5.1 OOP Wrapping Up

3 Sep 2015

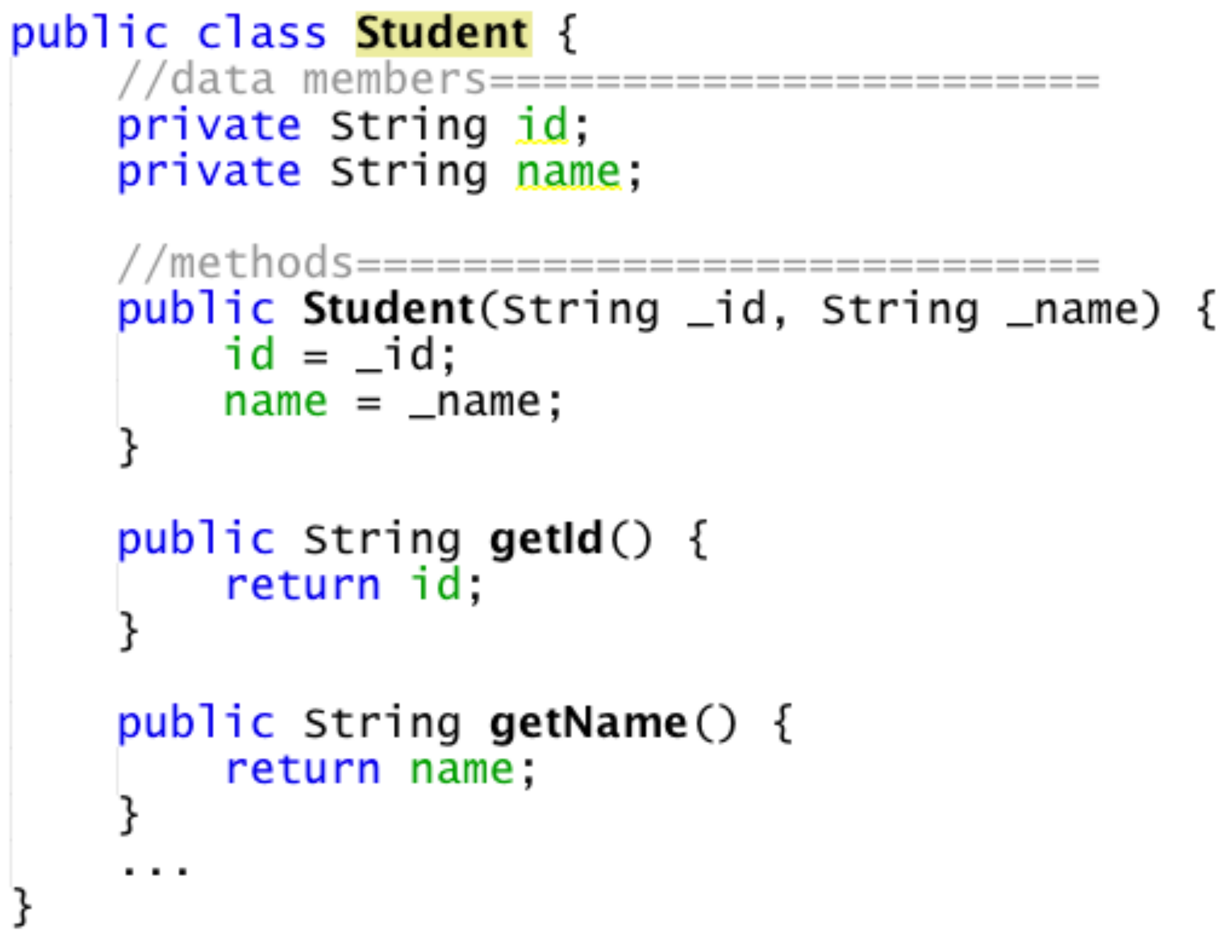
Objectives

- Define a class with data members and methods.
- Using ‘private’ and ‘public’.
- Using ‘static’ and non-static.

Class Creation

- Class contains:
  1. Data (called attribute, property or data member)
  2. Code (called method or behavior)
     - Constructor
     - Regular Method

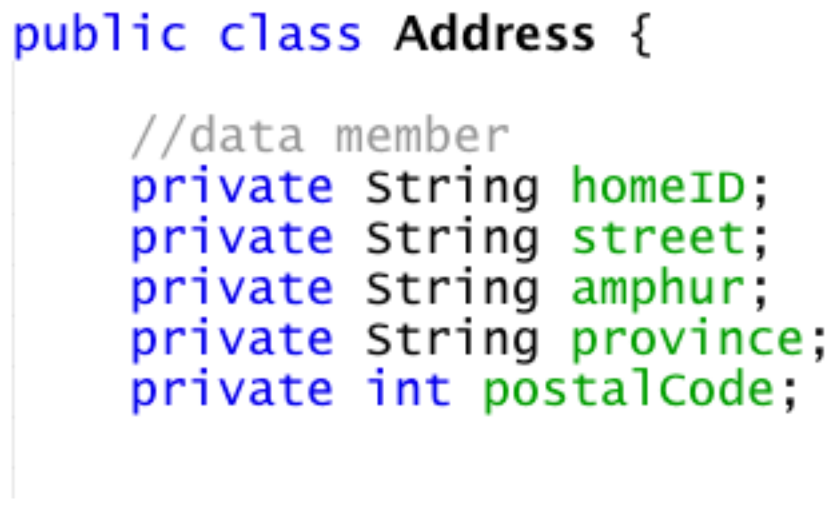
Class Creation



```java
public class Student {
    //data members========================
    private String id;
    private String name;

    //methods=============================
    public Student(String _id, String _name) {
        id = _id;
        name = _name;
    }

    public String getId() {
        return id;
    }

    public String getName() {
        return name;
    }
    ...
}
```

Class Creation Step-by-Step

1. List all data members (using ‘private’)



```
public class Address {

    //data member
    private String homeID;
    private String street;
    private String amphur;
    private String province;
    private int postalCode;
```

Class Creation Step-by-Step

2. Create all methods
   – Create constructor (using ‘public’)

```java
//constructor
public Address(String _homeID, String _street,
        String _amphur, String _province,
        int _postalCode) {
    homeID = _homeID;
    street = _street;
    amphur = _amphur;
    province = _province;
    postalCode = _postalCode;
}
```

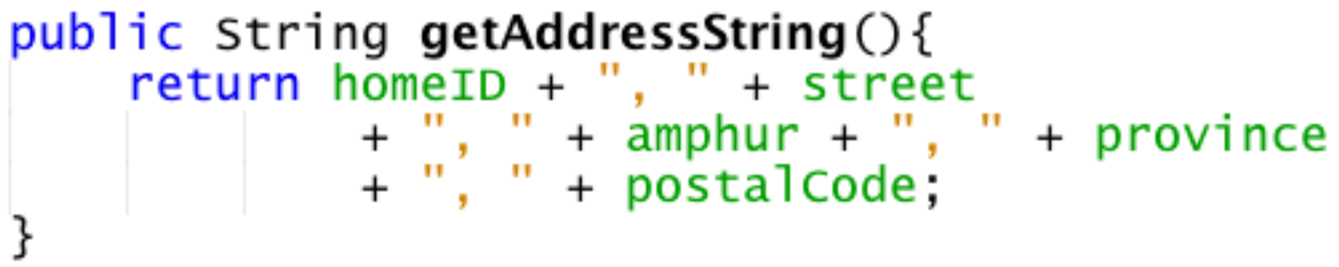
Class Creation Step-by-Step

– Create getter-setter (if need) (using ‘public’)

```
//getter-setter
public String getHomeID() {
     return homeID;
}

public void setHomeID(String _homeID) {
     homeID = _homeID;
}
...
```

Class Creation Step-by-Step

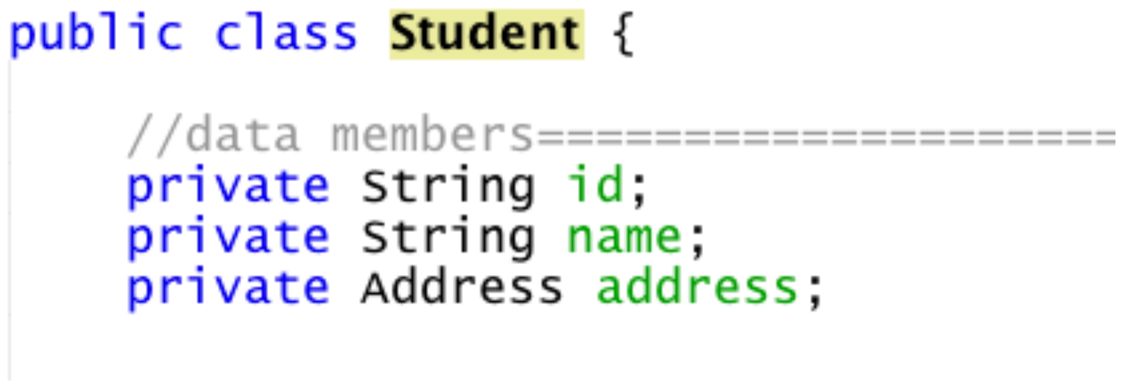
– Create other method (if need) (using ‘public’ for public method, using ‘private’ for internal use method)



```
public String getAddressString(){
    return homeID + ", " + street
            + ", " + amphur + ", " + province
            + ", " + postalCode;
}
```

Class is Complex Data Type

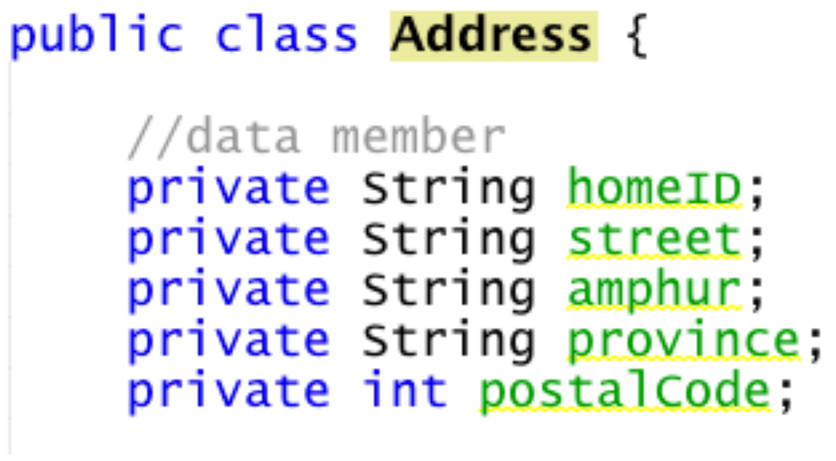
- Class is complex data type, data members of class can be:
  1. Simple Data Type eg. int, double, String, …
  2. Other Class eg. Address Class



```
public class Student {

    //data members====================
    private String id;
    private String name;
    private Address address;
```

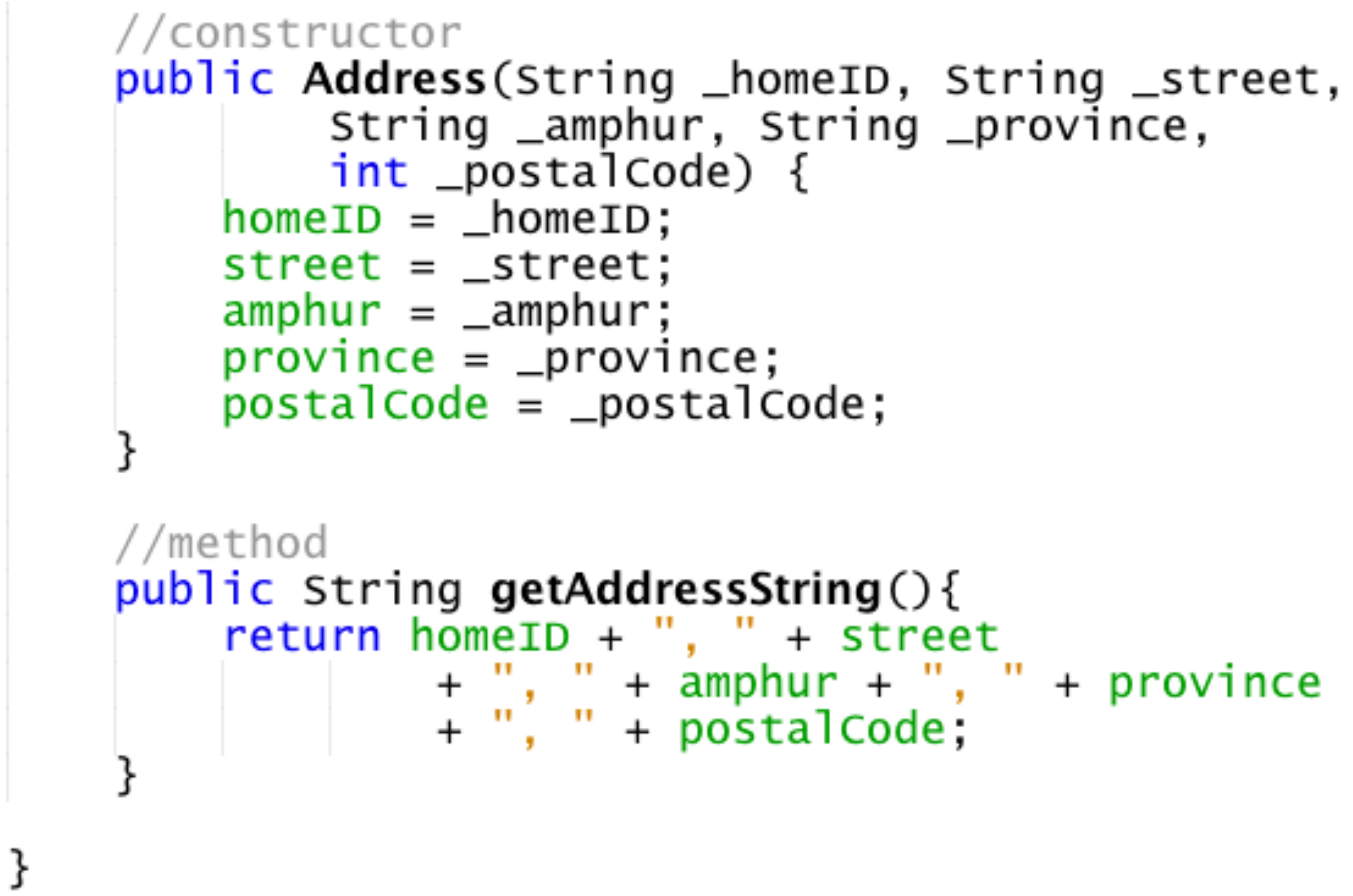
Class is Complex Data Type

- Ex. Using Address and Student Class
  - Address Class



```
public class Address {

    //data member
    private String homeID;
    private String street;
    private String amphur;
    private String province;
    private int postalCode;
```

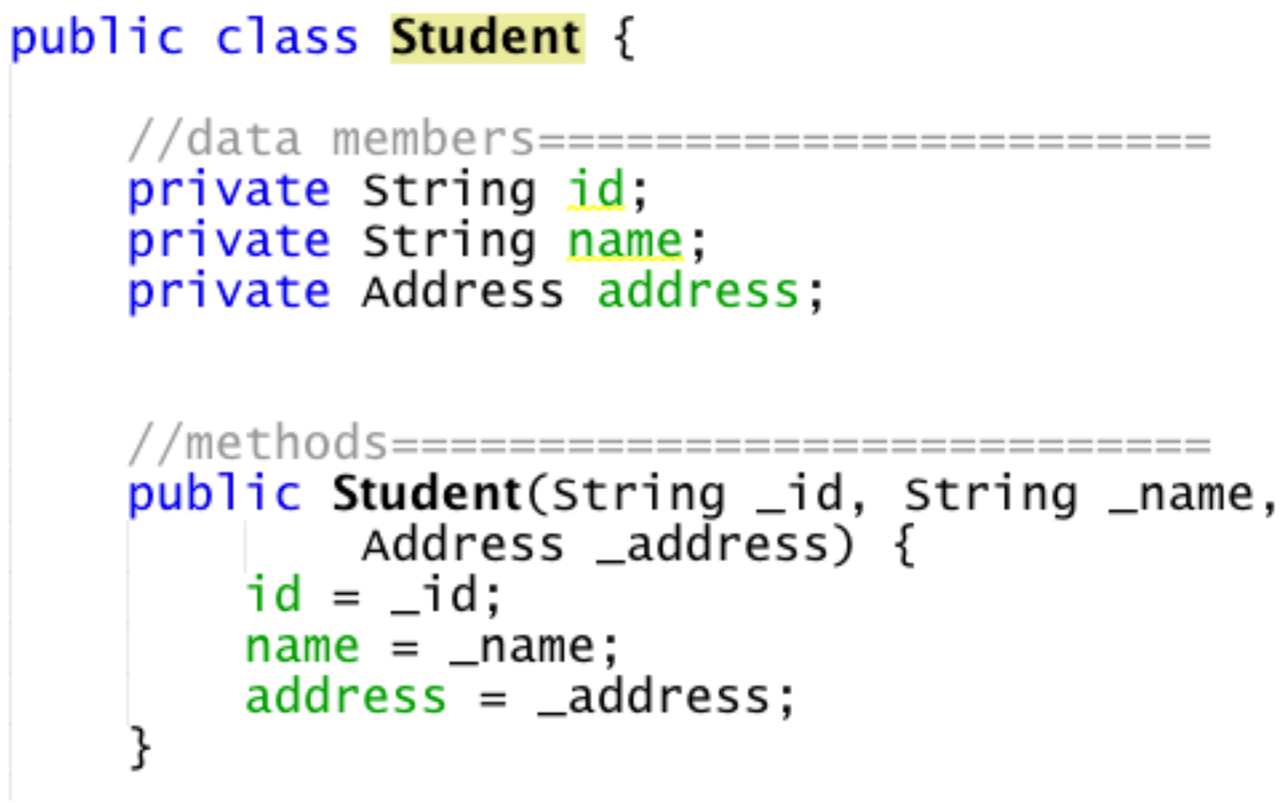
Class is Complex Data Type



```java
    //constructor
    public Address(String _homeID, String _street,
            String _amphur, String _province,
            int _postalCode) {
        homeID = _homeID;
        street = _street;
        amphur = _amphur;
        province = _province;
        postalCode = _postalCode;
    }

    //method
    public String getAddressString(){
        return homeID + ", " + street
                + ", " + amphur + ", " + province
                + ", " + postalCode;
    }

}
```

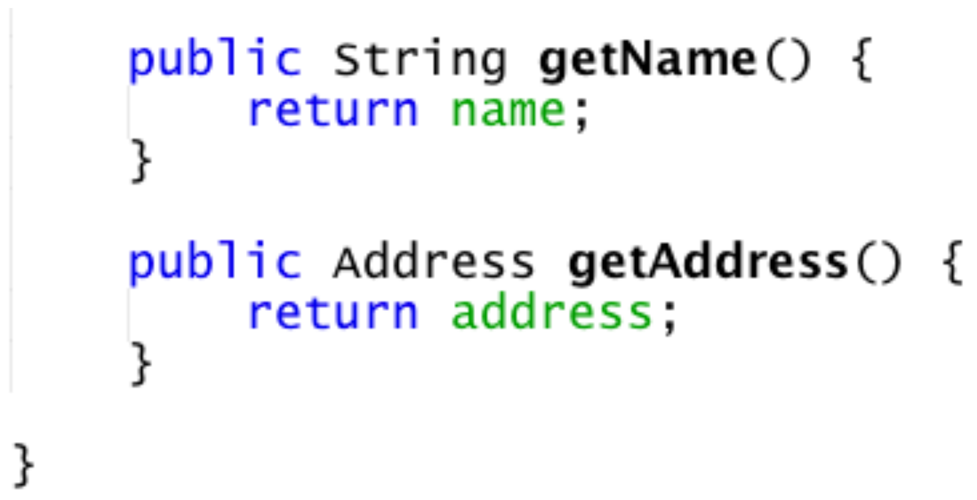
Class is Complex Data Type

– Student Class



```
public class Student {

    //data members=======================
    private String id;
    private String name;
    private Address address;


    //methods============================
    public Student(String _id, String _name,
            Address _address) {
        id = _id;
        name = _name;
        address = _address;
    }
```

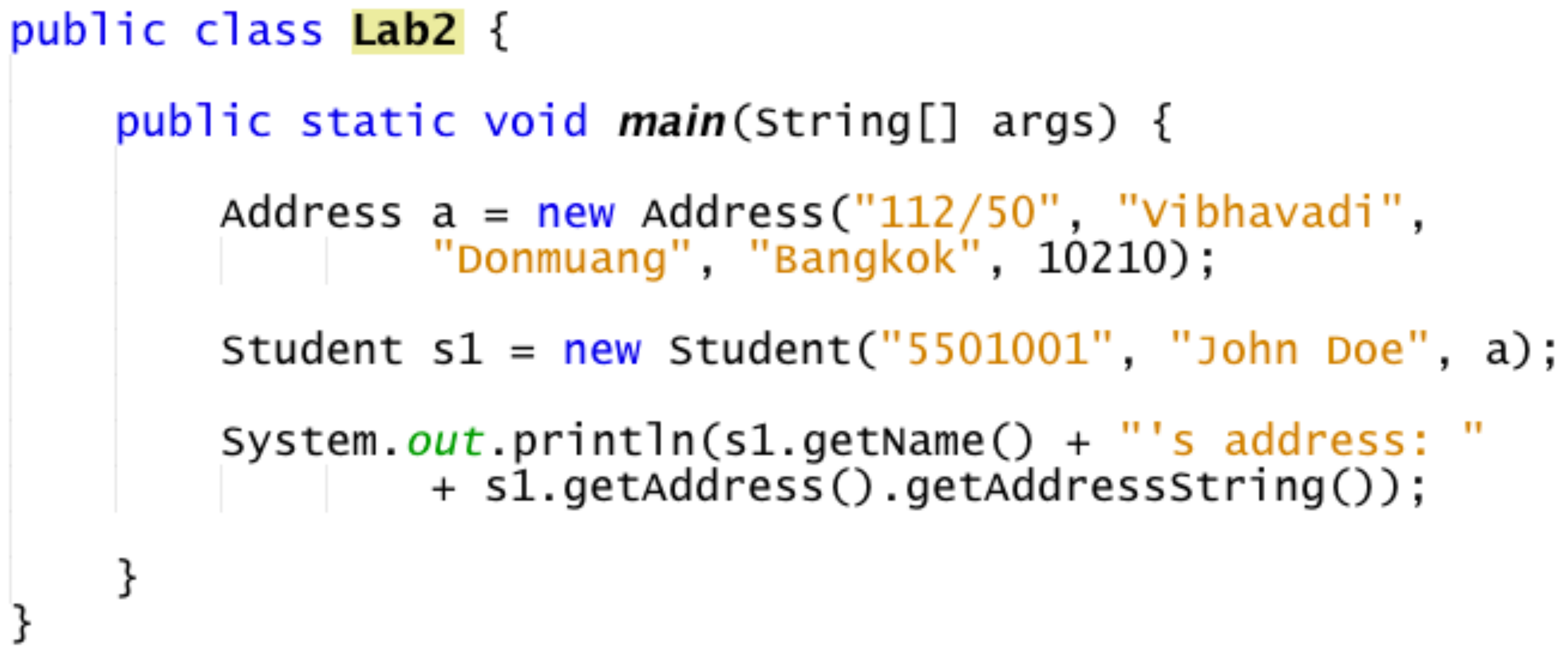
Class is Complex Data Type



```
    public String getName() {
        return name;
    }

    public Address getAddress() {
        return address;
    }

}
```

Class is Complex Data Type

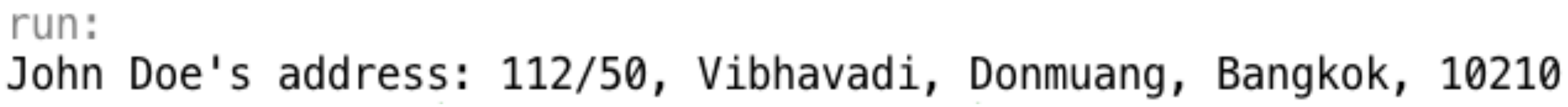
– Main Class



```
public class Lab2 {

    public static void main(String[] args) {

        Address a = new Address("112/50", "Vibhavadi",
                "Donmuang", "Bangkok", 10210);

        Student s1 = new Student("5501001", "John Doe", a);

        System.out.println(s1.getName() + "'s address: "
                + s1.getAddress().getAddressString());

    }
}
```

Class is Complex Data Type

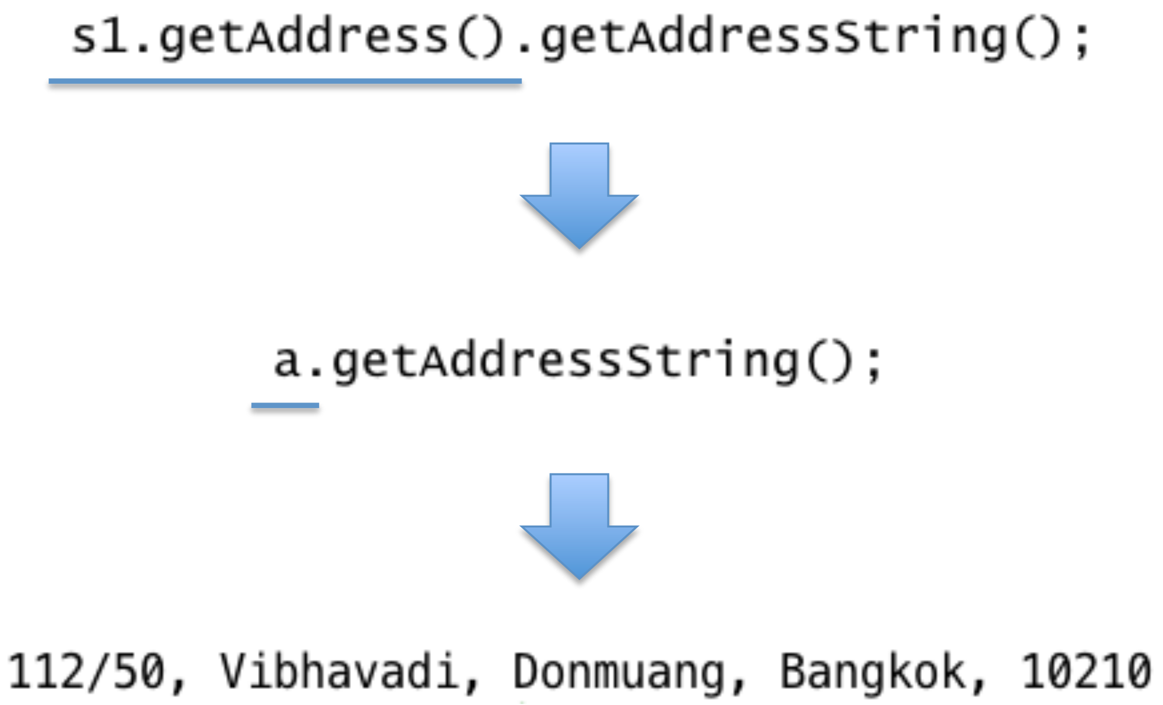
– Output:



```
run:
John Doe's address: 112/50, Vibhavadi, Donmuang, Bangkok, 10210
```

Class is Complex Data Type

– Expression: s1.getAddress().getAddressString() is called ‘Method Chaining’



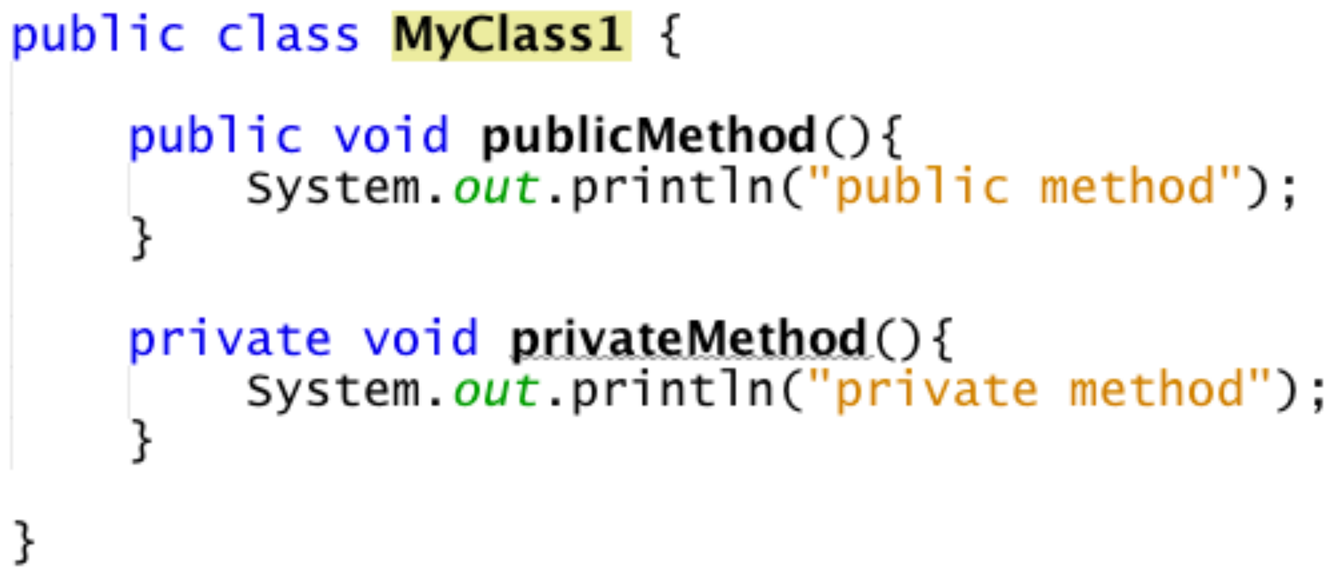
```
s1.getAddress().getAddressString();

a.getAddressString();

112/50, Vibhavadi, Donmuang, Bangkok, 10210
```

‘public’ vs ‘private’ method

- Public method is visible to all other classes.
- Private method is visible only in its class.

‘public’ vs ‘private’ method

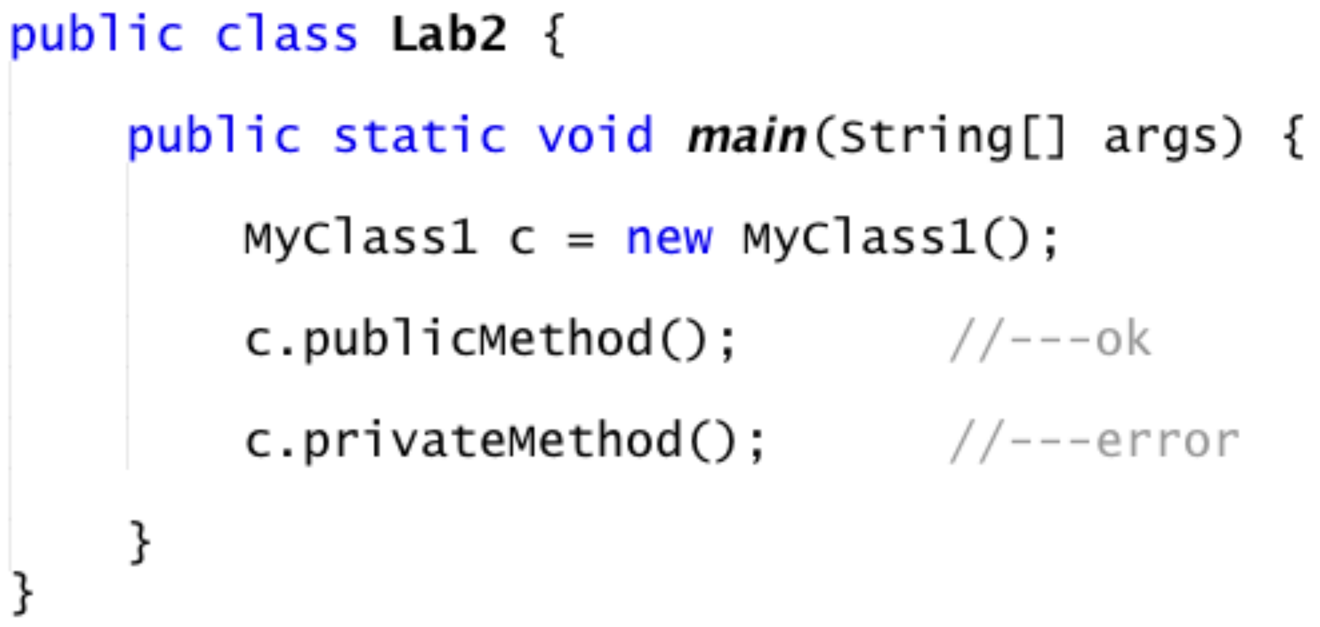
- Ex. Using public and private method
  - MyClass1 Class



```java
public class MyClass1 {

    public void publicMethod(){
        System.out.println("public method");
    }

    private void privateMethod(){
        System.out.println("private method");
    }

}
```

‘public’ vs ‘private’ method

– Main Class



```
public class Lab2 {
    public static void main(String[] args) {
        MyClass1 c = new MyClass1();
        c.publicMethod();       //---ok
        c.privateMethod();      //---error
    }
}
```

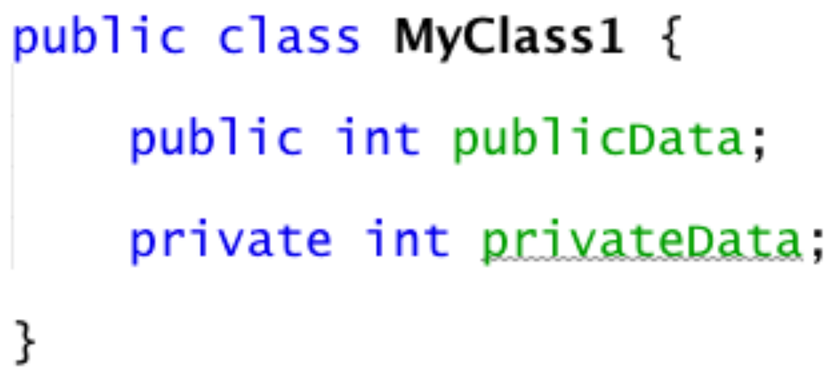
‘public’ vs ‘private’ data member

- Public data member is visible to all other classes.

- Private data member is visible only in its class.

‘public’ vs ‘private’ data member

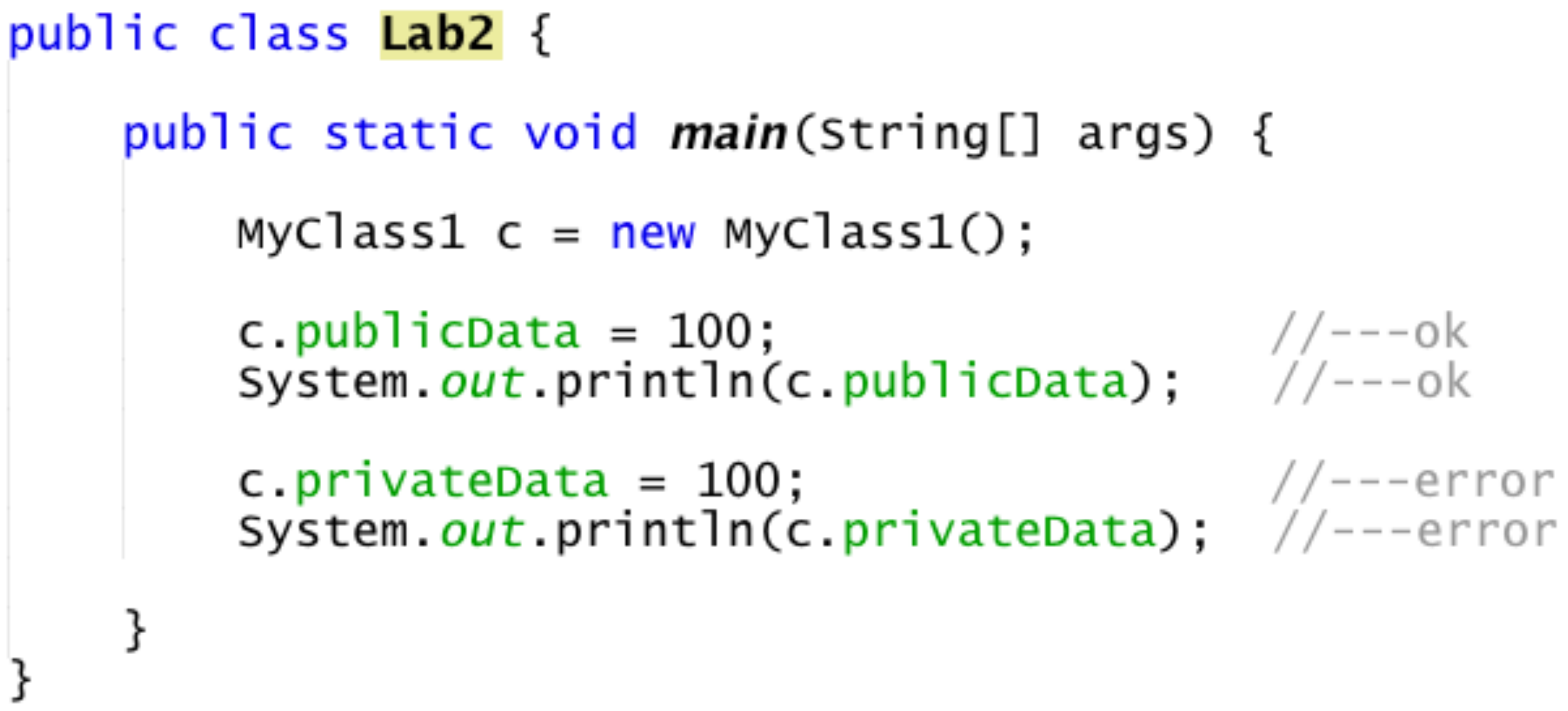
- Ex. Using public and private data member
  - MyClass1 Class



```
public class MyClass1 {
    public int publicData;
    private int privateData;
}
```

‘public’ vs ‘private’ data member

– Main Class



```
public class Lab2 {

    public static void main(String[] args) {

        MyClass1 c = new MyClass1();

        c.publicData = 100;                      //---ok
        System.out.println(c.publicData);        //---ok

        c.privateData = 100;                     //---error
        System.out.println(c.privateData);       //---error

    }
}
```

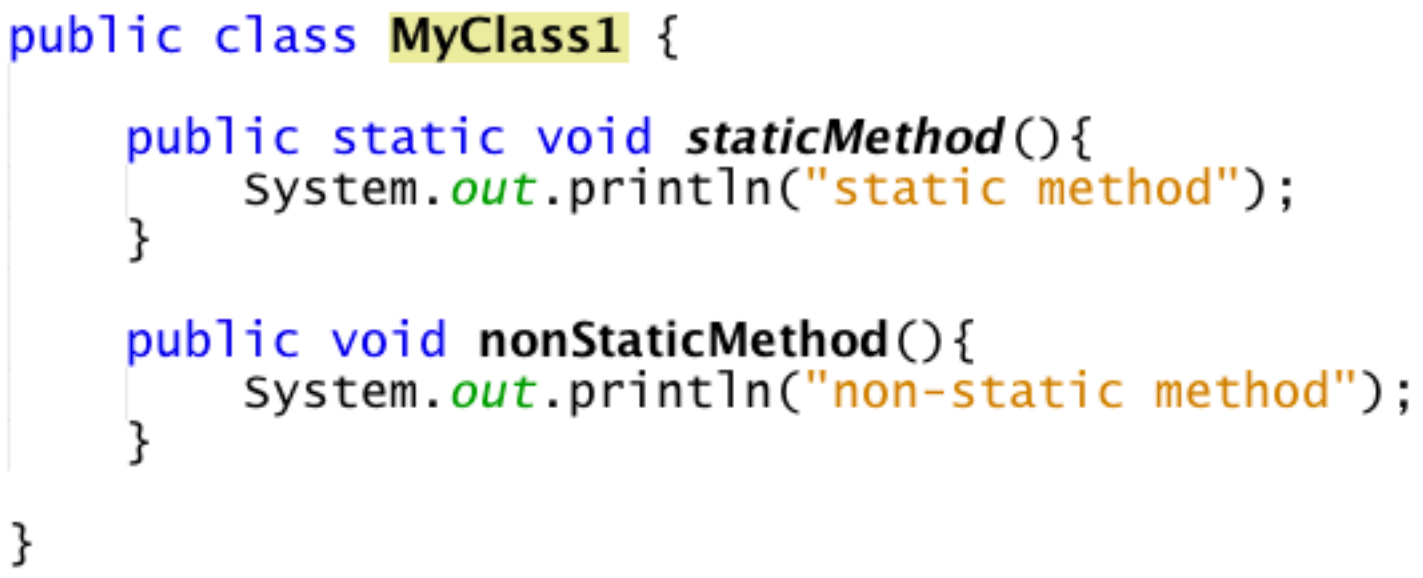
‘static’ vs ‘non-static’ method

- Static method is class method. To call static method using this syntax:
  - ClassName.staticMethod(<parameter>)

- Non-static method is instance method. To call instance method we must create new object then using this syntax:
  - objectName.nonStaticMethod(<parameter>)

‘static’ vs ‘non-static’ method

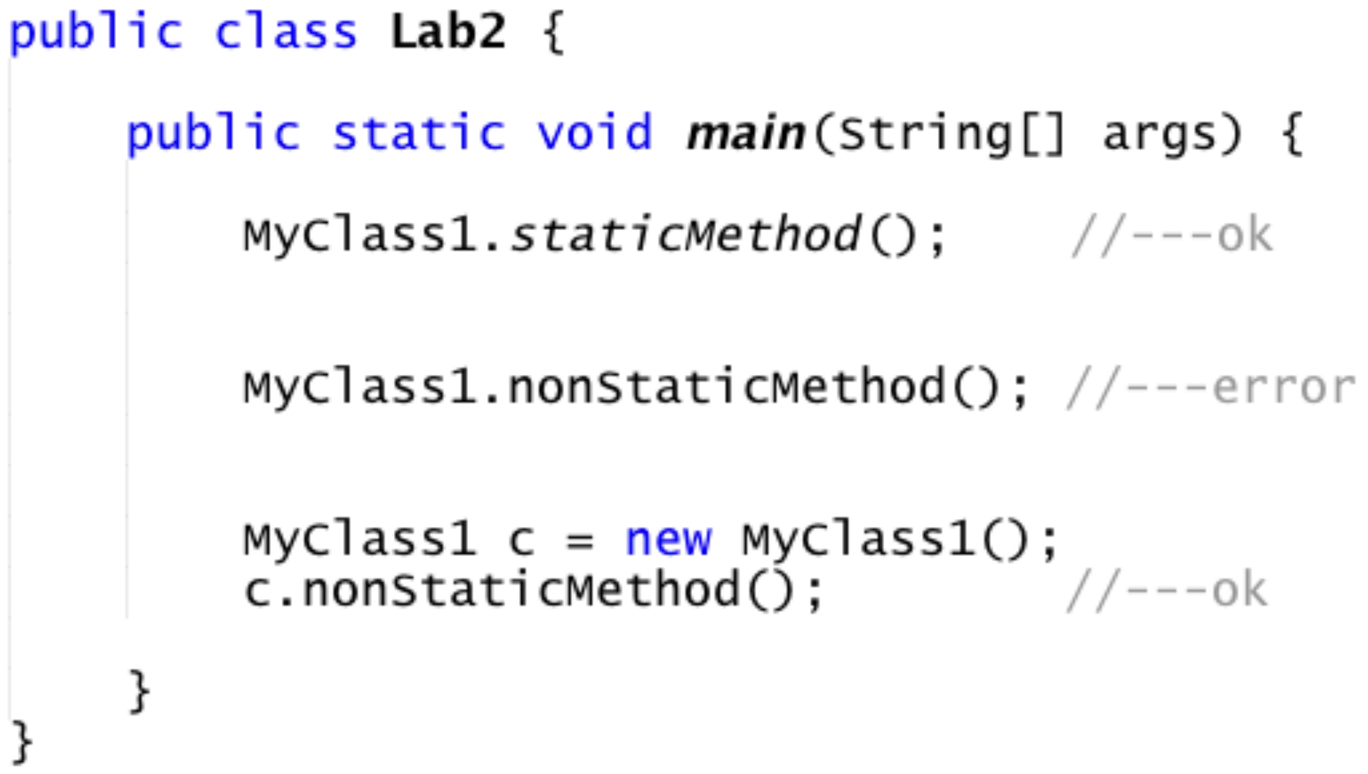
- Ex. Using static and non-static method
  - MyClass1 Class



```java
public class MyClass1 {

    public static void staticMethod(){
        System.out.println("static method");
    }

    public void nonStaticMethod(){
        System.out.println("non-static method");
    }

}
```

‘static’ vs ‘non-static’ method

– Main Class



```
public class Lab2 {

    public static void main(String[] args) {

        MyClass1.staticMethod();    //---ok

        MyClass1.nonStaticMethod(); //---error

        MyClass1 c = new MyClass1();
        c.nonStaticMethod();        //---ok

    }
}
```

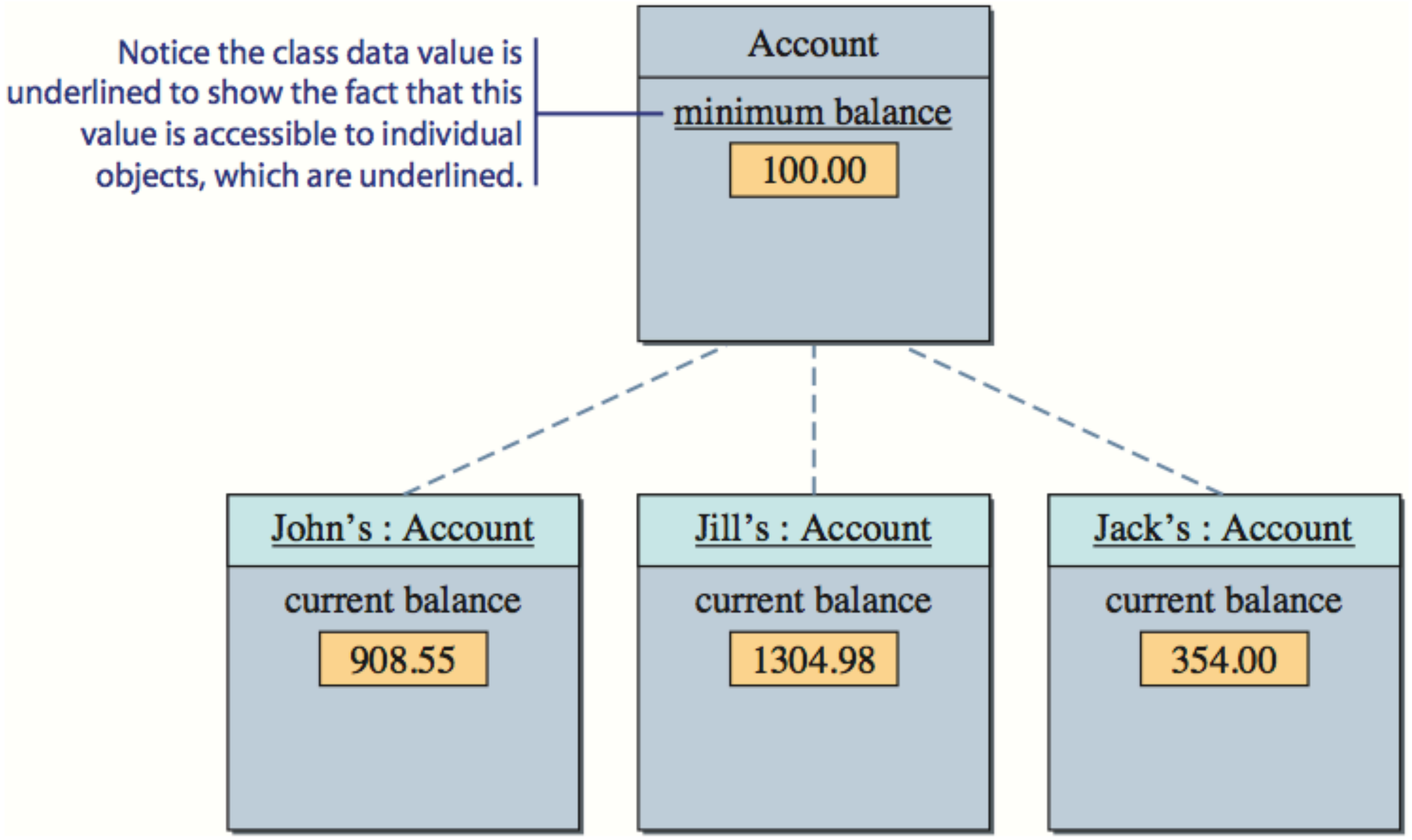
‘static’ vs ‘non-static’ data member

- Static data member is class data member. All objects of this class share the same copy of static data member.

- Non-static data member is instance data member. Each objects of this class contains its own copy of all instance variables.

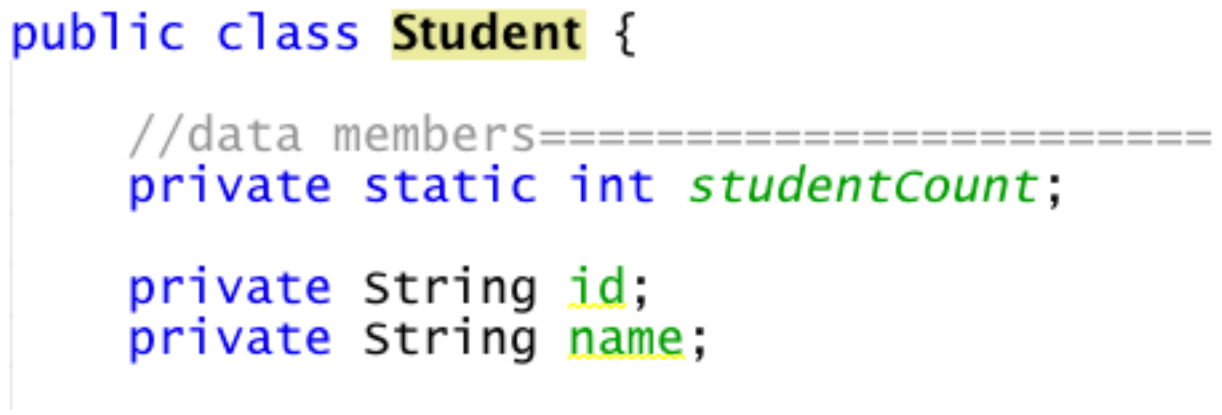
‘static’ vs ‘non-static’ data member

- Ex. Using static and non-static data member



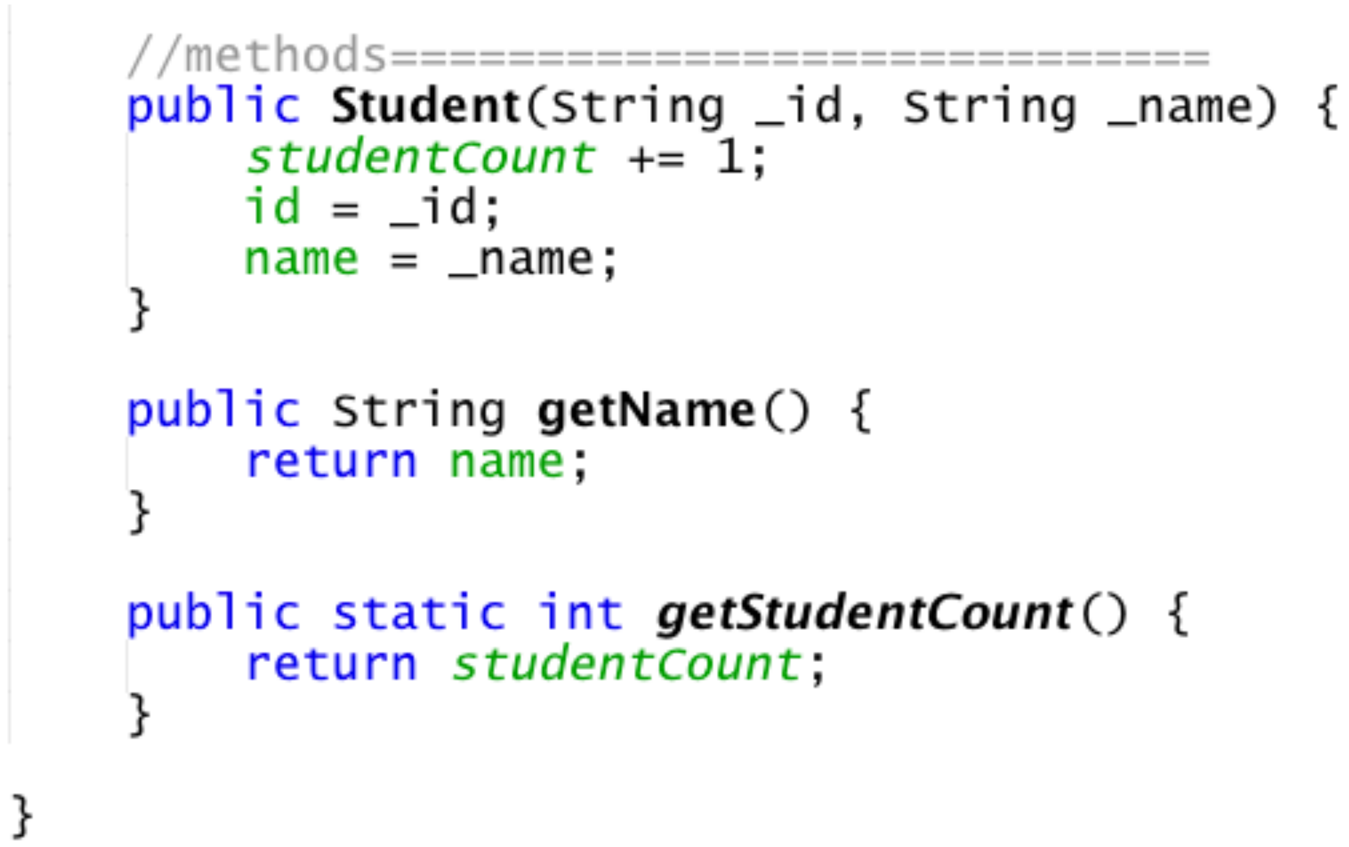
‘static’ vs ‘non-static’ data member

- Ex. Using static and non-static data member 2
  - Student Class



```java
public class Student {

    //data members=======================
    private static int studentCount;

    private String id;
    private String name;

```

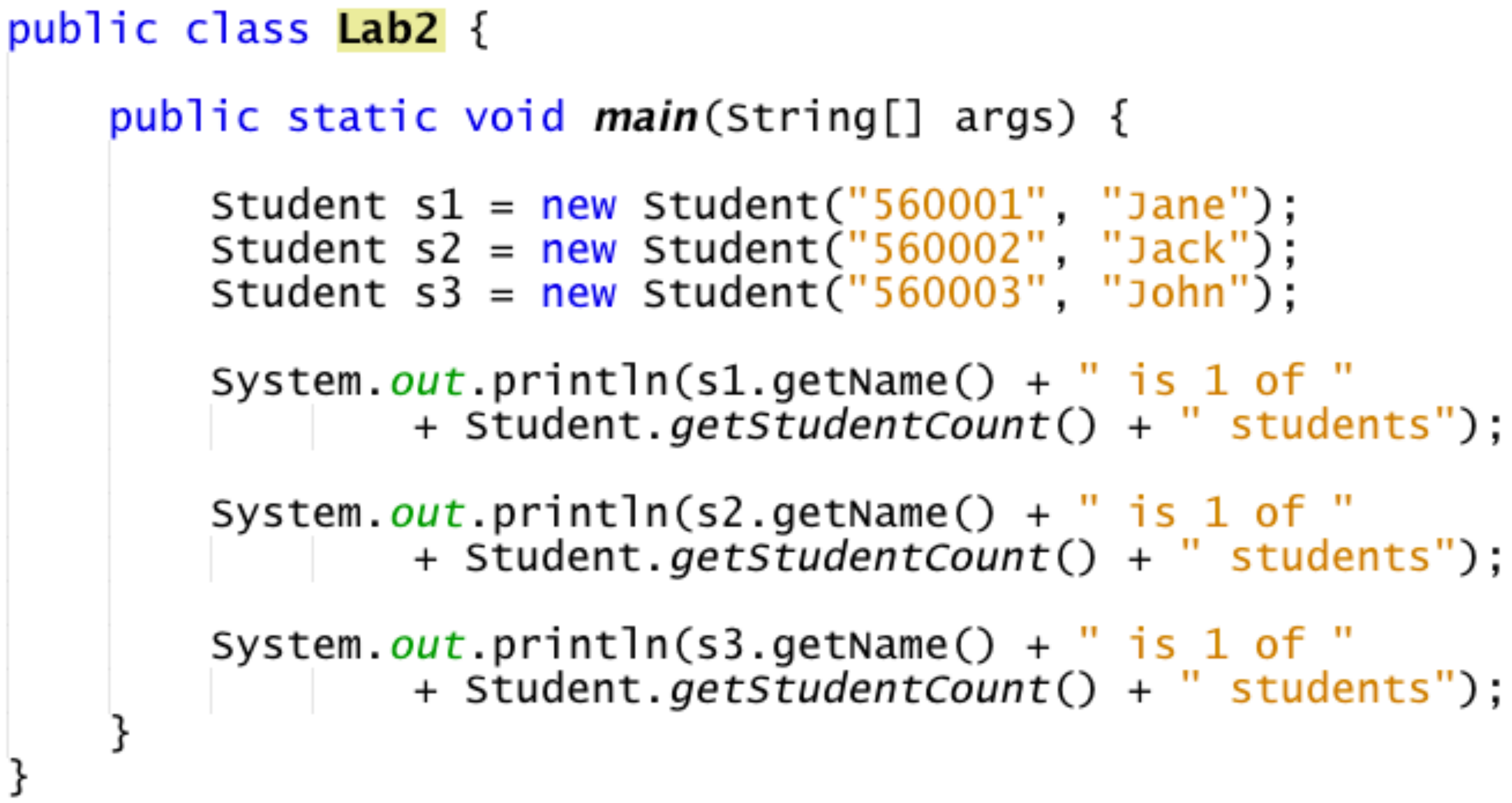
‘static’ vs ‘non-static’ data member



```java
    //methods==============================
    public Student(String _id, String _name) {
        studentCount += 1;
        id = _id;
        name = _name;
    }

    public String getName() {
        return name;
    }

    public static int getStudentCount() {
        return studentCount;
    }

}
```

‘static’ vs ‘non-static’ data member

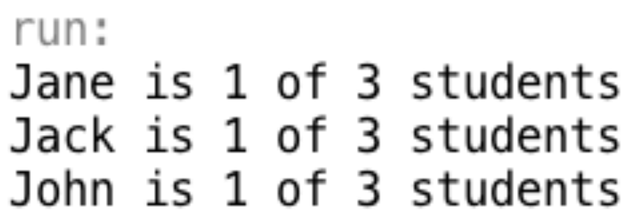
– Main Class



```java
public class Lab2 {

    public static void main(String[] args) {

        Student s1 = new Student("560001", "Jane");
        Student s2 = new Student("560002", "Jack");
        Student s3 = new Student("560003", "John");

        System.out.println(s1.getName() + " is 1 of "
                + Student.getStudentCount() + " students");

        System.out.println(s2.getName() + " is 1 of "
                + Student.getStudentCount() + " students");

        System.out.println(s3.getName() + " is 1 of "
                + Student.getStudentCount() + " students");
    }
}
```

‘static’ vs ‘non-static’ data member

– Output:



```
run:
Jane is 1 of 3 students
Jack is 1 of 3 students
John is 1 of 3 students
```

Question?