Northeastern University

Lecture 7: Object Oriented Programming - 2

Prof. Chen-Hsiang (Jones) Yu, Ph.D.

College of Engineering

Materials are edited by Prof. Jones Yu from

Liang, Y. Daniel. Introduction to Java Programming and Data Structures, Comprehensive Version, 12th edition, Pearson, 2019.

Outline

- Objects and Classes
- Thinking in Objects

Outline

- Objects and Classes
- Thinking in Objects

Objects and Classes

Static Variables, Constants, and Methods

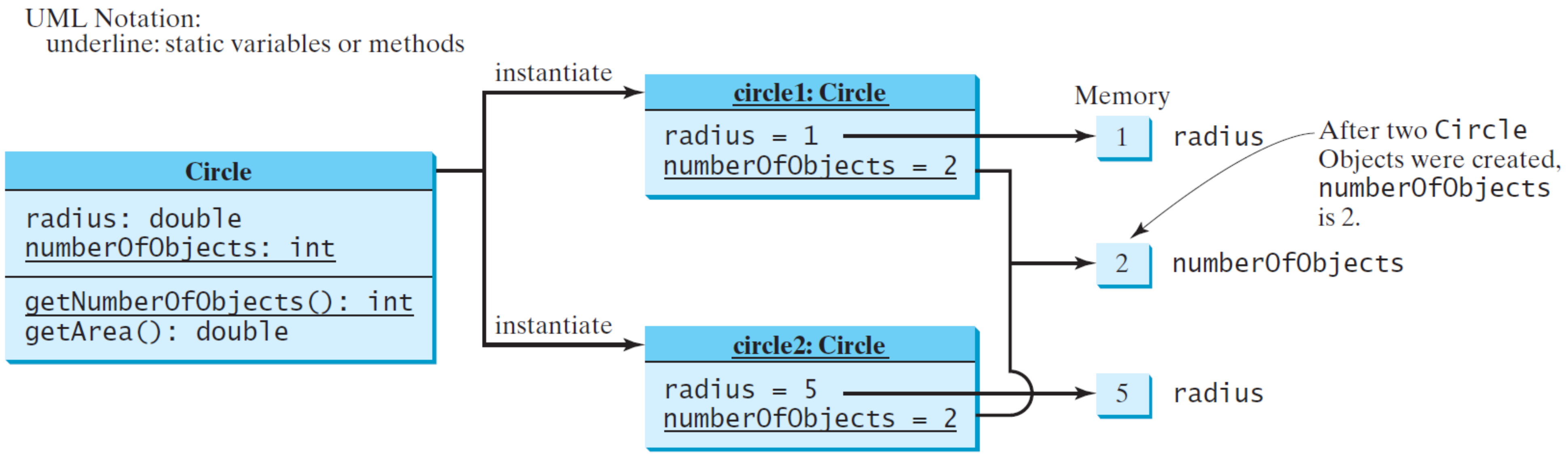
Instance Variables and Methods

- Instance variables belong to a specific instance (object).
- Instance methods are called by an instance of the class (object).

Static Variables, Constants and Methods

- Static variables are shared by all the instances of the class (objects).
- Static constants are final variables shared by all the instances of the class (objects).
- Static methods are not tied to a specific object. To call such method, using `ClassName.MethodName()`.
    - A static method cannot access instance members (instance data fields and methods).
- To declare static variables, constants, and methods, use the static modifier.

Static Variables, Constants and Methods



Exercise

- What is static variable?
- What's the difference between static variables and instance variables?

Answer

- Static variables:
  - » The variables that can be shared by all objects of a class.
  - » Static variables are also called *class variables.*
- Instance variables:
  - » The variables that belong to an object.
- Both static variables and instance variables are sometimes called *fields*, *data fields* or *data members.*
- Differences:
  - » Each object has one copy of its instance variables. They are not shared with other objects of the same class type.

Answer

Data Fields
(Member Variables)

Instance Variable

```
class City
{
   int count;
   ...
}
```

Class Variable

```
class City
{
   static int count;
   ...
}
```

Visibility Modifiers

Visibility Modifiers and Accessor/Mutator Methods

- By default, the class, variable, or method can be accessed by any class in the same package.
  - Public:
    - The class, data, or method is visible to any class in any package.
  - Private:
    - The data or methods can be accessed only by the declaring class.
    - The get (accessor) and set (mutator) methods are used to read and modify private properties.

The private modifier restricts access to within a class, the default modifier restricts access to within a package, and the public modifier enables unrestricted access.

```
package p1;

public class C1 {
  public int x;
  int y;
  private int z;

  public void m1() {
  }
  void m2() {
  }
  private void m3() {
  }
}
```

```
package p1;

public class C2 {
  void aMethod() {
    C1 o = new C1();
    can access o.x;
    can access o.y;
    cannot access o.z;

    can invoke o.m1();
    can invoke o.m2();
    cannot invoke o.m3();
  }
}
```

```
package p2;

public class C3 {
  void aMethod() {
    C1 o = new C1();
    can access o.x;
    cannot access o.y;
    cannot access o.z;

    can invoke o.m1();
    cannot invoke o.m2();
    cannot invoke o.m3();
  }
}
```

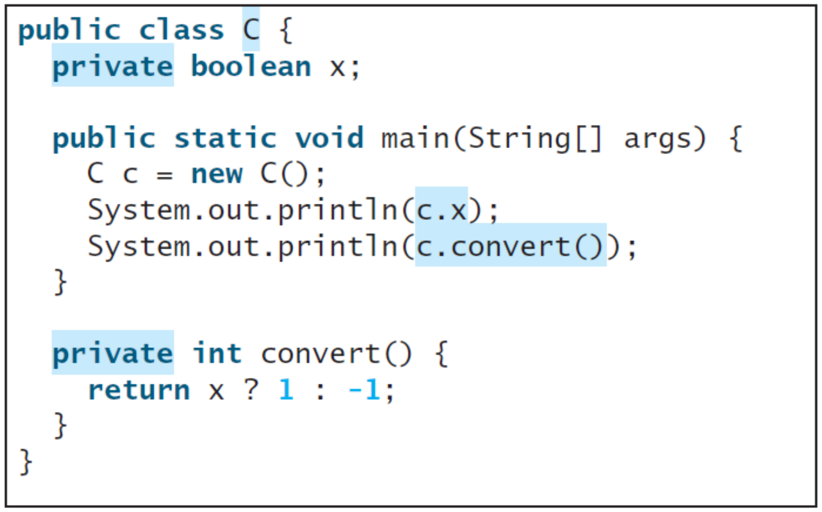
```
package p1;

class C1 {
  ...
}
```

```
package p1;

public class C2 {
  can access C1
}
```

```
package p2;

public class C3 {
  cannot access C1;
  can access C2;
}
```

Note

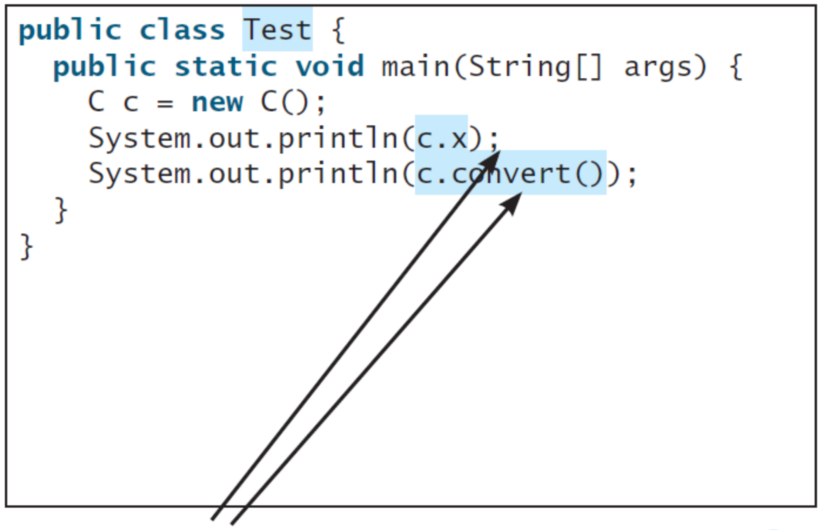
- An object cannot access its private members, as shown in (b). It is OK, however, if the object is declared in its own class. (a)



```
public class C {
  private boolean x;

  public static void main(String[] args) {
    C c = new C();
    System.out.println(c.x);
    System.out.println(c.convert());
  }

  private int convert() {
    return x ? 1 : -1;
  }
}
```

(a) This is okay because object c is used inside the class C.

OK



```
public class Test {
  public static void main(String[] args) {
    C c = new C();
    System.out.println(c.x);
    System.out.println(c.convert());
  }
}
```

(b) This is wrong because x and convert are private in class C.

Error

Exercise

- Type the following code into a Java project.
- Run the code, find the issue and fix the problem.

```java
public class Test {
    int x;

    public Test(String t) {
        System.out.println("Test");
    }

    public static void main(String[] args) {
        Test test = new Test();
        System.out.println(test.x);
    }
}
```

Test.java

Answer

- The program has a compile error because Test does not have a default constructor.