Introduction to Java Programming

Basic Syntax

Understand

Basic source file structure

Hello World

```java
/**
 * Simple "Hello World" text based application
 */
import java.io.*;

public class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```
Keywords & Identifiers

- Keywords are words that have specific meaning in Java (e.g., `public`, `import`, etc.).
- An identifier is a name given to a variable, class or method.
  - Can contain a-z, A-Z, 0-9, _, $, but cannot start with 0-9
  - Must not be a keyword
  - Case sensitive
  - Of any length

---

**Keywords**

- `abstract`  `extends`  `native`  `this`
- `boolean`  `false`  `new`  `throw`
- `break`  `final`  `null`  `throws`
- `case`  `finally`  `package`  `transient`
- `catch`  `for`  `protected`  `true`
- `char`  `goto*`  `public`  `void`
- `class`  `return`  `volatile`
- `const*`  `implements`  `short`  `while`
- `continue`  `import`  `static`
- `default`  `instanceof`  `strictfp`
- `do`  `int`  `super`
- `double`  `interface`  `switch`  `else`
- `long`  `synchronized`

---

**import**

- Some pre-defined classes exist for your use in creating Java programs.
- Pre-defined classes are grouped together into "packages" for distribution, i.e., `java.io`.
- The import statement may have many names separated by period:
  - `import java.awt.Graphics;`
  - `import java.io.*;`
- The asterisk (*) indicates that you wish to have access to all of the pre-defined classes in the `java.io` package.
- Import statements with the (*) are not recursive. Meaning, if we `import java.util.*`, and we want to use `java.util.jar.*`, the following will not include `java.util.jar.*`, even though it is located in `java.util`.
- `import java.util.*;`
import

- The classpath environment variable (or the -classpath option on the compiler) states where the directories containing packages on the disk are located.
- If the classpath is not set correctly, the JVM will not be able to locate your java class files, thus your program will not run.
- The package names correspond to a directory path to the file containing the classes desired, or a path to the classes in .jar or .zip files

import category.group.apackage.AClass;

Inside the "public class" :

```
/**
 * Simple "Hello World" text based application
 */

import java.io.*;

public class HelloWorldApp {
    public static void main (String[] arg) {
        System.out.println("Hello World");
        // Now we attempt to tell our age in celsius
    }
}
```

Class Identifier – an arbitrary name
Main method – all apps must have a main method
End Main
End Class
“The Problem”
Problems and data are not presented in a manner directly understandable by the machine.

Provide paychecks for all company employees.
Simulate the effects of an earthquake on the proposed high-rise office complex.
Assess effective power production of a variety of wind turbine generators.
Track a small business’ sales and inventory.
Process income tax returns from state residents.

Add 13 numbers together.

Simple algorithm

- Formula: celsius = (fahrenheit - 32) \* 5 / 9
- Java code:
```java
float fahrenheitAge = 45;
double celsiusAge;
celsiusAge = (fahrenheitAge-32) \* 5.0 / 9;
System.out.print("I am ");
System.out.print(celsiusAge);
System.out.println(" years old! (In Celsius)");
```

What’s an Algorithm?

- An algorithm are the logical steps taken to solve a problem.
- Problem Domain:
  - Need a Cake
- Algorithm:
  - 3 eggs
  - Tablespoon of oil
  - Stick of butter
  - Cup of Milk
  - 1) In a large mixing bowl, ....
  - 2) Preheat oven ....
Some Java rules

- Formula: \( celsius = \frac{(fahrenheit - 32) \times 5}{9} \)
- Java code:
  ```java
  float fahrenheitAge = 45;
  double celsiusAge;
  celsiusAge = (fahrenheitAge - 32) * 5.0 / 9;
  System.out.print("I am ");
  System.out.print(celsiusAge);
  System.out.println(" years old! (In Celsius)");
  ```

Variables must be declared before use. Java is case sensitive.

Variables have name, type, size, value (and life span)

Data Types

- Reference data type (Object)
  - Classes (will learn later)

- Primitive data type

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>1 bit</td>
<td><code>true</code>, <code>false</code></td>
</tr>
<tr>
<td>byte</td>
<td>8 bits</td>
<td><code>-128 .. +127</code> (INTEGER PRIMITIVE)</td>
</tr>
<tr>
<td>char</td>
<td>16 bits</td>
<td>Unicode! Not 8 bits like C</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td><code>-32,768 .. +32,767</code> (INTEGER PRIMITIVE)</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td><code>-2,147,483,648 .. +2,147,483,647</code></td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td><code>-9,223,372,036,854,775,808 .. +9,223,372,036,854,775,807</code></td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td><code>-3.4E38 .. +3.4E38</code> (6-7 significant figures accuracy)</td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td><code>-1.7E308 .. +1.7E308</code> (13-14 digits accuracy)</td>
</tr>
</tbody>
</table>

Data Types
Data Types

- Variables have name, type, size, value (and life span)
- int/ int = ?
- value + value = incorrect value?

Type casting

- Syntax: (Type) original type
- quotient = (double) sum / count;
- intVar = (int) longValue;
- longVar = (long) doubleValue

Code sample of type conversion

```java
int a;
long b;
double c;
b = a; //this is ok b is larger
a = b; //bad, a is smaller
a = (int) b; //ok cast, may lose data
c = b; //ok
b = c; /*java does not support implicit
conversion like this*/
b = (long) c; //but this is ok, but may lose
data
```

Bug Alert:
Type conversion side effects

```java
byte a, b, c;
a = b + c; /*this would produce a compile
time error since b and c would be
converted to ints and result of b + c
would be an int*/
a = (byte)(b + c); //this is ok
a += b; //this is ok
```
Widening Conversions

- When evaluating an arithmetic expression with two operands, the compiler converts primitives by widening according to these rules:
  1) If either is of type `double`, the other is converted to `double`.
  2) Otherwise, if either is a `float`, the other is converted to `float`.
  3) Otherwise, if either is of type `long`, the other is converted to `long`.
  4) Otherwise, both operands are converted to `int`.

Arithmetic

**Binary Operators - need two operands**
- Assignment: `=`
- Addition: `a + b`
- Subtraction: `c - 10`
- Multiplication: `c * 0`
- Division: `45 / 5`
- No exponent operator
- Modulus: `8 % 3`

* Remember the cast operator, `(type)`, has precedence above `*, /, %`

Numeric Literals

- Dec 25 = Oct 31
  ```java
  int myVar = 25; // decimal
  int myVar = 031; // octal
  int myVar = 0x19; // hex
  ```
- Forcing integer to `long` storage: `1l, 1L, 79l`
- Floating point literals: `563.84 == 5.6384e2`
- Forcing to `float`: `3.142657e2f`
Increment & Decrement operators -

- ++ (a++) is same as a=a+1
- postincrement: a++;
- preincrement: ++a;
- -- (a-- is same as a=a-1)
- also has postincrement and preincrement
- Example: a=12; b=34; c=a+b++;
- PLEASE: Don’t write code like this!
- c = (--a) + b++;
- Why not c = (a-1)+b; b = b+1;
- Why not c = (--a) + b++;

More assignment operators

- += (a+=5 is a=a+5)
- -= (a-=4 is a=a-4)
- *= (a*=3 is a=a*3)
- /= (a /=n is a=a/n)
- %= (b%=y is b=b%y)
- Example:
  - a=30; b=34;
  - a *= (b - 4);

Arithmetic Precedence

- Priority 1: () - negation
- Priority 2: * / %
- Priority 3: + -
Quick Practice

- If you were a computer, what would you give as a result for:
  \[ y = 2 \times 5 \times 5 + 3 \times 5 + 7; \]

Quick Practice Result

Quick Practice

- If you were a computer, what would you give as a result for:
  \[ y = 2 \times 5 \times 5 + 3 \times 5 + 7; \]
- What is the calculation order of:
  \[ z = p \times r \% q + w / x - y; \]

Java: \[ z = p \times r \% q + w / x - y; \]
Statement

- Statement: the most basic building block, causes Java to perform a single action or operation (often ends w/ ;)
- Java code:

```java
float fahrenheitAge = 45;
double celsiusAge;
celsiusAge = (fahrenheitAge - 32) * 5.0 / 9;
System.out.print("I am ");
System.out.print(celsiusAge);
System.out.println(" years old! (In Celsius)");
```

Special things on String

```java
/** "Hello World" simple applet */
import java.applet.*;
import java.awt.Graphics;
public class HelloWorld extends Applet {
    public void paint(Graphics g){
        int number = 7;
        g.drawString("Hello World", 25, 25);
        g.drawString(number, 25, 50);
        g.drawString(" + number", 25, 75);
        g.drawString("sum of 2 & 9 = " + 2 + 9, 25, 99);
    }
}
```

Statement diagram

```
quotient = 0.0;
add(prompt1);
add(prompt1);
add(prompt1);
```
Break time

- After break
- Coding Convention for Homework
- Comments
- JavaDoc

Hello World (applet)

```java
/**
 * "Hello World" simple applet
 */
import java.applet.*;
import java.awt.Graphics;

public class HelloWorld extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello World", 25, 25);
    }
}
```

HTML file for applet

```html
<html>
<body>
    <applet code="HelloWorld.class" width=275 height=55>
    </applet>
</body>
</html>
```
Conditional Statement: if()

- Syntax:
  - if (data1 cond.operator data2) {
    - statement1;
    - statement2;
    - ...
  }
Character Literals

- Single unicode character appear in "'
- Escape sequence
  - ' Single Quote
  - " Double Quote
  - \ Backslash
  - \b Backspace
  - \r Carriage Return
  - \n New Line
  - \t Horizontal Tab
  - \f Form Feed
  - \u{ddd} Unicode
  - \d{ddd} Octal char value

if structure
(pick up from where we left off)

- Syntax:
  - if (data1 cond.operator data2) {
    - statement1;
    - statement2;
    - ...
  }

Comments

- C style comment
  - /* comment */
  - /* All character between /* and */ are ignored */
- C++ style comment
  - // comment after the // up to end of line
- Java blend for javadoc documentation
  - /** comment */
if structure diagram

![Diagram of if structure]

if / else structure

- Syntax:
  - if ( logical condition ) {
    - statement1;
    - statement2;
    - ...
  } else {
    - statementA;
    - statementB;
    - ...
  }

if / else structure diagram

![Diagram of if / else structure]
Common if problems

- if ( count == 0 )
  - System.out.println( "Divided-by-Zero" );
- if ( count != 0 );
  - System.out.println( "Good divisor" );
- if ( count != 0 )
  - if ( sum != 0 )
    - System.out.println( "Both numbers are not zero" );
  - else
    - System.out.println( "Divided-by-Zero" );

Logical Operators

- if ( (count != 0) && (sum !=0) )
  - System.out.println( "Both numbers are not zero" );
- else if ( count==0 )
  - System.out.println ( "Divided -by-Zero" )
- Logical AND && truth table:
- Logical OR || truth table:
- Example: question = ( (bb) || !(bb) );
- Short-circuit evaluation

Bitwise Operators

- & (bitwise AND)
  - 0010101
  - & 0000011
- | (bitwise OR)
  - 0010101
  - | 0000011
- ^ (bitwise XOR)
  - 0010101
  - ^ 0000011
while structure

- Syntax:
  - `while (logical condition) {
    statement1;
    statement2;
    ... loop body ...
  }`
After break: Java Application

- public class AppName {
  - public static void main (String args[]) {
    - ...
  - }
- }

Compile with JavaC as usual
Run with JavaAppName

JDB: Java Debugger

- Compile with javac -g ClassName.java
- Run in debug mode
  - Use jdb ClassName to debug application.
  - Use appletViewer -debug Container.html to debug applet.
- Typical jdb commands: help, exit, list, stop at <class id>:<line>, clear <class id>:<line>, step, cont, print, dump
- jdb requires TCP/IP to work!

- brippe@fullcoll.edu - intro java
- http://staffwww.fullcoll.edu/brippe/cis226