CS206 Intro to Data Structures

Java Basics
Administrivia

• Course website
  ▫ www.cs.brynmawr.edu/cs206

• Homeworks
  ▫ Approximately weekly.
  ▫ Typically due on Thursday before midnight
  ▫ Help in lab Sunday-Thursday evenings
    ▫ starting next week

• Syllabus
  ▫ Subject to change
More Administrivia

• CS account
  □ If you do not have a cs account, sign up sheet
  □ If you do, make sure you can log in

• Lab: Park 231/TH 2:25pm-3:45pm

• Lab attendance is required.
  • Complete labs before starting assignments

• Software: Java and Eclipse
What is a Data Structure?
An Example Program

```java
public class Universe {
    public static void main(String[] args) {
        System.out.println("Hello Universe!");
    }
}
```
Components of a Java Program

• Name of main class and file must agree
  □ class Main <---> Main.java

□ Statements are placed in methods, that belong to class definitions.

• The static method named main is the first method to be executed when running a Java program.

• Any set of statements between the braces { and } define a program block.
Base/Primitive Types

- Variables must have types
- Primitive types define memory used to store the data

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>a boolean value: true or false</td>
</tr>
<tr>
<td>char</td>
<td>16-bit Unicode character</td>
</tr>
<tr>
<td>byte</td>
<td>8-bit signed two’s complement integer</td>
</tr>
<tr>
<td>short</td>
<td>16-bit signed two’s complement integer</td>
</tr>
<tr>
<td>int</td>
<td>32-bit signed two’s complement integer</td>
</tr>
<tr>
<td>long</td>
<td>64-bit signed two’s complement integer</td>
</tr>
<tr>
<td>float</td>
<td>32-bit floating-point number (IEEE 754-1985)</td>
</tr>
<tr>
<td>double</td>
<td>64-bit floating-point number (IEEE 754-1985)</td>
</tr>
</tbody>
</table>

```java
define variable:
boolean flag = true;
boolean verbose, debug;
char grade = 'A';
byte b = 12;
short s = 24;
int i, j, k = 257;
long l = 890L;
float pi = 3.1416F;
double e = 2.71828, a = 6.022e23;```
2s complement integers

The 2s complement of an $N$-bit number is defined as its complement with respect to $2^N$. For instance, for the three-bit number 010, the two's complement is 110, because $010 + 110 = 1000$. The two's complement is calculated by inverting the digits and adding one.

<table>
<thead>
<tr>
<th>Value</th>
<th>Binary</th>
<th>Complement</th>
<th>2s Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>111</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>100</td>
<td>111</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>101</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>100</td>
<td>101</td>
</tr>
<tr>
<td>-4</td>
<td>100</td>
<td>011</td>
<td>100</td>
</tr>
<tr>
<td>-3</td>
<td>101</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>-2</td>
<td>110</td>
<td>001</td>
<td>010</td>
</tr>
<tr>
<td>-1</td>
<td>111</td>
<td>000</td>
<td>001</td>
</tr>
</tbody>
</table>

Notes: leftmost bit is for sign in “Binary” column
Classes and Objects

• Every object is an instance of a class
• A class is a blueprint of what an object stores and how it functions
  ▫ instance variables
  ▫ methods
• Every variable is either a base type or a reference to an object
Creating and Using Objects

• In Java, a new object is created by using the `new` operator followed by a call to a constructor for the desired class.

• A constructor is a method that always shares the same name as its class. The `new` operator returns a reference to the newly created instance.

• Almost everything in Java is a class
Class Example

```java
public class Counter {
    private int count; // a simple integer instance variable
    public Counter() { } // default constructor (count is 0)
    public Counter(int initial) { count = initial; } // an alternate constructor
    public int getCount() { return count; } // an accessor method
    public void increment() { count++; } // an update method
    public void increment(int delta) { count += delta; } // an update method
    public void reset() { count = 0; } // an update method
}
```

- instance variable
- methods
  - constructor
  - accessor
Continued Example

```java
public static void main(String[] args) {
    Counter c;
    c = new Counter();
    c.increment();
    c.increment();
    System.out.println(c.getCount());
    c.reset();
    Counter d = new Counter(5);
    d.increment();
    Counter e = d;
    e.increment();
    System.out.println(c.getCount() + " " + d.getCount() + " " +
                       e.getCount());
}
```
Access Control Modifiers

- **public** designates that all classes may access

- **private** designates that access is granted only to code within that class.

- """" only classes within the package can access (I hate significant whitespace)
  - The package is generally the code you are working on.

- e.g., `System.out` is a package
Static

• When a variable or method of a class is declared as static, it is associated with the class as a whole, rather than with each individual instance of that class.
  • HH & HW example

• final
  • Variable
    • paired with static: set in class
    • not static: set in class or in every constructor
  • Method
    • Cannot be modified in subclasses
javadoc comments

• Comments
  □ /* */
  □ //

• A style/format of commenting for auto-generation of documentation in html
  /**
  */
  */
  □ used for method headers and classes
Example

/**
 * returns the sum of two integers
 * @param x The first integer
 * @param y The second integer
 * @return int The sum of x+y
 */

int sum(int x, int y)
Casting

• Assignment REQUIRES equal type

int x = 5;
double y = 1.2;
y = x;
x = y;
y = (double) x;
x = (int) y;
y = (double) x;

• Cast to change type
Implicit/Explicit Casting

- Widening cast – from a smaller/narrower type to a larger/wider - upcast
- Narrowing cast – the other way - downcast
- Java will perform an implicit cast when a widening is required, but not a narrowing
- Narrowing cast must be explicit