CS 240
Week 1

Intro

- Read the Web Page
- 4 Small Projects
  - Image Editor
  - Spell Checker
  - List'em
  - Evil Hangman
- Major Projects
  - Record Indexer Server
  - Record Indexer Client

Intro

- The Programming Test
- Tests: Midterm/Final
- Downloading Java
- Downloading Eclipse
  - Or this version

Getting Started in Java

- The class – No separate .h and .cpp files
  - public class
  - file name = class name
    - If the name of the class is “Stuff”, the files name is Stuff.java
  - Fields (or attributes) and methods
    - Similar to C++, But
      - Visibility designator on each
    - Static semantics review (similar to C++)
  - public static void main(String[] args) {…}
    - Similar to int argc, char* args
  - Simple I/O – System.out.println
  - Hello World example
  - System.out example

Simple Execution

- Command Line
  - Compilation to byte code
  - Running
    - Command line arg, e.g. –cp, -ea
- Eclipse
  - Compilation to byte code
  - Running

Java – C++ Similarities

- Control structure syntax: if, switch, for, while, do-while
- Primitive data types (some with slightly different names):
  char, short, int, long, **boolean**, float, double
  - Standard and consistent lengths
  - Can't treat boolean as int
  - We can treat chars as ints
  - No unsigned type
- Unary increment and decrement operators
- += and similar assignment operators
- Short circuit boolean expressions
- Arrays – difference in declaration and instantiation
  - Out of bounds error in Java
  - Length operator – names.length
- Basic variable declarations in classes
  - All non-built in types are pointers to objects
  - Must be initialized with "new" operation
- Basic method definitions in classes
- Class Person example

Java – C++ Differences

- Java is interpreted – byte code
  - JRE
  - Dynamic, run time linking
    - .class files
    - Notion of class path
      - From “Root”
      - User designated -- -classpath or –cp
  - Though the .class files can be in the same directory as the .java files, they are often placed elsewhere
    - Follow the same directory structure

Dynamic Variables and Garbage Collection

- Java has a “new” operator
  - Classes have constructors
  - “Null” value
  - See Person.java
- There is no delete operator nor destructors
  - Taken care of by garbage collector running in background

Method Declaration

- Signatures are similar but not the same
  - public, private, protected everywhere
  - final methods
- Overloading
  - All overloaded methods are virtual
- See Person Example

Project 1 Overview

- Command line syntax:

  java ImageEditor *inputFileName* *outputFileName* *transform*

  where the transform is one and only one of the following four words: invert, grayscale, emboss, motionblur

Image Concepts

- An Image is
  - A 2-dimensional array of pixels
  - Each pixel has a red, green and blue color value
  - Each color value is an integer whose value ranges from 0 to 255
- PPM format – Portable Pixel Map
  - Syntax – see project description

Transforms

- invert -- for every colorValue in every pixel

  colorValue = 255 – colorValue

- grayscale – for every pixel, the red, green, and blue values of that pixel are set to the average of the original color values of that pixel.

Transforms

- emboss
  - For every pixel, ***p***, that has an upper-left pixel, ***ulp***
    - Set the red, green, and blue values of **p** to:

      128 + the max(***p***.red-***ulp***.red, ***p***.green-***ulp***.green, ***p***.blue-***ulp***.blue)

      - If the computed value is < 0 store 0
      - If the computed value > 255 store 255
      - When finished, the three color values of every pixel will have the same value making the final image a grayscale image

Transforms

- Motionblur
  - Has a parameter ***length***
    - specified on the command line
  - For every pixel, ***p***, in row *r*, column *c* of the source image
    - For r and c we use 0-based index (i.e. the first column is column 0)
    - Let ***numberOfPixelsToTheRight*** be the minimum of the parameter ***length*** and (width of picture – c).
      - Example: If we are on column 10, length = 5, and the width of picture is 12, then numberOfPixels = min(5, 12 – 10) = 2
    - For every *colorValue* (red, green, or blue)
      - Sum the same color values in the pixels on row r of the image and in columns ***c*** *to* ***c + numberOfPixelsToTheRight - 1*** (inclusive)
        - » Using the above example, that would be columns 10 and 11
      - Set **p.colorValue** in the resulting image to

        sum/***numberOfPixelsToThe Right***

Decomposing Programs

- Most programs are decomposed into several classes
  - Notion of cohesion and coupling
  - Notion of class instance vs. class as object
    - Instance attributes and methods
    - Class attributes and methods (static)
- For most programs one class per file is suggested – it must be public
  - All other classes in the same file are non-public

Command Line Parameters

- Access to Command Line Parameters
  - String[] args
  - args[i]
  - args.length
- Example

A Simple Class

```
public class Person {
//Domain:
        private String name;
        private int age;
//Constructors
        public Person() {}
        public Person(String newName, int newAge){…}
//Queries
        public String getName(){…}
        public int getAge(){…}
        public Year getBirthYear(){
                return computeBirthYear();
        }
//Commands
        public void setName(String name){…}
        public void setAge(int age){…}
//Private Methods
        private Year computeBirthYear(){…}
}
```

Field Concepts

- Visibility: public, private, protected, *none*
- static
- final
- Initialization of fields
  - In their declaration
    - Arrays
      - int[] x = new int[5];
      - Int[] x = {1,2,5}; //similar to C++
    - Must be done in declaration if field is final
  - In the constructor
- Can be placed anywhere in class; doesn't have to appear before use

Method Declarations

- Signature followed by body
- Signature
  - Visibility
  - static – optional
  - Final (cannot be overridden)
  - return type
    - void – should modify object
    - type – should by query (i.e. not modify object)
  - Name
  - Parameter list

The main routine

- public static void main(String[] args) {…}
  - If it appears in class T then from command line we may execute “java T”
    - Not java T.class
    - In eclipse we execute it using “run as”
      - From run menu
      - After right clicking on class with main routine
- May exist in more than one class
- args[0] is third item on the command line

Packages
Organizing Classes

- All classes in directory are in same package
- Name of directory is name of package
  - The default package
  - For all but default package each file in package has
    package *directoryName*;
    as first line.
- Hierarchy of packages/classes
  - coalCombustion.oneDModel.Constants
- Classpath and packages
- Packages: User-defined, the java library, 3rd party software
  - Jars
- Import statement
  - import x.y.z;
  - import x.y.*;
  - Explicit reference to class in code – used to resolve conflict

Packages Example

- Source Files – In some directory

  Person.java

  support //a directory

    Age.java

    YearInSchool.java

- Class Files – In some other directory

  Person.class

  support //a directory

    Age.class

    YearInSchool.class

The Java Library

- Documentation
  - Demo
    - Packages
    - Classes
    - Index
    - Class Definitions

The Java Library

- Common library packages
  - java.io.*
    - java.io.File
    - Java.io.FileNotFoundException
  - java.util.*
    - java.util.Set<T>
      - java.util.HashSet<T>
    - java.util.Scanner
      - Construct using “System.in”
    - java.util.Map<K, V>
      - java.util.HashMap<K,V>
      - Entry<K,V>
    - java.util.List<T>
      - java.util.ArrayList<T>
  - java.lang.Math
    - java.lang.Math.min

Automatically Imported Classes

- java.lang.*
  - Atomic Types: Boolean, Byte, Character, Double, Float, Integer, Long, Short
    - autoboxing
  - Java.lang.System
    - InputStreams: System.in
    - PrintStreams: System.out, System.error
  - Math
  - Object
    - equals(Object obj)
    - hashCode()
    - toString()
  - String

Strings

- Literals
  - Special characters
- Immutable
- Concatenation – “+” operator
- == vs equals
- Useful methods
  - s.charAt() -- s. startsWith(String)
  - s.trim() -- s.indexOf(char)
  - s.substring(int) --s.substring(int,int)
  - Integer.parseInt(String) -- Integer.toString(int)
- Are not necessarily null terminated strings
- Cannot access them as arrays
- String Examples

Exceptions

- Defined for many built-in classes and methods
- Exception Types
  - NumberFormatException
  - FileNotFoundException
  - Need to import them
- Can define custom Exception types
- try
- catch
  - Multiple catches
    - Which one chosen – first one where parameter type matches exception type
      - Can cause problems if you are not careful
- propogation – like C++
- throws declaration in method
  - Required only of exceptions derived from Exception
  - Not required of throwable but no exception objects
- Exception Examples

I/O

- Design influenced by the decorator pattern
- The File class
  - Takes care of directory path syntax differences between different operating systems
- I/O Example
  - Input File for Example

I/O -- Output

- Design influenced by the decorator pattern
- The File class
  - Takes care of directory path syntax differences between different operating systems
  - FileNotFoundException
- Output
  - FileWriter – streams of characters
    - Common constructors – from File or file name
  - BufferedWriter – buffers file
    - (need not be FileWriter)
  - PrintWriter
    - Print and write simple types
    - System.out, System.err
  - print vs. println
    - System.out.println(“The current value of I = “ + i)
  - Automatic use of toString()
  - f.close()

I/O -- Input

- FileInputStream – reads bytes from File
  - Constructor – commonly from File or file name
  - FileNotFoundException
- BufferedInputStream – buffers it up
- Scanner – parses primitive types and strings using regular expressions
  - input.hasNextInt()
  - input.nextInt() throws IOException
  - input.hasNextLine()
  - input.nextLine() throws IOException

Bit Manipulation

- Usually applied to int, long, short, char, byte
- No unsigned types make it difficult
- Operators

  >>

  >>>

  <<

  &

  |

  ~

- Use of Hex representation for literals
- Bit manipulation example