Writing Classes - Introduction

CSC 1051 – Data Structures and Algorithms I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/
Writing Classes

• We've been using predefined classes from the Java API. Now we will learn to write our own classes.

• Chapter 4 focuses on:
  – class definitions
  – instance data
  – encapsulation and Java modifiers
  – method declaration and parameter passing
  – constructors
  – graphical objects
  – events and listeners
  – buttons and text fields
Bank Account: Example of user-defined class

acct1

acctNumber 72354
balance 102.56
name "Ted Murphy"

acct2

acctNumber 69713
balance 40.00
name "Jane Smith"
public class Account {
    final double RATE = 0.035; // interest rate of 3.5%

    long acctNumber;
    double balance;
    String name;

    // Sets up the account by defining its owner, account number, and initial balance.
    public Account (String owner, long account, double initial) {
        name = owner;
        acctNumber = account;
        balance = initial;
    }
}

continue

//-----------------------------------------------------------------
// Deposits the specified amount into the account.
//-----------------------------------------------------------------
public void deposit (double amount)
{
    balance = balance + amount;
}

//-----------------------------------------------------------------
// Withdraws the specified amount from the account and applies
// the fee.
//-----------------------------------------------------------------
public void withdraw (double amount, double fee)
{
    balance = balance - amount - fee;
}
// Adds interest to the account and returns the new balance.
public void addInterest ()
{
    balance += (balance * RATE);
}

// Returns the current balance of the account.
public double getBalance ()
{
    return balance;
}

// Returns a one-line description of the account as a string.
public String toString ()
{
    return (acctNumber + "\t" + name + "\t" + balance);
}
Account class: Creating objects

Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
Account acct1 = new Account ("Ted Murphy", 72354, 102.56);

Account acct2 = new Account ("Jane Smith", 69713, 40.00);
Account class: Using methods

```java
acct1.deposit (25.85);
```
Account class: Using methods

acct1.deposit (25.85);

// Deposits the specified amount into the account.
public void deposit (double amount)
{
    balance = balance + amount;
}
acct1.deposit (25.85);

// Deposits the specified amount into the account.
//
public void deposit (double amount)
{
    balance = balance + amount;
}
acct1.deposit (25.85);
Account class: Using methods

acct1.deposit (25.85);

// Deposits the specified amount into the account.
public void deposit (double amount) {
    balance = balance + amount;
}
Account class: Another Example

acct1.withdraw (60,2);

acct1

acctNumber 72354
balance 128.41
name "Ted Murphy"
Account class: Another Example

acct1.withdraw (60, 2);

// Withdraws the specified amount from the account
// and applies the fee.
public void withdraw (double amount, double fee)
{
    balance = balance - amount - fee;
}
acct1.withdraw (60, 2);

account acctNumber 72354
balance 66.41
name "Ted Murphy"

// Withdrawing the specified amount from the account and applies the fee.
public void withdraw (double amount, double fee)
{
    balance = balance - amount - fee;
}
Classes

• A class can contain data declarations and method declarations

```java
int size, weight;
char category;
```
Account Class

• A class can contain data declarations and method declarations

long acctNumber;
double balance;
String name;

Constructor
deposit()
withdraw()
toString()
public class Transactions1
{
    //--- Create some bank accounts and requests various services.---
    public static void main (String[] args)
    {
        Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
        Account acct2 = new Account ("Jane Smith", 69713, 40.00);
        Account acct3 = new Account ("Edward Demsey", 93757, 759.32);

        System.out.println (acct1);
        System.out.println (acct2);
        System.out.println (acct3);

        acct1.deposit (25.85);
        acct1.withdraw (60, 2);

        System.out.println ();
        System.out.println (acct1);
        System.out.println (acct2);
        System.out.println (acct3);
    }
}
public class Transactions1 {
    // -------------------------------
    //  Creates some bank accounts and requests various services.
    // -------------------------------
    public static void main (String[] args) {
        Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
        Account acct2 = new Account ("Jane Smith", 69713, 40.00);
        Account acct3 = new Account ("Edward Demsey", 93757, 759.32);

        System.out.println (acct1);
        System.out.println (acct2);
        System.out.println (acct3);

        acct1.deposit (25.85);
        acct1.withdraw (60,2);

        System.out.println ();
        System.out.println (acct1);
        System.out.println (acct2);
        System.out.println (acct3);
    }
}
## Examples of Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Attributes</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>Name</td>
<td>Set address, Set major, Compute grade point average</td>
</tr>
<tr>
<td></td>
<td>Address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Major</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Grade point average</td>
<td></td>
</tr>
<tr>
<td>Rectangle</td>
<td>Length</td>
<td>Set length, Set width, Set color</td>
</tr>
<tr>
<td></td>
<td>Width</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Color</td>
<td></td>
</tr>
<tr>
<td>Aquarium</td>
<td>Material</td>
<td>Set material, Set length, Set width, Set height, Compute volume, Compute filled weight</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Width</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Height</td>
<td></td>
</tr>
<tr>
<td>Flight</td>
<td>Airline</td>
<td>Set airline, Set flight number, Determine status</td>
</tr>
<tr>
<td></td>
<td>Flight number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Origin city</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Destination city</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Current status</td>
<td></td>
</tr>
<tr>
<td>Employee</td>
<td>Name</td>
<td>Set department, Set title, Set salary, Compute wages, Compute bonus, Compute taxes</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Salary</td>
<td></td>
</tr>
</tbody>
</table>
Next: Method Declaration

• Syntax for declaring a method – method header
• Method invocation - flow of control
• Return values
Method Header

- A method declaration begins with a *method header*

```java
char calc (int num1, int num2, String message)
```

- **method name**
- **return type**
- **parameter list**

The parameter list specifies the type and name of each parameter.

The name of a parameter in the method declaration is called a *formal parameter*.
Method Body

- The method header is followed by the *method body*

```java
char calc (int num1, int num2, String message)
{
    int sum = num1 + num2;
    char result = message.charAt (sum);

    return result;
}
```

The return expression must be consistent with the return type

*sum and result are local data*

They are created each time the method is called, and are destroyed when it finishes executing
Method Control Flow

• If the called method is in the same class, only the method name is needed
Method Control Flow

- The called method is often part of another class or object

```java
obj.doIt();
main

helpMe();
doIt

CSC 1051 M.A. Papalaskari, Villanova University
```
The return Statement

• The *return type* of a method indicates the type of value that the method sends back to the calling location.

• A *return statement* specifies the value that will be returned:

  ```java
  return expression;
  ```

• Its expression must conform to the return type.

• A method that does not return a value has a *void* return type.
Parameters

- When a method is called, the *actual parameters* in the invocation are copied into the *formal parameters* in the method header.

```java
ch = obj.calc (25, count, "Hello");

char calc (int num1, int num2, String message)
{
    int sum = num1 + num2;
    char result = message.charAt (sum);

    return result;
}
```
Bank Account Example

• There are some improvements that can be made to the `Account` class

• The design of some methods could also be more robust, such as verifying that the `amount` parameter to the `withdraw` method is positive

• Some of these improvements are in the book examples
  
  • `Account.java`, `Transactions.java` (simplified versions)
  
  • `Account.java`, `Transactions.java` (book versions)
Another example: The Die Class

- See RollingDice.java
- See Die.java
public class RollingDice
{
    // Creates two Die objects and rolls them several times.
    public static void main (String[] args)
    {
        Die die1, die2;
        int sum;

        die1 = new Die();
        die2 = new Die();

        die1.roll();
        die2.roll();
        System.out.println("Die One: " + die1 + ", Die Two: " + die2);
    }
}
continue

die1.roll();
die2.setFaceValue(4);
System.out.println ("Die One: " + die1 + ", Die Two: " + die2);

sum = die1.getFaceValue() + die2.getFaceValue();
System.out.println ("Sum: " + sum);

sum = die1.roll() + die2.roll();
System.out.println ("Die One: " + die1 + ", Die Two: " + die2);
System.out.println ("New sum: " + sum);
}
```java
Sample Run

Die One: 5, Die Two: 2
Die One: 1, Die Two: 4
Sum: 5
Die One: 4, Die Two: 2
New sum: 6
```

public class Die
{
    private final int MAX = 6;  // maximum face value

    private int faceValue;  // current value showing on the die

    public Die()
    {
        faceValue = 1;
    }
}
public int roll()
{
    faceValue = (int)(Math.random() * MAX) + 1;
    return faceValue;
}

public void setFaceValue (int value)
{
    faceValue = value;
}

public int getFaceValue()
{
    return faceValue;
}
public String toString() {
    String result = Integer.toString(faceValue);
    return result;
}
Homework

• Chapter 4, Sections 4.1, 4.2, 4.4

• Try the Alice tutorials (from http://www.alice.org - see links below)– here are some simple projects you can do for practice (from one of my other courses). They really help with understanding classes and methods.
  http://www.csc.villanova.edu/~map/1040/F10/lab12/lab12.html

• http://www.csc.villanova.edu/~map/1040/F10/lab13/lab13.html

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus

CSC 1051 M.A. Papalaskari, Villanova University