Slide 1

Administrivia

- Reminder: Homework 1 design due Tuesday (at 11:59pm).
- Open lab times (announced via e-mail): Tuesdays and Wednesdays 3:30pm to 5:30pm, in HAS 329. Completely optional but usually helpful.
- Code from class available via “Sample programs” page (soon after class).

Slide 2

A Little More About Homework 1

- You’re not committing yourself to anything at this point, but try to be as detailed as you can — so I can try to spot potential trouble. Also good to think in terms of a basic design (not too ambitious) plus extras. Keep in mind that what you do has to fit into an existing framework. (That’s actually one of the pedagogical goals.)
- What you will actually turn in is HTML documentation of your planned game’s main class — put it in your `Local/HTML-Documentation` and send me mail saying “ready to be graded”. (Complete instructions in homework writeup.)

Slide 3

Quick Tour of Eclipse — Recap

- Last time we used Eclipse to
  - Create a project.
  - Create a package within the project.
  - Create two classes.
  - Write some code and test it.
  - Generate HTML documentation. (There's a maybe-better way to do this than what we did last time: option under `Project` menu.)
- Code we wrote last time — "throwaway programs", often useful as you read about / hear about new things.

Slide 4

Compiling and Running Programs — Java Versus C/C++

- With C/C++, your program ("source code") is transformed by a compiler into . . .

  "object code" (different for different processors), which is combined with library object code to produce . . .

  an "executable" (different for different operating systems) that can be run like other applications.
- With Java, your program (source code) is transformed by a compiler into . . .

  "byte code" (same on any processor), which is executed by . . .

  "Java virtual machine" (which has access to library byte code).

Slide 5

Java Basics — Recap

- Java programs consist of classes. Each class can contain
  - Variables — instance and static.
  - Methods — instance and static.
  - Classes (more about this later).
- Variables and methods can be `public` or `private`.
- Variables and methods can be `final`. (Use `static final` for constants.)

Slide 6

Variables

- Primitive types provided for efficiency (not purely object-oriented):
  - `boolean`, `short`, `int`, `long`, `float`, `double` are pretty much as in C.
  - `char` is 16-bit Unicode.
  - `byte` is 8-bit byte.
- All other variables are *references to objects*, similar to pointers:
  - `MyClass x` creates a *reference*, not an object — use `new` to create objects.
    Type of `x` is `MyClass` (just as type of an `int` variable is `int`).
  - Value of `null` means it doesn't point to anything.

Java Syntax

- Basic syntax based on C — variable declarations, method definitions, expressions — with some additions (as discussed in class and in “From C to Java”).
- (This was by design.)

Slide 7

Creating and Deleting Objects

- Create object of class `MyClass` using `new` operator, e.g.,

  ```
  MyClass x = new MyClass();
  ```

  This object contains its own copy of all instance variables defined in `MyClass`.

  `new` above invokes no-parameters constructor for `MyClass`. Can have additional constructor(s) with parameters as desired.
- No need to explicitly free/delete objects — Java has “garbage collection”. (This may not seem remarkable unless you’ve used a language without it — e.g., C or C++.)

Slide 8

Referencing Objects, Variables, and Methods

Slide 9

- Within `MyClass`, reference members of class (variables and methods) using just their names. If you have multiple objects of this class, which one is meant? “current object”.
- In code using `MyClass`, reference as, e.g., `x.foo(parameters)` for instance methods, and `MyClass.staticFoo(parameters)` for static methods.

  Similar syntax for variables, but likely to be used less, since variables are normally private. (Exception is constants.)

Passing Parameters

Slide 10

- Syntax is like C.
- *Everything is passed by value* — but for reference variables, copying just creates two pointers to the same object, and the called method can change the object.

  (More about this later.)

Comments

Slide 11

- Can use C-style comments, C++-style comments.
- One type of C-style comments are special — “documentation comments” or “Javadoc comments”. These start with `/**` and end with `*/`, and the command-line tool `javadoc` turns them into HTML documentation similar to what Sun provides for the library functions.
- Use documentation comments to describe what people using your class need to know. Use other types of comments to document code itself — something that would be useful to humans reading it.

Control Structures

Slide 12

- Most control structures are the same as C — `if` , `while`, `do`, `switch`, `for`, etc. Also a simplified `for`, as of Java 5.0 (a.k.a. 1.5). More about it later.
- Also have “exceptions” — a way to deal with unusual or error conditions, break out of current flow of control. Can be “thrown” and “caught” (or not caught, in which case the program crashes). More about them later.

Arrays, Briefly

Slide 13

- Syntax is like C, except for explicit `new`:
  - `int[] x = new int[10];` creates 10 integers.
  - `String[] args = new String[20];` creates 20 *references* to strings.
- Arrays are “first-class” objects, with `length` variable.
- Java checks for out-of-bounds array references.

Miscellaneous Other Stuff

Slide 14

- No operator overloading (except “+” for `String` class).
- On reference variables, `=` and `==` operate on references, not objects. (So, you may instead want copy constructors or `equals()`.)
- No C-style strings, but a `String` class.
- A little about packages, and Java “generics” (new with 5.0), next time.

Example, Continued

- More work on `Account` class from last time.

Minute Essay

- Make your best try at writing a method for our `Account` class that computes one month's interest and adds it to the balance. Assume `interestRate` is the monthly interest.

Slide 17

Minute Essay

- Make your best try at writing a method for our `Account` class that computes one month's interest and adds it to the balance. Assume `interestRate` is the monthly interest.

Slide 18

Minute Essay Answer

- See the version of `Account` on the “Sample programs” page.