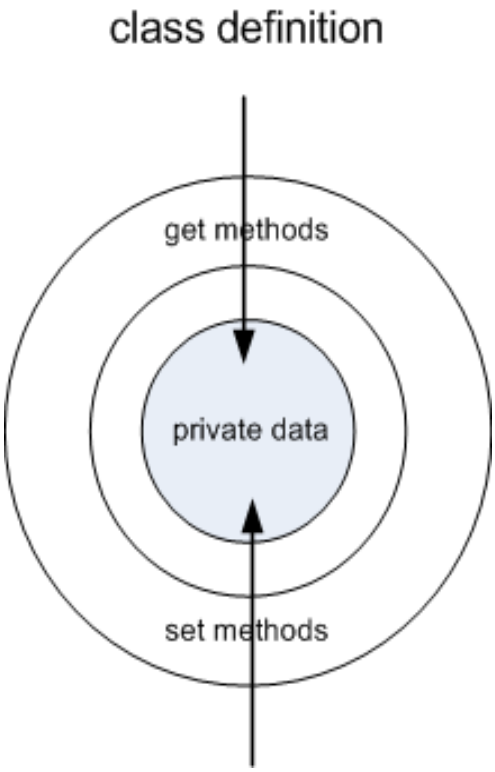
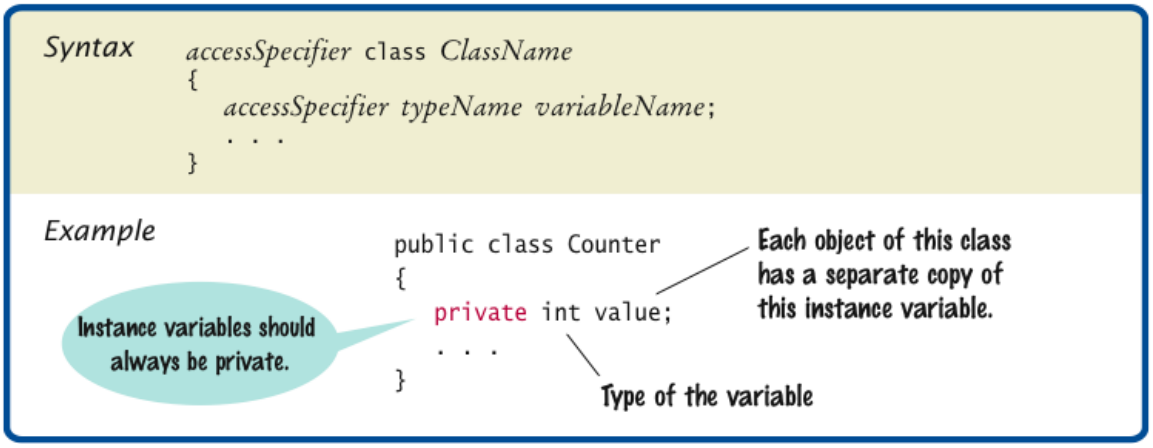
I. Classes

week 10/11

- a class is a specification (think of it as a blueprint or pattern and a set of instructions) of how to construct something
- classes will have names to identify objects created from that class
- classes may have private data, i.e. data that is private to that class, often referred to as **instance variables**
- protecting internal data fields within a class from external code access is **encapsulation**
- this is one of the fundamental principles of OOP
- classes may have method members, i.e. methods that work on the private class data





II. Creating and Manipulating Objects

- in general, there are 3 types of non-static methods for creating and manipulating objects

  1. **constructors**
  2. **accessors**, aka getters
  3. **mutators**, aka setters

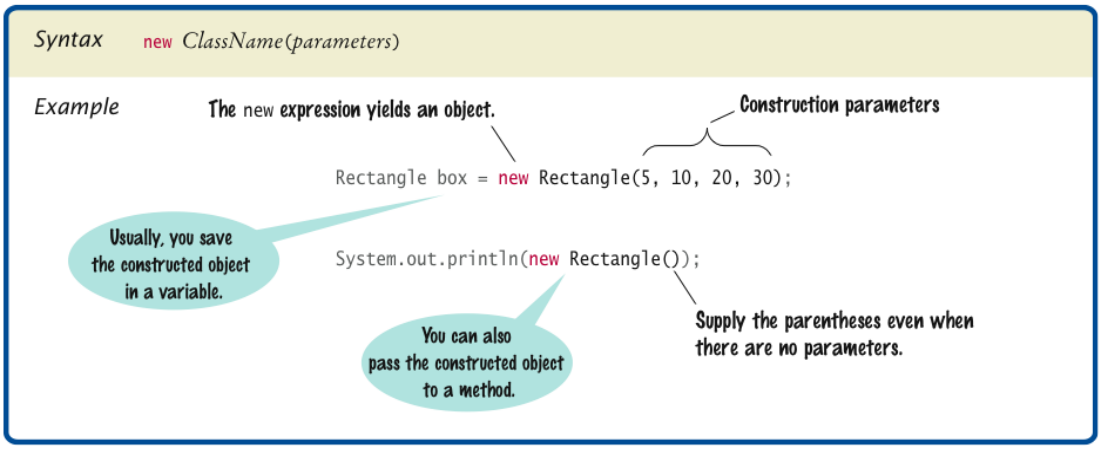
- non-static methods must be instantiated by creating a new object – each method then is a part of that specific object

- the process of creating a new object is called construction (see BJ, ch 8.6)

- constructing a new object uses the **new** operator

- the new operator returns an instance of the object, along with any non-static method members

- to actually use the newly created object, you'll need to assign it to a variable of type ???

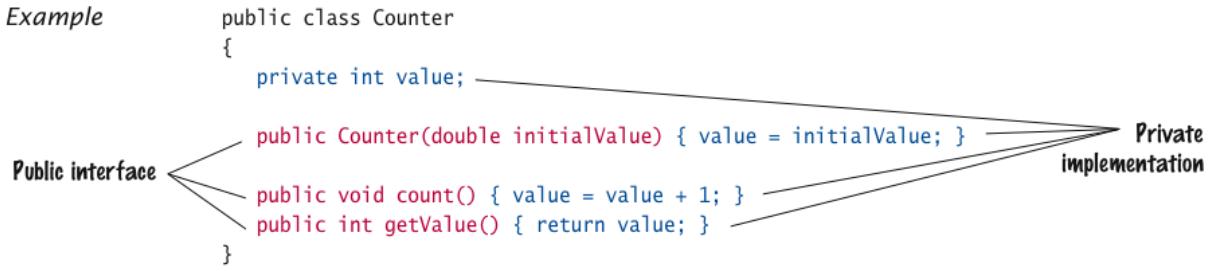


- constructors **must** have the same name and case as the name of the class
- constructors create objects that are ready-to-go (i.e. use)
- constructors with the same name and different parameter lists are said to be overloaded, the specific one executed is decided by its signature
- accessor methods usually are named with the word **get**, and usually return a specific value
- accessor methods should not change values, only return values
- mutator methods usually are named with the word **set**, and usually set a specific value
- either accessor or mutator can be omitted, i.e. omitting the set accessor creates a read-only property and omitting the get accessor creates a write-only property

Syntax

```
accessSpecifier class ClassName
{
   instance variables
   constructors
   methods
}
```

Example



How to create an object from a class

Figure 4-8 shows how to create an object with one or two statements. Most of the time, you can use one statement to create an object. However, in some situations, you need to use two statements to create an object.

When you use two statements to create an object, the first statement declares the class and the name of the variable. However, the object isn't created until the second statement is executed. This statement uses the new keyword to call the constructor for the object. This creates the object by initializing its instance variables, and it stores the object in memory. Then, the assignment operator (=) assigns a *reference* to this object to the variable. That's why objects are known as *reference types*.

When you send arguments to the constructor of a class, you must make sure that the constructor can accept the arguments. To do that, you must send the correct number of arguments, in the correct sequence, and with compatible data types. When a class contains more than one constructor, Java executes the constructor that matches the arguments.

The two-statement example in this figure creates a new Product object without passing any arguments to the constructor of the Product class. The first one-statement example accomplishes the same task. Then, the second one-statement example shows how to send three arguments to the constructor. Of course, this assumes that the Product class has a constructor with three parameters that are compatible with these three arguments.

How to create an object in two statements

Syntax

```
ClassName variableName;
variableName = new ClassName(argumentList);
```

No arguments

```
Product product;
product = new Product();
```

How to create an object in one statement

Syntax

```
ClassName variableName = new ClassName(argumentList);
```

No arguments

```
Product product = new Product();
```

Three arguments

```
Product product = new Product("java", "Murach's Java Programming", 57.50);
```

Description

- To create an object, you use the new keyword to create a new instance of a class. Each time the new keyword creates an object, Java calls the constructor for the object, which initializes the instance variables for the object and stores the object in memory.
- After you create an object, you assign it to a variable. When you do, a reference to the object is stored in the variable. Then, you can use the variable to refer to the object. As a result, objects are known as reference types.
- The variable for a *reference type* stores a *reference* to an object.
- To send arguments to the constructor, code the arguments between the parentheses that follow the class name. To send more than one argument, separate the arguments with commas.
- When you send arguments to the constructor, the arguments must be in the sequence called for by the constructor and they must have data types that are compatible with the data types of the parameters for the constructor.

Example:

```
public class Counter
{
        // private member data
        private int value;

        // constructor methods, 1 with an arg, 1 without
        public Counter () { value = 0 ; }
        public Counter (int initial_value) { value = initial_value; }

        // public mutator method - a setter method
        public void setCount () { value++;}

        // public accessor method - a getter method
        public int getValue() { return value; }

    public static void main (String[] args) {

        Counter clicker = new Counter();          // construct clicker object from Counter class

        clicker.setCount();
        clicker.setCount();

        System.out.println("The clickers value is: " + clicker.getValue());

     }
}
```

- see inputDialogExample program