5.5: Introducing Access Control

IT1406 - Introduction to Programming

Level I - Semester 1

©2020 e-Learning Centre, UCSC

5.5. Introducing Access Control

- Encapsulation provides another important attribute: *access control*.
- Through encapsulation, you can control what parts of a program can access the members of a class. By controlling access, you can prevent misuse. For example, allowing access to data only through a welldefined set of methods, you can prevent the misuse of that data.
- How a member can be accessed is determined by the *access modifier* attached to its declaration. Java supplies a rich set of access modifiers. Some aspects of access control are related mostly to inheritance or packages. (A *package* is, essentially, a grouping of classes.)
- Java’s access modifiers are
  - **public**,
  - **private**
  - **protected**

© 2020 e-Learning Centre, UCSC

5.5. Introducing Access Control

- Java also defines a default access level.
- **protected** applies only when inheritance is involved.
- When a member of a class is modified by **public**, then that member can be accessed by any other code.
- When a member of a class is specified as **private**, then that member can only be accessed by other members of its class.
- Why **main( )** has always been preceded by the **public** modifier? Because, it is called by code that is outside the program—that is, by the Java run-time system.
- When no access modifier is used, then by default the member of a class is public within its own package, but cannot be accessed outside of its package.

© 2020 e-Learning Centre, UCSC

5.5. Introducing Access Control

- An access modifier precedes the rest of a member’s type specification. That is, it must begin a member’s declaration statement. Here is an example:

```
public int i;
private double j;
private int myMethod(int a, char b) { //…
```

- To understand the effects of public and private access, consider the following program:

```
//This program demonstrates the difference between public and private.
class Test {
    int a; // default access
    public int b; // public access
    private int c; // private access
    // methods to access c
    void setc(int i) { // set c's value
        c = i;
    }
    int getc() { // get c's value
        return c;
    }
}
class AccessTest {
    public static void main(String args[]) {
        Test ob = new Test();
        // These are OK, a and b may be accessed directly
        ob.a = 10;
        ob.b = 20;
        // This is not OK and will cause an error
        // ob.c = 100; // Error!
        // You must access c through its methods
        ob.setc(100); // OK
        System.out.println("a, b, and c: " + ob.a + " " + ob.b + " " + ob.getc());
    }
}
```

© 2020 e-Learning Centre, UCSC

5.5. Introducing Access Control

- As you can see, inside the **Test** class, **a** uses default access, which for this example is the same as specifying **public**. **b** is explicitly specified as **public**. Member **c** is given private access. This means that it cannot be accessed by code outside of its class. So, inside the **AccessTest** class, **c** cannot be used directly. It must be accessed through its public methods: **setc( )** and **getc( )**. If you were to remove the comment symbol from the beginning of the following line,

```
// ob.c = 100; // Error!
```

then you would not be able to compile this program because of the access violation.

© 2020 e-Learning Centre, UCSC