



CSE 143 Java





Programming as Modeling

Reading: Ch. 1-6

1/29/2003
(c) University of Washington
01-1

Building Virtual Worlds

- Much of programming can be viewed as building a **model** of a real or imaginary world in the computer
 - a banking program models real banks
 - a checkers program models a real game
 - a fantasy game program models an imaginary world
 - a word processor models an intelligent typewriter
- Running the program (the model) simulates what would happen in the modeled world
- Often it's a lot easier or safer to build models than the real thing
 - Example: a tornado simulator

1/29/2003
(c) University of Washington
01-2

Java Tools for Modeling

- **Classes** in Java can model *things* in the (real or imaginary) world
 - The bank: Customers, employees, accounts, transactions...
 - Checkers: The Checkerboard, pieces, players, game history
 - Video game: Characters, landscapes, obstacles, weapons, treasure, scores
 - Documents: paragraphs, words, symbols, spelling dictionaries, fonts, smart paper-clip

1/29/2003
(c) University of Washington
01-3

Basic Java Mechanisms for Modeling

- A **class** describes a *template* or *pattern* for things; an **object** or **instance** is a *particular* thing
- **Constructors** model ways to create new instances
- **Methods** model *actions* that these things can perform
- **Messages** (method calls) model requests from one thing to another
- **Instance variables** model the state or properties of things
 - **public** vs. **private**
 - Instance variables should usually be private

1/29/2003
(c) University of Washington
01-4

What Makes a Good Model?

- Often, closer the model matches the (real or imaginary) world, the better
 - More likely it's an accurate model
 - Easier for human readers of the program to understand what's going on in the program
- Sometimes, a too detailed model of reality is not a good thing. Why?

1/29/2003

(c) University of Washington

01-5

What Else Makes a Good Model?

- The easier the model is to extend & evolve, the better
 - May want to extend the model...
 - May need to change the model...
- Sad law of life: "A Program is Never Finished"
- Why??

1/29/2003

(c) University of Washington

01-6

More Java Tools for Good Modeling

- One way to aid evolution is to define good **interfaces** separate from the implementation (code)
- An interface specifies to clients (users of the class) what are the operations (methods) that can be invoked; anything else in the class is hidden
 - Clients get a simpler interface to learn
 - Implementors protect their ability to change the implementation over time without affecting clients

1/29/2003

(c) University of Washington

01-7

Behavior vs. State

- A Java interface prescribes only behavior (methods, operations, queries)
- The state (properties) is not part of the interface
 - state is hidden, or accessible only through methods
- Example: Bank accounts have balances
 - Does this mean they must have a "balance" instance variable??
- Keeping behavior and state separate is an important aspect of design
 - important, and often difficult

1/29/2003

(c) University of Washington

01-8

Which is More Fundamental?

- Behavior or State?
- What do you think, and why?

1/29/2003

(c) University of Washington

01-9

The High vs. The Low

- Some aspects of system design are very high-level
- Yet... programming requires attention to low-level details
- This spectrum is one thing that makes our job hard
 - hard, and interesting

1/29/2003

(c) University of Washington

01-10

A Review Example

```
/** A Bank Account */

public class BankAccount {
    private double balance; // the current balance of the account
    private String ownerName; // the name of the person who owns this account
    private int accountNumber; // the account number of this account

    /** Create a new bank account with a zero balance and a unique account number
     * @param ownerName the name of the person who owns this account */
    public BankAccount(String ownerName) {
        this.ownerName = ownerName;
        this.balance = 0.0;
        this.assignNewAccountNumber();
    }

    ...
}
```

1/29/2003

(c) University of Washington

01-11

Bank Example (2)

```
...

/** Assign this account a new unique account number */
private void assignNewAccountNumber() {
    this.accountNumber = ...;
}

/** Return the current balance.
 * @return the current balance */
public double getBalance() {
    return this.balance;
}

...
```

1/29/2003

(c) University of Washington

01-12

Bank Example (3)

```
...

/** Deposit into account.
 * @param amount the amount to deposit
 * @return whether or not the transaction was successful */
public boolean deposit(double amount) {
    return this.updateBalance(amount);
}

/** Withdraw from account.
 * @param amount the amount to withdraw
 * @return whether or not the transaction was successful */
public boolean withdraw(double amount) {
    return this.updateBalance(- amount);
}

...
```

1/29/2003

(c) University of Washington

01-13

Bank Example (4)

```
...

/** A helper method that adds its argument to the balance, if it doesn't cause overdraft.
 * @param amount the amount to add to the balance (negative to withdraw)
 * @return whether or not the transaction was successful */
private boolean updateBalance(double amount) {
    if (this.balance + amount < 0) {
        // don't change the balance, if this would overdraw it. print an error message instead.
        System.out.println("Sorry, you don't have that much money to withdraw.");
        return false;
    } else {
        // update the balance
        this.balance = this.balance + amount;
        return true;
    }
}

...
```

1/29/2003

(c) University of Washington

01-14

toString: Recommended for All Classes

A method with this exact signature:

Public String toString();

```
/** Compute a string representation of the account, e.g. for printing out */
public String toString() {
    return "BankAccount#" + this.accountNumber +
        " (owned by " + this.ownerName + "); current balance: " + this.balance;
}
```

- Java treats toString in a special way
 - In many cases, will automatically call toString when a String value is needed:

System.out.println(myObject);

1/29/2003

(c) University of Washington

01-15

toString

- Good while debugging
System.out.println(myObject.toString());
- Secret Java lore:
 - All Objects in Java have a built-in, default toString method
 - So why define your own??

1/29/2003

(c) University of Washington

01-16

Another Good Practice

- Place a static method in each class, just for testing it.
 - No special name; could even be main().
 - Even simple tests are helpful
 - Run the test method every time the class is modified

```
/** A method to test out some of the BankAccount operations */
public static void test() {
    BankAccount myAccount = new BankAccount("Joe Bob");
    myAccount.deposit(100.00);
    myAccount.deposit(250.00);
    myAccount.withdraw(50.00);
    System.out.println(myAccount); // automatically calls myAccount.toString()
}

} // end of BankAccount
```

1/29/2003

(c) University of Washington

01-17

Required vs. Recommended

- writing toString is "recommended"
- creating test methods is "recommended"
- You've probably been given other recommendations:
 - comments, variable naming, indentation, etc.
 - Use this library, don't use that library
- Why bother, when the only thing that matters is whether your program runs or not?
 - Answer: Whether your program runs or not is *not* the only thing that matters!

1/29/2003

(c) University of Washington

01-18

Software Engineering and Practice

- Building good software is not just about getting it to produce the right output
- Many other goals may exist
- "Software engineering" refers to practices which promote the creation of good software, in all its aspects
 - Some of this is directly code-related: class and method design
 - Some of it is more external: documentation, style
 - Some of it is higher-level: system architecture
- Attention to software quality is important in CSE143
 - as it is in the profession

1/29/2003

(c) University of Washington

01-19