Objects

Lecture 7

Dylan Boliske

Object Oriented Programming

- Class :
  - A template to define or manipulate an object
  - A tool for encapsulating data and operations into one package
- Object :
  - Data created by using a class and its methods
  - An object is an instance of a class
  - Creating an object is called instantiation

User-Defined Classes

- User-Defined Class : a class whose attributes and methods have been designed an implemented for a specific application
- Primitive data types (int, char, double, ...) are great, but in the real world, we deal with more complex objects: products, websites, flight records, employees, students, etc.
- Object-Oriented programming enables us to manipulate real-world objects.

User-Defined Classes

- Combine data and the methods that operate on the data
- Advantages:
  - Class is responsible for the validity of the data.
  - Implementation details can be hidden.
  - Class can be reused.
- Client of a Class : a program that instantiates objects and calls methods of the class

Categories of Data

- Instance Data : the internal representation of a specific object. It records the object's state.
- Class Data : accessible to all objects of a class.
- Local Data : specific to a given call of a method.

Instance Data

- Instance Data : the internal representation of a specific object.

```
public class Name {
    //Instance Variables

    private String first;

    private String middle;

    private String last;

    ...

}
```

Class Data

- Class Data : accessible to all objects of a class.
- Fields declared as static belong to the class rather than to a specific instance.

```
public class Name {
        //Class Constant

        public static String hyphen = “-“;

        ...

}
```

Local Data

- Local Data : specific to a given call of a method.
- The JVM allocates space for this data when the method is called and de-allocates it when the method returns.

```
public int compareTo(Name otherName) {
        int result;     //Local Variable

        ...

        return result;

}
```

Definitions

- Fields
  - Instance Variables : data for each object
  - Class Variables : data that all objects share
- Members
  - Fields and Methods
- Access Modifier
  - Determines access rights for the class and its members
  - Defines where the class and its members can be used

Instance

- Instance Fields : a field that exists in every instance of a class

```
String first;
String last;
```

- Instance Method : a method that exists in every instance of a class

```
void setName(String arg1, String arg2);
myName.setName("Tyler", "Durden");

Name yourName = new Name();

yourName.setName("Robert", "Paulson");
```

Class

- Class Field : a field that belongs to a class rather than its object instance; has modifier static
- Class Method : a method that belongs to a class rather than its object instance; has modifier static

Rules for Static and Non-Static Methods

	static Method	non-static Method
Access instance variables?	no	yes
Access static class variables?	yes	yes
Call static class methods?	yes	yes
Call non-static instance methods?	no	yes
Use the object reference this?	no	yes

Terminology

- Object Reference : identifier of the object
- Instantiating an Object : creating an object of a class
- Instance of the Class : the object
- Methods : the code to manipulate the object data
- Calling a Method : invoking a service for an object

Encapsulation

- Instance variables are usually declared to be private, which means users of the class must reference the data of an object by calling methods of the class.
- Thus the methods provide a protective shell around the data. We call this encapsulation.
- Benefit: the class methods can ensure that the object data is always valid.

Method Return Values

- Can be a primitive data type, class type, or void
- A value-returning method
  - Return value is not void
  - The method call is used in an expression. When the expression is evaluated, the return value of the method replaces the method call.
- Methods with a void return type
  - Have no value
  - Method call is complete statement (ends with ;)

Dot Notation

- Use when calling method to specify which object's data to use in the method
- Syntax:

  objectReference.methodName(arg1, arg2, ...)

  Note: no data types in method call; values only!

Access Modifiers

Access Modifier	Class or member can be referenced by...
public	methods of the same class, and methods of other classes.
private	methods of the same class only.
protected	methods of the same class, methods of subclasses, and methods of classes in the same package
No access modifier (package access)	methods of the same package only.

Public vs Private

- Classes are usually declared to be public
- Instance variables are usually declared to be private
- Methods that will be called by the client of the class are usually declared to be public
- Methods that will be called only by other methods of the class are usually declared to be private

Writing Methods

Syntax:

```
accessModifier returnType methodName(
parameterList)     //method head
{
    //method body
}
```

- parameterList is a comma-separated list of data types and variable names.
  - To the client, these are arguments
  - To the method, these are parameters

Constructors

- Constructor : special methods that are called when an object is instantiated using the new keyword.
- A class can have several constructors.
- The job of the class constructor is to initialize the instance variables of the new object.

Class Scope

- Instance variables have class scope.
  - Any method of a class can directly refer to instance variables.
- Methods also have class scope.
  - Any method of a class can call any other method of a class (without using an object reference)

Local Scope

- A method's parameters have local scope; meaning:
  - a method can directly access its parameters.
  - a method's parameters can not be accessed by other methods.
- A method can define local variables which also have local scope; meaning:
  - a method can access its local variables
  - a method's local variables can not be accessed by other methods.

Accessor Methods

- Clients can not directly access private instance variables, so classes provide public accessor methods with this standard form:

```
public returnType getInstanceVariable() {
    return instanceVariable;

}
```

(the returnType is the same type as the type of instanceVariable)

Mutator Methods

- Allow clients to change the values of instance variables

```
public void setInstanceVariable(dataType newValue) {
        //validate newValue,

        //assign to instanceVariable

}
```

Data Manipulation Methods

- Perform the “business” of the class.
- Example: a method to calculate miles per gallon:

```
public double calculateMilesPerGallon() {
    if (gallonsOfGas != 0.0) {

        return milesDriven/gallonsOfGas;

    } else {

        return 0.0;

    }

}
```

The Object Reference this

- How does a method know which object's data to use?
- this is an implicit parameter sent to methods and is an object reference to the object for which the method was called.
- When a method refers to an instance variable name, this is implied.

The toString Method

- Returns a String representing the data of an object
- Client can call toString explicitly by coding the method call.
- Client can call toString implicitly by using the object reference where a String is expected.

The equals Method

- Determine if the data in another object is equal to the data in this object

Return Type	Method Name and Argument List
boolean	equals(Object obj) returns true if the data is the *Object obj* is the same as in this object; *false* otherwise.

Composite Class Relationship

- Composite Relationship : a class that contains another class. (Also known as a “has a” relationship)
- A composite relationship has the following properties:
  - One class in the relationship can represent the whole.
  - The parts of the relationship exist only as long as the whole.
  - A part may only belong to one whole at a time.
- Example:
  - A Book class might have an instance variable for the publishing date, which would be an instance of a Date class.

Unified Modeling Language Class Diagrams

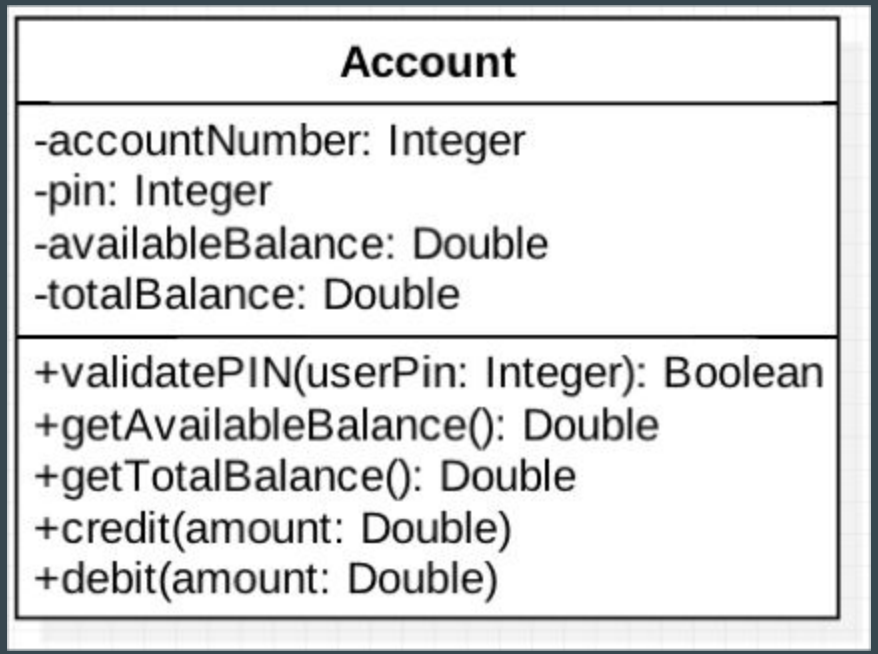
...

Unified Modeling Language (UML) Class Diagrams

- UML class diagrams model the classes and their relationships
  - Top compartment contains the name of the class
  - Middle compartment contains the class's attributes
  - Bottom compartment contains the class's operations

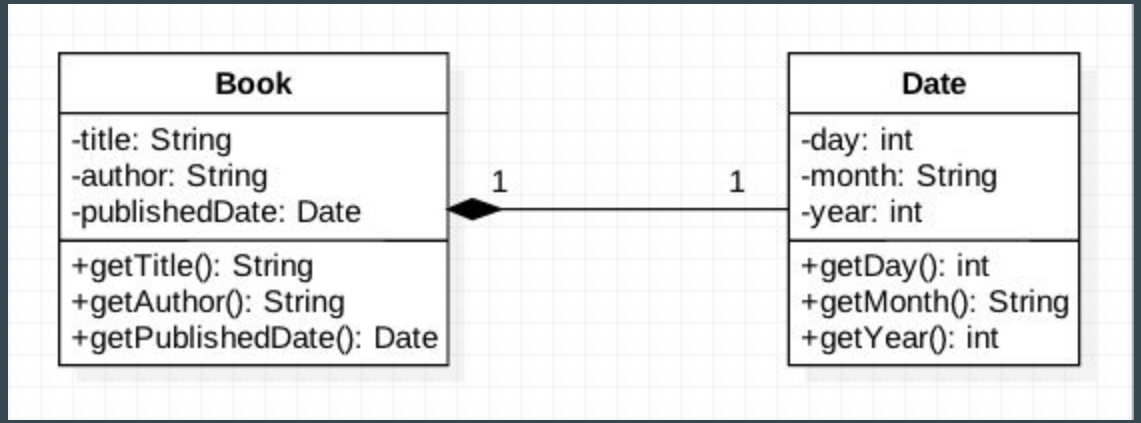
Class Operations

- An operation is a function that the object provides to clients of the class
- Operations are placed in the third section of the class diagram



Composite Relationship

- Shows how classes Book and Date relate to each other:
  - The line between the two classes shows the composite relationship.
  - Multiplicity values indicate how many objects of each class are involved in the composite relationship.



Multiplicity

Symbol	Meaning
0	None
1	One
m	An integer value
0..1	Zero or one
m, n	*m* or *n*
m..n	At least *m*, but no more than *n*.
*	Any nonnegative integer (zero or more)
0..*	Zero or more (identical to *)
1..*	One or more