
CSE 142

Classes and Objects in Java

1/10/2003

(c) 2001-3, University of Washington

E-1

Outline for Today

- Review of objects and classes
- Bank account analysis and class design
- Class definitions in Java
- Rules and conventions
- Specifications and Implementations
- Specifying methods in Java

1/10/2003

(c) 2001-3, University of Washington

E-2

Objects Reviewed

- Objects have **properties** and **responsibilities**
- **Properties**
 - Sets of values
 - Have a specific type (simple or reference to an object type)
 - The current collection of property values is the object's state
- **Responsibilities**
 - The collection of messages the object understands – what it can do
 - Queries and commands

1/10/2003

(c) 2001-3, University of Washington

E-3

Classes Reviewed

- A pattern for a collection of similar objects is called a class
 - All objects in the class have the same properties and responsibilities
- Every object is an instance of some class
- The basic unit of programming in Java is a class definition
 - Specifies properties and responsibilities of instances
 - Individual objects are created as needed
- Each class defines a new type
 - Object properties can be references to other objects

1/10/2003

(c) 2001-3, University of Washington

E-4

Exercise

- Design a class to represent a simple bank account
 - What are the properties?
 - What are the responsibilities?
 - Commands?
 - Queries



1/10/2003

(c) 2001-3, University of Washington

E-5

Bank Account Design

- Properties

- Responsibilities (commands/queries)

1/10/2003

(c) 2001-3, University of Washington

E-6

Translating This to Java

- Class definition

```
/** Representation of a simple bank account */
public class BankAccount {
    ...
}
```
- Defines a class and gives it a name
- Between the braces { ... } we give details of
 - **Instance variables**: the properties of the object
 - **Methods**: sequences of Java code that carry out the object's responsibilities (commands and queries)
(In other programming languages these are sometimes called functions, procedures, or subroutines)

1/10/2003

(c) 2001-3, University of Washington

E-7

What Do We Have?

- When we finish translating our analysis to Java code...

- Did we have an object?
- Did we have a program?

1/10/2003

(c) 2001-3, University of Washington

E-8

Rules and More Rules

- In the class definition
 - `public class BankAccount { ... }`
 - There are some things we have freedom to choose
 - There are some things we have no choice in
 - This is a basic characteristic of programming
- Example: In a class definition like this, we must use curly braces `{ }`
 - We can't choose to use `[]` or `()` instead
- Example: we can't change the word order from `class BankAccount` to `BankAccount class`
- These are said to be rules of *syntax* or form

1/10/2003

(c) 2001-3, University of Washington

E-9

Identifiers – Names of Things

- In the class definition
 - `public class BankAccount { ... }`
 - BankAccount** is the name of the class
- We *do* have a choice about how we name the class – within limits
- Names in Java are called *identifiers*
- We'll see many uses for identifiers in programs

1/10/2003

(c) 2001-3, University of Washington

E-10

Not Just Any Name...

- Programming languages all have rules about what constitutes a legal identifier (name)
- In Java (and C, C++, etc.):
 - Combination of letters, digits, underscores (`_`) starting with a letter (\$ is also allowed, but best to avoid)
 - Must start with a letter
 - Case sensitive (`abc`, `Abc`, `ABC` are all different)
 - Details in the book
- May not be a *keyword* or reserved word that has a special meaning in Java
 - `class`, `public`, `if`, `for`, `int`, `double`, `boolean`, ...

1/10/2003

(c) 2001-3, University of Washington

E-11

Not Just Any Legal Name, Either

- Picking good names is an essential part of programming
- General rule of thumb: for names that describe classes (types), queries, and properties, use a noun phrase that describes instances of the class or the property
 - `accountNumber`, `totalSales`, `quantityInStock`, `getBalance`
 - Avoid cryptic, cute, or vague names
 - `"value"` or `"count"` contains no useful information
- For methods, use a verb phrase that describes action performed
 - `setBalance`, `deposit`, `withdraw`, `changeDate`

This advice is a convention, not a rule of Java

1/10/2003

(c) 2001-3, University of Washington

E-12

Naming Conventions



- A *convention* is a customary practice that falls just short of being a rule
- Example: Capitalization of identifiers
- Java has no syntax rule about when to choose a capital letter
- Java *programmers* almost universally follow this convention:
 - Instance variables and methods: begin with lower case letter
 - Class names: capitalized
- For now: A class named Foo should be in a file named Foo.java
 - Later we'll explain exceptions to this convention
- Please follow these conventions in CSE142!
 - Exercise: look at some Java code in the textbook and see if it follows these conventions

1/10/2003

(c) 2001-3, University of Washington

E-13

Comments in a Program

- Comments help the human reader; otherwise ignored
 - Essential to record information needed to understand the program that is not reflected directly in the code (design decisions, strategies, etc.)
- Two forms of Java comments
 - `//` the rest of the line following `//` is a comment
 - `/*` everything after `/*` is a comment until reaching this: `*/`
 - `/**` special comment form for documentation ("doc comments") `*/`

1/10/2003

(c) 2001-3, University of Washington

E-14

Comments in CSE142

- Good commenting is an art
 - Need to include essential information, but don't overdo it
- Java has an elaborate set of conventions for commenting
 - "JavaDoc"
 - Widely followed by professional programmers
- "Do I have to comment my program in CSE142?"
 - Indirect answer #1: You should *want* to comment every program you write, whether or not it's for 142
 - Indirect answer #2: Your work in 142 should communicate well to a human reader and show professionalism.

1/10/2003

(c) 2001-3, University of Washington

E-15

Specification vs Implementation

- Specification – view of the class as seen by *client* code that uses instances of the class
 - Often called the interface of the class (although the word interface has a particular technical meaning in Java, which we will get to eventually)
- Implementation – internal details
 - Client should not know anything about this
- Some specifications in real life
 - Automobile "user interface" – steering wheel, pedals, etc.
 - Electric power outlet

1/10/2003

(c) 2001-3, University of Washington

E-16

Specifying a BankAccount

- Class: BankAccount
- Queries
 - getAccountBalance
 - getAccountName
 - getAccountNumber
- Commands
 - setAccountName
 - setAccountNumber
 - deposit
 - withdraw
- Special "command": constructor – initialize new BankAccount instance when it is created

1/10/2003

(c) 2001-3, University of Washington

E-17

BankAccount Specification in Java

- In Java, the specification and implementation are given in a single file
- To create a class we start by writing the specification parts of methods (i.e., the operations available to client code)
- After specifying, we'll fill in the implementation details (next lecture)

1/10/2003

(c) 2001-3, University of Washington

E-18

Specifying Methods for Queries

• Example

```
/** return the current balance in this BankAccount */  
public double getBalance() { ... }
```

- "public" – defines this as part of the public specification
- "double" (or int, boolean, BankAccount, etc.) – defines the type of the value returned by this query
- "getBalance" – the name of the method; when a getBalance message is sent to a BankAccount object, this method will be used to carry out that responsibility

1/10/2003

(c) 2001-3, University of Washington

E-19

Specifying Methods for Commands

• Example

```
/** Transfer the given amount from otherAccount to this BankAccount */  
public void transfer(double amount, BankAccount otherAccount) { ... }
```

- "public" – same as for a query; this is part of the specification
- "void" – special keyword to identify this as a command that does not return a value
- "deposit" – the name of the method
- "double amount" and "BankAccount otherAccount" – these are parameters, pieces of information supplied when the object is given this command

Like the 5 in a "clap 5" message sent to an Actor

1/10/2003

(c) 2001-3, University of Washington

E-20

Constructors

- **Example**

```
/** Construct a new BankAccount */  
public BankAccount() { ... }
```

- Like a command, but no “void” keyword
- Every time a new BankAccount instance is created, the constructor is used to initialize the new object's state to some sensible value

1/10/2003

(c) 2001-3, University of Washington

E-21

Summary

- **Class Definitions are the unit of programming in Java**
 - Individual objects are created as instances of these classes
- **Program must follow certain rules and should follow certain conventions**
- **Specification vs Implementation**
 - What is publicly available to client code vs what is private information hidden inside the class
- **Specifications for class methods**
 - Queries
 - Commands
 - Constructors – a specialized kind of command

1/10/2003

(c) 2001-3, University of Washington

E-22