Important Stuff to Know As you Get Started

CSE 143

Winter 03

Brian Bershad

Office Hours: M 12.30:1.30, W: 12.30-1.30

Administration

Course Goals

- By the end of this course, there will be
  - No Java code you can't read.
  - Little code you can't modify.
  - Some code you can't extend.
  - Probably a lot of code you can't write.

A Simple Plan

- Watch me write a lot of code.
- Read a lot of code.
- Write a lot of code.
- Extend a lot of code.
- Write about a lot of code.
- Understand the point of every lectures

What You Should Know Now

- **Essential Java Programming**
  - **Fundamental programming paradigms**
    - **For, while, conditional, objects, drivers, etc…**
  - **Classes**
    - **Definitions**
      - **public/private**
      - **Methods**
      - **Constructors**
      - **instance variables**
    - **Inheritance**
      - **Subtypes**
    - **Interfaces**
  - **input/output, uwcse graphics library, common java classes**
- **Putting it into practice**
  - **writing small programs from scratch, (some of you) modifying moderate size programs**
  - **Compiling, running, etc**

Roadmap

- Part 1: Relationships
  - Classes, Interfaces, Packages, Handling Errors
- Part 2: Advanced Program Structuring Techniques
  - GUI, Common Class Paradigms, Powerful data structures
- Part 3: Algorithms and Data Structures
  - Big O, lists, stacks, recursion, trees, sorting,…
- Part 4: Parsing

Course Organization

- 3 lectures per week (MWF)
- Quiz section twice per week (T & Th)
  - Exercises, review, discussions, etc.
- Frequent quizzes
  - To keep you up with the reading and assignment instructions
  - To test mastery of current material
  - To provide TAs and me with feedback

Assignments

- Typically (but not always!) due Wed. night 9pm (electronic) and/or in sections or lecture Thursday or Friday morning (written)
- Primarily fairly substantial programming projects with written reports
- Maybe some shorter problems and programming drills
- Assignments will more complex than in CSE142
  - Assignment directions, too!
- No late assignments accepted
  - But be sure to talk with your TA about problems truly beyond your control like illness or family emergency so we know what happened.

Academic (Mis)conduct

- Goal: balance the following
  - *Learning:* each student must do the work to learn effectively
  - *Cooperation:* people learn best when they can cooperate with others
  - *Fairness* and *honesty:* Nobody should ever represent the work of someone else as their own or try to claim credit for it
- Tolerance: Zero.

Academic (Mis)conduct

- Policy
  - You must do assignments by yourself or with your assigned partner (unless explicitly stated otherwise in an assignment)
  - You may discuss general approaches and ideas with others, but
  - You *may not ever* give code to or receive code from others
- We check this and act when trouble is discovered
- Use your common sense and ask first if unclear
  - Rule of thumb: *any activity you engage in for the purpose of earning credit while avoiding learning, or to help others do so, is likely to be an act of academic misconduct* (from CSE dept. policy – see link on the web)

Exams & Quizzes

- Exams
  - 2 midterm exams in class; Dates TBD.
  - Final exam: Time set by the university, location tba
- The exams will not be given on any other days. Don't make plans which would take you away!
- Format: mixture of short answer, short essay, multiple choice, programming

Grading

- Grade distribution (subject to change)
  - 30% homework assignments and projects
  - 15% + 15% midterm exams
  - 25% final exam
  - 10% quizzes
  - 5% participation and service
- Class is curved
  - Median of final course grades is around 3.0
    - Maybe a bit higher when there are a lot of drops

Grading

- Assignment and quiz grading will be very coarse
- No partial points
- Typical scale: 4, 3, 2, 1, 0 for assignments and written reports
  - Occasionally may use 0..1 or 0..2, etc.
  - Mastery || Good Job! || On the Right Track || Honest Effort, but... || Really, Now!
  - Separate scores for Operation/Practice
    i.e., Yes! Style, clarity, readability matters
  - Written reports count as much as the actual code (being able to communicate what you do is a crucial skill)
- Quiz question grading: usually right or wrong

Resources to Help You Succeed

- Course staff
  - Your TA is your primary contact, but please feel free to talk to any of us
    - *Especially*: don't leave me lonely in office hours!
  - I'll try to be available right *after* class on Monday and Wednesday for as long as there are questions
    - but *before* class, it's panic time. Please forgive me in advance if I'm grouchy then.
  - Consultants in the IPL
    - A limited resource!
  - CLUE – MGH evening learning center
    - CSE143 will have a presence; still working out the details

More Resources

- Help each other! Form study groups, spend time on the discussion list, etc.
- Undergraduate advisors, for general questions about the CSE programs (Sieg 114)
- College of Engineering has some special resources for women and minorities
- Other university resources

For Reading and Study

- Lecture slides and course notes
  - Alert! Not all lecture material is on the slides!
  - Slides used will be posted on the web
    - NOT distributed in lecture
- Textbook: Next slide
- Other Material
  - Possibly handouts
  - All e-mail announcements, assignment descriptions, etc. should be considered required reading. They could even be tested on!

Textbooks

- Textbooks is a reference, not a roadmap.
- Textbook: Niño & Hosch, *An Introduction to Programming and Object-Oriented Design using Java,* Wiley, 2002
  - Alert! We may not follow the book very closely!
  - There will be reading assignments from this book.
    - If you choose not to buy it, be sure you have access to a copy
  - Covers material from both CSE142 & CSE143 – good review source
  - Will not always match our way of doing things, or our order!

Communicating Electronically

Course web site
- www.cs.washington.edu/143/

- Discussion Board: will be linked from Web site
  - UWNetID required
  - Open discussion – please contribute!
  - Course staff monitors and contributes as needed
- Email to us
  - Addresses on the web
  - Email works better for some things than other (e.g., very bad for trying to debug code)
- Anonymous Feedback Page
- E-mail from us
  - Sent directly to your UWNetID account
  - You are responsible for reading our postings.

Computing Facilities

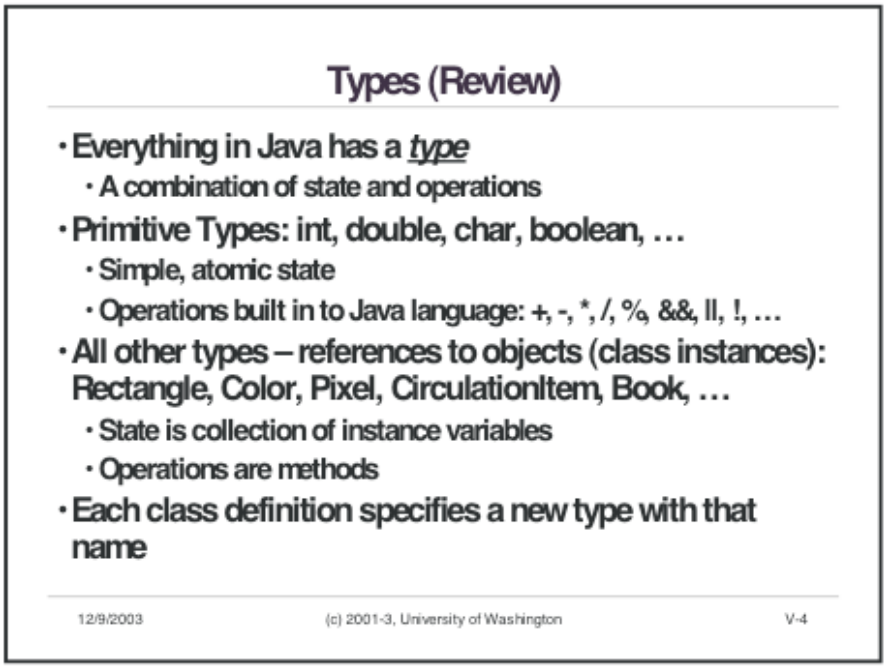
- Introductory Programming Lab (IPL)
  - Mary Gates Hall 334
  - CSE 143 consulting staff in IPL
    - Hours posted on the web
    - Goal is to provide quick help when you're stuck and have already tried to diagnose and fix the problem
- Computing at home
  - Java software and tools are freely available for download
    - Java version MUST be 1.4+ Install entire SDK (Windows, Linux), or run software update (Mac OS X)
    - You may use any Java development environment you like.
    - See Computing At Home page for links and details
- Even if you plan to compute at home, learn your way around the UW labs

Some Technical Review

Possibly some new stuff, but pretty important and pretty quick.

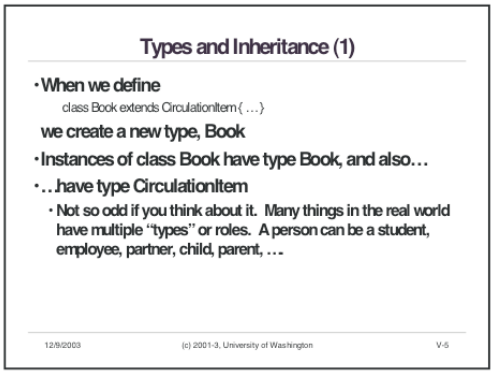
Core Java Concepts

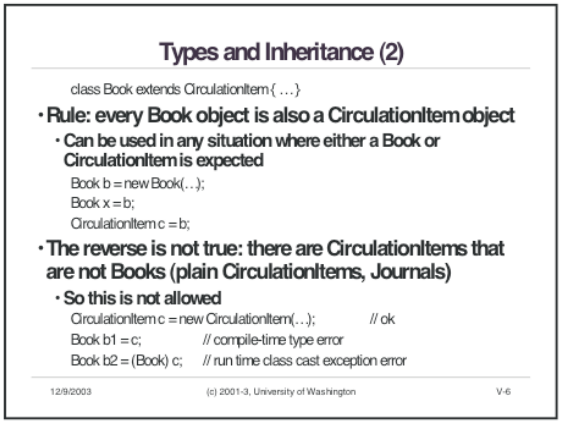
- A *class* describes a *template* or *pattern* for things; an *object* or *instance* of a class is a *particular* thing
- *Constructors* model ways to create new instances
- *Methods* model *actions* that these things can perform (i.e., to carry out their responsibilities)
- *Messages* (method calls)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



Inheritance

- A class can be defined as an extension of an existing class
  - **class MP3 extends Music {…}**
- MP3 is at least the same as Music in terms of its methods and responsibilities
  - **Can add new methods**
  - **Can change the definition of existing methods**





Dynamic Types

- What are the dynamic types of the variables in the following code?

```
Book b = new Book("Short Story", "A. U. Thor", "P34.56");
CirculationItem c = new CirculationItem("Rather Bland", "A1");
CirculationItem d = new Journal("Long 'n Boring", "Q45.367");
c = b;
```


The dynamic type of an object determines how the object behaves when you call it.

The static type of an object determines how you can use the object when you compile it.

Static Types and Methods

- If we declare a variable

  CirculationItem c = …

  the only guarantee we have is that it refers to some sort of CirculationItem
  - Compiler doesn't attempt to trace values assigned to variables to decide type information
  - So the only methods we can call using the variable c are the ones available in its static type (CirculationItem)

Method Override and Dynamic Dispatch

- When we extend a class, we can redefine a method that we would otherwise inherit from the original class
- The redefined method is said to *override* the original method definition
- When we call a method, the *dynamic type* of the object is used to select the appropriate method

```
CirculationItem c = new Book( …);
System.out.println(c);          // dynamic type of c here is Book, so
                                //   toString( ) from Book is used
```

  - This is called *dynamic (method) dispatch*

Dynamic Dispatch and Class Hierarchy Design

- Overriding and dynamic dispatch are powerful design tools
- Idea: when designing a class hierarchy, define in the original class methods which we want to be available for all objects in the hierarchy
- Use overriding to provide specialized implementations in extended classes
- Dynamic dispatch guarantees that the appropriate overriding methods will be called

Class Object

- The Java class structure has a root class: Object
- All Java classes implicitly extend Object if they don't explicitly extend some other class (which itself extends Object directly or indirectly)

```
class CirculationItem { …}
```

  means exactly the same thing as

```
class CirculationItem extends Object { …}
```

- Classes like ArrayList have parameters and results of type Object, so will handle any non-primitive type

```
public void add(Object obj) { …}
public Object get(int position) { …}
```

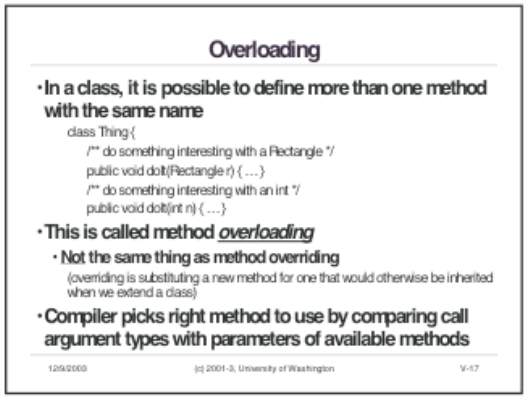

What's in Class Object?

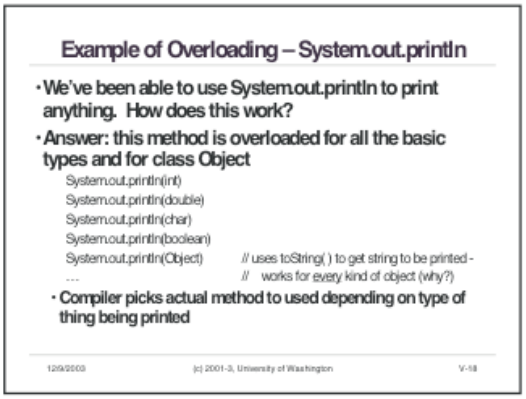
- Object contains methods (not many) that are suitable for all classes
- Class definitions can override these to provide more appropriate, specific versions
- Examples we've seen frequently
  - toString( )
  - equals( )

toString

- Good while debugging

  System.out.println(anObject.toString());
- Secret Java lore:
  - *All* Objects in Java have a built-in, default toString method
  - So why define your own??
- Question: how does the notion of dynamic/static typing reveal itself in toString().





Abstract

- A method is abstract if it defines a signature that has no implementation.
- A class is abstract if it includes an abstract method
  - or inherits one without implementing it.
- An abstract class can not be instantiated.
  - What would it do?
- Useful for defining the features of a class w/o concern for its implementation.
- See: http://java.sun.com/docs/books/tutorial/java/javaOO/abstract.html

JavaDoc

- Java provides a clean way of including documentation as part of the source code – JavaDoc comments
  - Begin with /** and end with */
- Can be automatically formatted to produce web documentation
  - Eg, "javadoc *.java"
- Special tags to control formatting
  - @author – specify author
  - @version – version number, date, etc.
  - @param – description of a method parameter
  - @return – description of a non-void method result
  - Others (links, see also, …), plus can use arbitrary html
- Used to produce all online Java API documentation