Designing Classes

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/
Where do objects come from?
Where do objects come from?

Good question!

Defining our own classes will make it possible to design our own objects

We need to learn:

1. How do we define our own classes?
2. How do we define what happens when an object is instantiated?
3. How do we define methods that can be invoked through our objects?
1. How do we define our own classes? Example: Defining the **Account** Class

- A class to represent a generic bank account

```java
long acctNumber;
double balance;
String name;
```

**Data declarations**

**Method declarations**

- Constructor
- deposit()
- withdraw()
- toString()
2. How do we define what happens when an object is instantiated?

Creating Objects – old example

- We have already seen something like this:

```java
scan = new Scanner (System.in);
```

This calls the Scanner *constructor*, which is a special method that sets up the object.
2. How do we define what happens when an object is instantiated?

Creating Objects – User-defined Account class

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

This calls the Account constructor, which is a special method that sets up the object.
3. How do we define what happens when an object’s method is invoked?

Invoking Methods

• We have seen that once an object has been instantiated, we can use the **dot operator** to invoke its methods

```java
ans = scan.nextLine();

numChars = title.length();
```

3. How do we define what happens when an object’s method is invoked?

Method invocation
acct1.deposit (25.85);

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

acct1
acctNumber 72354
balance 102.56
name "Ted Murphy"
Chapter 4: Writing Classes

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

    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;
}
public void addInterest ()
{
    balance += (balance * RATE);
}

public double getBalance ()
{
    return balance;
}

public String toString ()
{
    return (acctNumber + "\t" + name + "\t" + balance);
}
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);
    }
}
Transactions class:

Creating Account objects

Transactions class

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

Account class

```java
public Account (String owner, long account, double initial) {
    name = owner;
    acctNumber = account;
    balance = initial;
}
```

Transactions class

Constructor method

long acctNumber;
double balance;
String name;

Constructor

deposit()
withdraw()
transactions class: Creating Account objects

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

Account class

public Account (String owner, long account, double initial)
{
    name = owner;
    acctNumber = account;
    balance = initial;
}

Constructor method

CSC 1051 M.A. Papalaskari, Villanova University
Transactions class:
Creating more Account objects

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);

```java
public void deposit (double amount) {
    balance = balance + amount;
}
```

acct1

<table>
<thead>
<tr>
<th>acctNumber</th>
<th>72354</th>
</tr>
</thead>
<tbody>
<tr>
<td>balance</td>
<td>102.56</td>
</tr>
<tr>
<td>name</td>
<td>&quot;Ted Murphy&quot;</td>
</tr>
</tbody>
</table>
Account class: Using methods

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

```java
public void deposit (double amount) {
    balance = balance + amount;
}
```

```
acct1.deposit (25.85);
```

```
acct1

acctNumber: 7254
balance: 128.41
name: "Ted Murphy"

acct1.deposit (25.85);
```
Account class: Another Example

acct1.withdraw (60, 2);

acct1

acctNumber 72354
balance 128.41
name "Ted Murphy"
acct1.withdraw (60,2);

public void withdraw (double amount, double fee) {
    balance = balance - amount - fee;
}

acct1

acctNumber 72354
balance 128.41
name "Ted Murphy"
acct1.withdraw (60,2);

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

acct1

acctNumber: 72354
balance: 66.41
name: "Ted Murphy"
Class definitions

• A class can contain data declarations and method declarations

```java
long acctNumber;
double balance;
String name;
```

Data declarations

Constructor

deposit()
withdraw()
toString()

Method declarations
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)
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 (acct1);
        System.out.println (acct2);
        System.out.println (acct3);
    }
}
public class Transactions1
{
    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);
    }
}

Sample Run

<table>
<thead>
<tr>
<th>Account ID</th>
<th>Name</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>72354</td>
<td>Ted Murphy</td>
<td>$102.56</td>
</tr>
<tr>
<td>69713</td>
<td>Jane Smith</td>
<td>$40.00</td>
</tr>
<tr>
<td>93757</td>
<td>Edward Demsey</td>
<td>$759.32</td>
</tr>
</tbody>
</table>

72354  Ted Murphy      $102.56
69713  Jane Smith      $40.00
93757  Edward Demsey   $759.32

72354  Ted Murphy      $66.41
69713  Jane Smith      $40.00
93757  Edward Demsey   $759.32
# toString() Method

```java
public String toString ()
{
    NumberFormat fmt = NumberFormat.getCurrencyInstance();
    return (acctNumber + "\t" + name + "\t" + fmt.format(balance));
}
```

 acctNumber: 72354
 balance: 102.56
 name: "Ted Murphy"

```
System.out.println(acct1);
```
static methods

OtherClass.doSomething(acct1);

```java
public static void doSomething()
{
    System.out.println(" At your service. ");
}
```

OtherClass
# 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</td>
</tr>
<tr>
<td></td>
<td>Address</td>
<td>Set major</td>
</tr>
<tr>
<td></td>
<td>Major</td>
<td>Compute grade point average</td>
</tr>
<tr>
<td></td>
<td>Grade point average</td>
<td></td>
</tr>
<tr>
<td>Rectangle</td>
<td>Length</td>
<td>Set length</td>
</tr>
<tr>
<td></td>
<td>Width</td>
<td>Set width</td>
</tr>
<tr>
<td></td>
<td>Color</td>
<td>Set color</td>
</tr>
<tr>
<td>Aquarium</td>
<td>Material</td>
<td>Set material</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>Set length</td>
</tr>
<tr>
<td></td>
<td>Width</td>
<td>Set width</td>
</tr>
<tr>
<td></td>
<td>Height</td>
<td>Set height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute volume</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute filled weight</td>
</tr>
<tr>
<td>Flight</td>
<td>Airline</td>
<td>Set airline</td>
</tr>
<tr>
<td></td>
<td>Flight number</td>
<td>Set flight number</td>
</tr>
<tr>
<td></td>
<td>Origin city</td>
<td>Determine status</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</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td>Set title</td>
</tr>
<tr>
<td></td>
<td>Title</td>
<td>Set salary</td>
</tr>
<tr>
<td></td>
<td>Salary</td>
<td>Compute wages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute bonus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute taxes</td>
</tr>
</tbody>
</table>
Another example: The Die Class

- See RollingDice.java
- See Die.java
UML Class Diagrams

- A UML class diagram for the `RollingDice` program:

```plaintext
RollingDice

main (args : String[]) : void

Die

faceValue : int

roll() : int
setFaceValue (int value) : void
getFaceValue() : int
toString() : String
```
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
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);
```

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

    // ------------------------------------------------------------------------------------------------------------------
    // Constructor: Sets the initial face value.
    // ------------------------------------------------------------------------------------------------------------------
    public Die()
    {
        faceValue = 1;
    }
}
// ---------------------------------------------------------------
//  Rolls the die and returns the result.
// ---------------------------------------------------------------
public int roll()
{
    faceValue = (int)(Math.random() * MAX) + 1;
    return faceValue;
}

// ---------------------------------------------------------------
//  Face value mutator.
// ---------------------------------------------------------------
public void setFaceValue (int value)
{
    faceValue = value;
}

// ---------------------------------------------------------------
//  Face value accessor.
// ---------------------------------------------------------------
public int getFaceValue()
{
    return faceValue;
}

continue
continue

//-----------------------------
// Returns a string representation of this die.
//-----------------------------
public String toString()
{
    String result = Integer.toString(faceValue);
    return result;
}
}