทอดที่มีชื่อเหมือนกัน แต่มี Method Signature ต่างกัน
- ❁ Method Signature หมายถึง รายการพารามิเตอร์ของเมทอด อันประกอบด้วยจำนวนของพารามิเตอร์ และชนิดของพารามิเตอร์
- ❁ Overloaded Method อาจเป็นเมทอดที่จำนวนของพารามิเตอร์ต่างกัน หรือชนิดของพารามิเตอร์ต่างกัน หรือทั้งสองอย่างก็ได้

# Overloaded Method

กรณีมี Method Signature ต่างกัน

❁ ตัวอย่างของ Overloaded Method เช่น

- `void calArea(int);`
- `void calArea(int, int);`
- `void calArea(String, int);`
- `void calArea(String, float);`

❁ เมื่อมีการเรียกเมทอด `calArea()` การที่คอมไพเลอร์จะพิจารณาว่าควรเรียกใช้เมทอดใด จะพิจารณาจากอาร์กิวเมนต์ที่ส่งมา ดังตัวอย่าง

- `calArea(5);`
- `calArea(5,10);`
- `calArea("circle",5);`
- `calArea("rectangle",10.5);`

## ตัวอย่างโปรแกรมที่ 1

โปรแกรมเพื่อสร้าง **Overloaded Method** สำหรับ  
การทำงาน 2 อย่าง คือ

- การหาผลบวกของเลขหนึ่งจำนวนโดยบวกเข้ากับค่าเดียวกัน
- การหาผลบวกของเลขสองจำนวน

<b>MyClass</b>
<b>- ans</b>
<b>+ addData(int)</b> <b>+ addData(int,int)</b>

```
class MyClass{  
    private int ans;  
    public void addData(int data){  
        ans = data + data;  
        System.out.println("ans = " + ans);  
    }  
    public void addData(int data1, int data2) {  
        ans = data1 + data2;  
        System.out.println("ans = " + ans);  
    }  
}
```

โปรแกรม

```
public class JavaApp1 {  
    public static void main(String[] args) {  
        MyClass myObj = new MyClass();  
        myObj.addData(5);  
        myObj.addData(4,2);  
    }  
}
```

ผลการทำงาน

```
ans = 10  
ans = 6
```

# Overloaded Method

กรณีเมทอดที่มี **Method Signature** เหมือนกัน แต่  
มี **Return Type** ต่างกัน

จะไม่สามารถสร้างเป็น **Overloaded Method** ได้



เช่น

- **int calArea(int);**
- **float calArea(int);**



เช่น

- **void calArea(int,int);**
- **int calArea(int,int);**

## Overloaded Method

กรณี **Method Signature** มีจำนวนพารามิเตอร์เท่ากันแต่มีชนิดของพารามิเตอร์กำกับหรือคล้ายคลึงกัน

❁ การเรียกใช้จะขึ้นอยู่กับชนิดข้อมูลที่ส่งไป เช่น มีเมทอด  
**void calArea(float);** และ  
**void calArea(double);**

❁ หากมีการเรียกใช้ด้วยคำสั่ง **calArea(5.0);**  
จะทำให้เมทอด **void calArea(double);** ทำงาน

❁ หากมีการเรียกใช้ด้วยคำสั่ง **calArea(5.0f);**  
จะทำให้เมทอด **void calArea(float);** ทำงาน

❁ มีเมทอด **void calArea(float);** และ  
**void calArea(double);**

❁ หากมีการเรียกใช้ด้วยคำสั่ง **calArea(5);**  
เมทอด **void calArea(float);** สามารถทำงานได้

เพราะคอมไพเลอร์จะสามารถแปลงข้อมูล คือเลข 5 (default คือชนิด int) ไปเป็นชนิดอื่นที่มีขนาดใหญ่กว่าให้โดยอัตโนมัติดังนี้

**byte → int → long → float → double**

นั่นหมายความว่า หากมีการนิยามเมทอด **void calArea(long);**  
ไว้ด้วย เมทอดนี้ก็จะถูกเรียกใช้แทน



❁ ดังนั้นหากมีความจำเป็นต้องสร้างเมทอดฟังก์ชั่ที่มีพารามิเตอร์กำกวมก็สมำารถทำไ้ เพียงแต่ในการเรียกใช้งานให้ระบุชนิดของอาร์กิวเมนต์ให้ชัดเจนตัวอย่างเช่น

- `calArea((byte)5);` จะเรียกใช้ `void calArea(byte);`
- `calArea((long)5);` จะเรียกใช้ `void calArea(long);`
- `calArea((float)5);` จะเรียกใช้ `void calArea(float);`
- `calArea((double)5);` จะเรียกใช้ `void calArea(double);`

## ตัวอย่างโปรแกรมที่ 2

โปรแกรมเพื่อสร้าง **Overloaded Method** ที่มี  
พารามิเตอร์กำกวม

<b>MyClass</b>
<b>- ans</b>
<b>+ overLoad(float,float)</b> <b>+ overLoad(double,double)</b>

```

class MyClass {
    private double ans;
    public void overLoad(float data1, float data2) {
        ans = data1 - data2;
        System.out.println("ans = " + ans);
    }

    public void overLoad(double data1, double data2){
        ans = data1 + data2;
        System.out.println("ans = " + ans);
    }
}

```

โปรแกรม

```

public class JavaApp2 {
    public static void main(String[] args) {
        MyClass myObj = new MyClass();

        myObj.overLoad((float)5, (float)5);
        myObj.overLoad((double)5, (double)5);
    }
}

```

ans = 0.0  
ans = 10.0

ผลการทำงาน




# **การพ้องชื่อของตัวดำเนินการ (Operator Overloading)**

**Matinee Kiewkanya  
Computer Science, CMU**

## การพ่วงชื่อของตัวดำเนินการ (Operator Overloading)

- ❁ ตัวดำเนินการ(Operator) โดยทั่วไปจะสามารถนำไปใช้ทำงานกับข้อมูลชนิดพื้นฐานเท่านั้น แต่ตัวดำเนินการส่วนใหญ่ที่ภาษาโปรแกรมได้จัดเตรียมไว้ ไม่สนับสนุนการทำงานกับชนิดข้อมูลที่ผู้เขียนโปรแกรมกำหนดขึ้นมา (User-defined Type) ได้
- ❁ ในกรณีที่ผู้เขียนโปรแกรมต้องการนำตัวดำเนินการพื้นฐานไปใช้งานกับชนิดข้อมูลใหม่ จะต้องสร้างเมทอดขึ้นมาใหม่ที่มีชื่อเหมือนกับตัวดำเนินการพื้นฐาน กล่าวคือให้สร้างตัวดำเนินการพ่วงชื่อนั่นเอง

- ❁ ตัวดำเนินการพ้องชื่อ(Overloaded Operator) คือ ตัวดำเนินการที่มีชื่อเหมือนกัน แต่มีการทำงานที่แตกต่างกัน โดยขึ้นอยู่กับชนิดข้อมูลที่นำมากระทำกับตัวดำเนินการ
- ❁ การทำงานของตัวดำเนินการพ้องชื่อนี้จะยังคงรักษาลำดับการมาก่อน (Precedence) ของตัวดำเนินการไว้ในลำดับเดิม และจะต้องมีตัวถูกดำเนินการ (Operand) ในจำนวนเท่าเดิม

-  ข้อดีของการสร้างตัวดำเนินการฟังก์ชันคือ
- สามารถนำตัวดำเนินการที่มีอยู่แล้ว มาใช้งานในหน้าที่ใหม่กับชนิดข้อมูลใหม่
  - ชื่อของตัวดำเนินการที่มีอยู่แล้วจะสื่อความหมายในการทำงาน ทำให้ลดปัญหาในการจดจำชื่อของตัวดำเนินการ

## ภาษา Java ไม่สนับสนุนการเขียนโปรแกรมเพื่อสร้าง Overloaded Operator

 ตัวดำเนินการของภาษา C++ ที่สามารถสร้างเป็นตัวดำเนินการพ้องชื่อได้

+	-	*	/	%	^	&
	~	!	=	<	>	+=
-=	*=	/=	%=	^=	&=	=
<<	>>	>>=	<<=	==	!=	<=
>=	&&		++	--	,	->*
->	()	[]	new	delete	new[]	delete[]



## รูปแบบการนิยามตัวดำเนินการพ้องชื่อในภาษา C++

```
return_type operator operator_name ([parameters]) {  
    [body]  
}
```

โดยที่

- ❁ **return\_type** คือ ชนิดข้อมูลที่เมทอดจะคืนค่ากลับ  
หากไม่คืนค่ากลับให้ระบุเป็น **void**
- ❁ **operator\_name** คือ ชื่อตัวดำเนินการที่จะสร้างเป็น  
ตัวดำเนินการพ้องชื่อ
- ❁ **parameters** คือ ตัวแปรที่กำหนดขึ้นเพื่อรับข้อมูลเข้า  
จากผู้เรียกใช้
- ❁ **body** คือ ส่วนกำหนดการทำงานของเมทอด

# ตัวอย่างภาษา C++ โปรแกรมที่ 1

โปรแกรมเพื่อสร้าง Overloaded Operator ++  
และ -- สำหรับคลาส OddNumber

OddNumber
- number
<ul style="list-style-type: none"><li>+ OddNumber()</li><li>+ OddNumber(int)</li><li>+ getNumber():int</li><li>+operator++()</li><li>+operator--()</li></ul>

- ❁ ความหมายตามปกติของเครื่องหมาย ++ และ -- คือการเพิ่มค่าขึ้นทีละ 1 และการลดค่าลงทีละ 1
- ❁ ++ และ -- จะใช้กับ type ดั้งเดิม (primitive type) ได้แก่ type ที่เป็นตัวเลข เช่น int, float, long, double รวมทั้ง char
- ❁ overload operator ++ และ -- เพื่อตัวอย่างนี้ จะให้สามารถใช้งานกับ type ที่สร้างขึ้นใหม่ คือ object ของ class OddNumber (คลาสเลขคี่)
- ❁ หน้าทีใหม่ของ operator ++ และ -- ที่เกิดขึ้น คือ การเพิ่มค่าและลดค่าทีละ 2

## โปรแกรม

```
#include <iostream>
using namespace std;
class OddNumber {
    private :
        int number;
    public :
        OddNumber() { number = 1;} //constructor
        OddNumber(int x) { //constructor
            if (x%2==1)
                number=x;
            else
                number=1;
        }
        int getNumber( ) {
            return number;
        }
        void operator++() { //overloaded operator
            number=number+2;
        }
        void OddNumber::operator--() { //overloaded operator
            number=number-2;
        }
};
```

```
int main(){

    OddNumber oddNum1;
    OddNumber oddNum2(11);
    cout << " oddNumber1 = " << oddNum1.getNumber();

    ++oddNum1;
    cout << "\n ++oddNumber1 = " << oddNum1.getNumber();

    ++oddNum1;
    cout << "\n ++oddNumber1 = " << oddNum1.getNumber();

    --oddNum1;
    cout << "\n --oddNumber1 = " << oddNum1.getNumber();
    cout << "\n\n oddNumber2 = " << oddNum2.getNumber();

    --oddNum2;
    cout << "\n --oddNumber2 = " << oddNum2.getNumber();

    ++oddNum2;
    cout << "\n ++oddNumber2 = " << oddNum2.getNumber();
}
```

## ผลการทำงาน

```
oddNumber1 = 1  
++oddNumber1 = 3  
++oddNumber1 = 5  
--oddNumber1 = 3  
  
oddNumber2 = 11  
--oddNumber2 = 9  
++oddNumber2 = 11
```

## ตัวอย่างภาษา C++ โปรแกรมที่ 2

โปรแกรมเพื่อสร้าง Overloaded Operator ==  
สำหรับคลาส Salary

Salary
<ul style="list-style-type: none"><li>- base</li><li>- bonus</li></ul>
<ul style="list-style-type: none"><li>+ setSalary (int)</li><li>+ operator==(Salary):int</li></ul>

```
#include <iostream>
```

```
using namespace std;
```

โปรแกรม

```
class Salary{ // define class Salary
```

```
private:
```

```
    float base;
```

```
    float bonus;
```

```
public:
```

```
void setSalary(int i) {
```

```
    cout<<"Enter base and bonus of salary #"<<i<<": ";
```

```
    cin>>base>>bonus;
```

```
}
```

```
int operator== (Salary sal2) { // overloaded operator ==
```

```
    float sum1,sum2;
```

```
    sum1 = base + bonus;
```

```
    sum2 = sal2.base + sal2.bonus;
```

```
    if (sum1==sum2)
```

```
        return 1;
```

```
    else
```

```
        return 0;
```

```
}
```

```
};
```



```
int main()
{
    Salary emp1,emp2;
    emp1.setSalary(1);
    emp2.setSalary(2);
    if ( emp1==emp2 )
        cout<<"Both salaries are equal";
    else
        cout<<"Both salaries are not equal";
}
```

### ตัวอย่างผลการทำงาน

**Enter base and bonus of salary #1: 10000 5000**  
**Enter base and bonus of salary #2: 12000 3000**  
**Both salaries are equal**

**Enter base and bonus of salary #1: 10000 5000**  
**Enter base and bonus of salary #2: 15000 3000**  
**Both salaries are not equal**



**เมท็อดภายนอกคลาส**

**Benjamas Panyangam  
Matinee Kiewkanya  
Computer Science, CMU**

## การออกแบบเมทอด

- ❁ หลักการของการออกแบบ method ที่เป็น member method ของคลาส คือ method นั้นมักจะใช้ข้อมูลภายในคลาส
- ❁ ในกรณีที่การทำงานต้องใช้ข้อมูลจากคลาสหลายคลาส ต้องแยกเขียนให้เป็น method ภายนอกต่างหาก ไม่ใช่ method ของคลาสใดคลาสหนึ่ง

## การพิจารณาเมทอดภายนอกคลาส

การจะพิจารณาว่าเมทอดใดควรเป็นเมทอดภายในคลาส หรือเมทอดภายนอกคลาส ให้พิจารณาการกระทำกับข้อมูล ซึ่งโดยปกติแล้วเมทอดภายในคลาสหรือเมทอดสมาชิกจะมีการเข้าถึงข้อมูลของวัตถุเพียงวัตถุเดียว (ยกเว้นตัวดำเนินการพ้องชื่อ) ตัวอย่างเช่น

- ต้องการหาพื้นที่ของวงกลม 1 วง จะต้องสร้างเป็น **member method** ของ **class** วงกลม
- ต้องการหาพื้นที่ของวงกลม N วง แต่จะหาทีละวง กรณีนี้จะต้องสร้างเป็น **member method** ของ **class** วงกลม
- ต้องการหาค่าเฉลี่ยของพื้นที่ของวงกลมทั้ง N วง กรณีนี้การหาค่าเฉลี่ยเป็นการกระทำกับวงกลมหลายวง เพราะต้องหาผลรวมของพื้นที่ทุกวงกลมก่อน จึงจะหาค่าเฉลี่ยได้ ดังนั้นจะต้องสร้างเป็น **method** ภายนอก

## ทดลองวิเคราะห์โจทย์

เพื่อแก้ปัญหาดังต่อไปนี้ ควรออกแบบให้มีคลาสอะไรบ้าง และ  
เมทอดสำหรับทำงานเหล่านี้ ควรเป็นเมทอดของคลาสหรือ  
เมทอดภายนอก ?

- 1) ต้องการหาความแตกต่างระหว่างเงินเดือนของ  
นายแพทย์มานะและอาจารย์วิชุดา
- 2) หากแพทย์ต้องเสียภาษีเป็นจำนวน 5% ของ  
เงินเดือน ต้องการหาคำนวณหาภาษีที่แพทย์แต่ละ  
คนจะต้องจ่าย
- 3) ต้องการหานักศึกษาที่ได้คะแนนสูงสุดและ  
ต่ำสุดคือนักศึกษาคนใด

## ตัวอย่างโปรแกรมที่ 1

โปรแกรมเพื่อหาผลรวมของพื้นที่วงกลม 1 ชั้น กับ  
พื้นที่สี่เหลี่ยม 1 ชั้น

Circle
- radius
+ setRadius() + getRadius()

Rectangle
- width - height
+ setWidthHeight() + getWidth() + getHeight()

## วิเคราะห์โจทย์

- ❁ หากต้องการหาพื้นที่วงกลม จะต้องกำหนดให้เมทรีอดสำหรับการหาพื้นที่เป็นเมทรีอดสมาชิกของคลาสวงกลม เพราะการทำงานของเมทรีอดจะใช้ข้อมูลภายในคลาส คือ รัศมี เท่านั้น
- ❁ หากต้องการหาพื้นที่สี่เหลี่ยม จะต้องกำหนดให้เมทรีอดสำหรับการหาพื้นที่เป็นเมทรีอดสมาชิกของคลาสสี่เหลี่ยม เพราะการทำงานของเมทรีอดจะใช้ข้อมูลภายในคลาส คือ ความกว้างและความยาว เท่านั้น
- ❁ แต่หากต้องการคำนวณหาผลรวมของพื้นที่วงกลมกับพื้นที่สี่เหลี่ยม จะต้องใช้ข้อมูลจากทั้งสองคลาส ดังนั้นเมทรีอดสำหรับการทำางานนี้ จะต้องไม่ใช่เมทรีอดภายในคลาสใดคลาสหนึ่ง กล่าวคือจะต้องสร้างเป็น เมทรีอดภายนอก

```
import java.util.Scanner;
class Circle {
    private float radius;

    public void setRadius() {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter radius of circle : ");
        radius = input.nextFloat();
    }

    public float getRadius(){
        return(radius);
    }
} //end class Circle
```



```
class Rectangle {  
    private float width, height;  
  
    public void setWidthHeight() {  
        Scanner input = new Scanner(System.in);  
        System.out.print("Enter width and height : ");  
        width = input.nextFloat();  
        height = input.nextFloat();  
    }  
  
    public float getWidth(){  
        return(width);  
    }  
  
    public float getHeight(){  
        return(height);  
    }  
} // end class Rectangle
```

```
public class JavaApp1{
```

```
// Method Name: totArea()
```

```
// Description: Calculates the sum of areas of a circle
```

```
// object and a rectangle object.
```

```
static float totArea (Circle cirObj, Rectangle recObj) {
```

```
float cirArea;
```

```
float recArea;
```

```
// Calculate the circle area and the rectangle area.
```

```
cirArea = 3.14f *cirObj.getRadius()*cirObj.getRadius();
```

```
recArea = recObj.getWidth() * recObj.getHeight();
```

```
// Return the sum of the circle and rectangle areas.
```

```
return cirArea + recArea;
```

```
}
```

```
// Method Name: main()
public static void main(String[] args) {
    Circle myCircle = new Circle();
    myCircle.setRadius();

    Rectangle myRectangle = new Rectangle();
    myRectangle.setWidthHeight();

    System.out.print("Total Area = ");
    System.out.println(totArea(myCircle, myRectangle));

} //end main
} //end class JavaApp1
```

**ตัวอย่างผลการทำงาน**

```
Enter radius of circle : 1
Enter width and height of rectangle : 1  1
Total area = 4.1400003
```

## ตัวอย่างโปรแกรมที่ 2 : AF

❁ จงเขียนโปรแกรมเพื่อรับชื่อและคะแนนในการร้องเพลงของ  
นักร้อง AF2 จำนวน 5 คน แล้วทำการตัดเกรดแต่ละคนดังนี้  
80 ขึ้นไป ได้ A

50-79 ได้ B

ต่ำกว่า 50 ได้ C

พร้อมทั้งนับจำนวนคนที่ได้ในแต่ละเกรด  
เพื่อเปรียบเทียบกับเกรดของนักร้อง AF1 จำนวน 5 คน  
ซึ่งทราบจำนวนคนที่ได้ในแต่ละเกรดอยู่แล้ว

ให้หาว่านักร้องกลุ่ม AF1 และ AF2 ใครจะมีคุณภาพกว่ากัน  
โดยในการเปรียบเทียบเกรดให้มีค่าน้ำหนักดังนี้  
A 3 แต้ม B 2 แต้ม และ C 1 แต้ม

## class **Singer**

### data

#### **private :**

**String name**  
**int score**  
**char grade**

### method

#### **public :**

**void setData(int i)**  
**void setGrade(char g)**  
**int getScore()**  
**void printData()**

## class **Grade**

### data

#### **private :**

**int a\_no**  
**int b\_no**  
**int c\_no**

### method

#### **public :**

**Grade()**  
**Grade(int a,int b,int c)**  
**void setData()**  
**void printData()**  
**void increaseA()**  
**void increaseB()**  
**void increaseC()**  
**int getA()**  
**int getB()**  
**int getC()**

```
import java.util.Scanner;
```

```
class Singer{
```

```
    private String name;
```

```
    private int score;
```

```
    private char grade;
```

```
    public void setData(int i) {
```

```
        Scanner input = new Scanner(System.in);
```

```
        System.out.println("Data of singer #" + (i+1));
```

```
        System.out.print("Enter name and score : ");
```

```
        name = input.next();      score = input.nextInt();
```

```
    }
```

```
    public void setGrade(char g){
```

```
        grade = g;
```

```
    }
```

```
public int getScore(){  
    return(score);  
}
```

```
public void printData(){  
    System.out.print("Name : " + name + " Score: " +score);  
    System.out.println(" Grade :" + grade);  
}
```

```
} //end class Singer
```

```
class Grade {  
    private int a_no, b_no, c_no;  
    Grade() {} //constructor with 0 parameter  
    Grade(int a,int b,int c) { //constructor with 3 parameters  
        a_no=a; b_no=b; c_no=c;  
    }  
  
    public void setData() {  
        Scanner input = new Scanner(System.in);  
        System.out.print("Enter number of singers who got A : ");  
        a_no = input.nextInt();  
        System.out.print("Enter number of singers who got B : ");  
        b_no = input.nextInt();  
        System.out.print("Enter number of singers who got C : ");  
        c_no = input.nextInt();  
    }  
  
    public void printData() {  
        System.out.println("A = " + a_no + " B= " + b_no + " C= " + c_no);  
    }  
}
```



```
public void increaseA(){  
    a_no++;  
}  
public void increaseB(){  
    b_no++;  
}  
public void increaseC(){  
    c_no++;  
}  
public int getA(){  
    return(a_no);  
}  
public int getB(){  
    return(b_no);  
}  
public int getC(){  
    return(c_no);  
}
```

```
} // end class Grade
```

```

public class JavaApp2{
// Method Name: setAndCountGrade()
static Grade setAndCountGrade(Singer [] s){
    int i, score;
    Grade temp=new Grade(0,0,0);//call constructor แบบ 3 parameters
    for(i=0;i<5;i++){
        score = s[i].getScore();
        if (score >= 80){
            s[i].setGrade('A');
            temp.increaseA();
        }
        else if (score >= 50){
            s[i].setGrade('B');
            temp.increaseB();
        }
        else {
            s[i].setGrade('C');
            temp.increaseC();
        }
    } //end for
    return temp;
}

```

**// Method Name: compare()**

**static void compare(Grade g1,Grade g2){**

**int tot1,tot2;**

**tot1=g1.getA()\*3 + g1.getB()\*2 + g1.getC();**

**tot2=g2.getA()\*3 + g2.getB()\*2 + g2.getC();**

**if (tot1==tot2)**

**System.out.println("\nBoth AFs are equal");**

**else if (tot1>tot2)**

**System.out.println("\nAF1 win");**

**else**

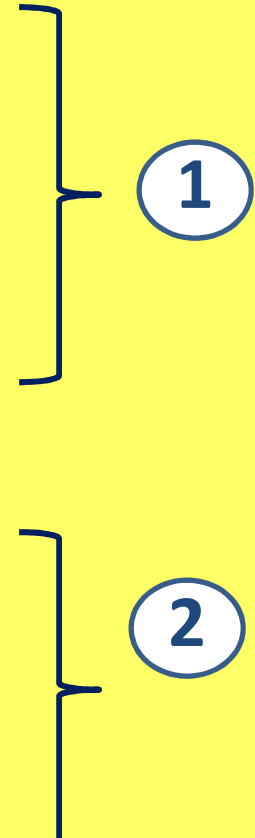
**System.out.println("\nAF2 win");**

**System.out.println("\nAF1 gets "+tot1+" points");**

**System.out.println("\nAF2 gets "+tot2+" points");**

**}**

```
public static void main(String[] args) {  
    int i;  
    Grade gAF1, gAF2; //call constructor with 0 parameter  
    Singer [] sAF2 =new Singer[5];  
  
    System.out.println("\nData of AF2 ");  
    System.out.println(".....");  
    for(i=0;i<5;i++){  
        sAF2[i] = new Singer();  
        sAF2[i].setData(i);  
    }  
  
    System.out.println("\nData of AF1");  
    System.out.println(".....");  
    gAF1=new Grade();  
    gAF1.setData();  
}
```



The diagram shows two groups of code lines. The first group, labeled with a circled '1', includes the loop for initializing the sAF2 array. The second group, labeled with a circled '2', includes the code for initializing the gAF1 object.

```
gAF2=setAndCountGrade(sAF2);  
compare(gAF1,gAF2);
```

} 3

```
System.out.println("\nGrade of AF1");  
gAF1.printData();
```

} 4

```
System.out.println("\nGrade of AF2");  
gAF2.printData();
```

} 5

```
for(i=0;i<5;i++)  
    sAF2[i].printData();
```

} 6

```
} // end main
```

```
} //end class JavaApp2
```

## ผลลัพธ์

### Data of AF2

.....

Data of singer # 1

Enter name and score : Tui 40

Data of singer # 2

Enter name and score : Nik 50

Data of singer # 3

Enter name and score : Nok 85

Data of singer # 4

Enter name and score : Dee 70

Data of singer # 5

Enter name and score : Pat 35

1

### Data of AF1

.....

Enter number of singers who got A: 4

Enter number of singers who got B: 1

Enter number of singers who got C: 0

2

**AF1 win**  
**AF1 gets 14 points**  
**AF2 gets 9 points**

3

**Grade of AF1**  
**A = 4 B = 1 C = 0**

4

**Grade of AF2**  
**A = 1 B = 2 C = 2**

5

**Name : Tui Score : 40 Grade : C**  
**Name : Nik Score : 50 Grade : B**  
**Name : Nok Score : 85 Grade : A**  
**Name : Dee Score : 70 Grade : B**  
**Name : Pat Score : 35 Grade : C**

6