Introduction to Java
E177
April 29, 2008
http://jagger.me.berkeley.edu/~pack/e177

Copyright 2005-8, Andy Packard, Trent Russi. This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/2.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

A simple application (StringFormatDemo.java)

This application illustrates the.

```java
class StringFormatDemo {
    public static void main(String[] args) {
        double A, B;
        A = 14.2;
        B = -9;
        String s1 = String.format("B=%d, A=%g, sqrt(A)=%g", B, A, Math.sqrt(A));
        s2 = String.format("Length of s1 = %d", s1.length());
        System.out.println(s1);
        System.out.println(s2);
    }
}
```

Format specifiers

The "standard output" object `PrintLine` method
Input arg

Direct call to "format" method of String class

Direct call to "sqrt" method of Math class

Object of class String

Method of String class
Input arg to method

Recall in Matlab, the object is an explicit input argument to method...

A simple program (StringFormatDemo2.java)

This application illustrates the.

```java
class StringFormatDemo2 {
    public static void main(String[] args) {
        String s1, s2, s3;
        double A = 14.2;
        int B=-9, i, L1;
        s1 = String.format("B=%d, A=%g, sqrt(A)=%g", B, A, Math.sqrt(A));
        L1 = s1.length();
        s3 = new String("23rd character in s1 is " + s1.charAt(22));
        System.out.println("Length of s1 = "+ L1);
        System.out.println(s1);
        for (i=1;i<=L1;i++) {
            s2 = String.format("%d",i%10);
            System.out.print(s2);
        }
        System.out.print("\n");
        System.out.println(s3);
    }
}
```

A simple program (BeerSong.java)

This application illustrates the. While loop

```java
class BeerSong {
    public static void main(String[] args) {
        int beerNum = 99;
        String sword = "bottle";
        String pword = "bottles";
        String uword;
        if (beerNum>1) {uword = pword;} else { uword = sword;}
        while (beerNum > 0) {
            System.out.println(beerNum + " " + uword + " of beer on the wall");
            System.out.println(beerNum + " " + uword + " of beer.");
            System.out.println("Take one down.");
            System.out.println("Pass it around.");
            beerNum = beerNum - 1;
            if (beerNum>1) {uword = pword;} else { uword = sword;}
            if (beerNum > 0) {
                System.out.println(beerNum + " " + uword + " of beer on the wall.");
                beerNum = beerNum - 1;
                System.out.println("No more " + pword + " of beer on the wall.");
            } else {
                System.out.println("\n");
            }
        }
    }
}
```

Input Args to main (BeerSongControl.java)

This application illustrates how input arguments to main arise, and are used.

```java
class BeerSongControl {
    public static void main(String[] args) {
        int beerNum;
        String sword, pword, uword;
        beerNum = Integer.valueOf(args[2]).intValue();
        sword = args[0];
        pword = args[1];
        if (beerNum>1) {uword = pword;} else { uword = sword;}
        while (beerNum > 0) {
            System.out.println(beerNum + " " + uword + " of beer on the wall");
            System.out.println(beerNum + " " + uword + " of beer.");
            System.out.println("Take one down.");
            System.out.println("Pass it around.");
            beerNum = beerNum - 1;
            if (beerNum>1) {uword = pword;} else { uword = sword;}
            if (beerNum > 0) {
                System.out.println(beerNum + " " + uword + " of beer on the wall.");
            } else {
                System.out.println("\n");
            }
        }
    }
}
```
Input Args to main (BeerSongControl.java)

beerNum = Integer.valueOf(args[2]).intValue();

This application illustrates how input arguments to main arise, and are interpreted as doubles.

public class FPAdder {
    public static void main(String[] args) {
        double num = 0;
        if (args.length <= 1) {
            System.out.println("You need to specify at least two numbers.");
        } else {
            while (i < args.length) {
                num += Double.valueOf(args[i]).doubleValue();
                i++;
            }
            System.out.println("The sum of the numbers is: "+num);
        }
    }
}

primitives (referenceTest.java)

Primitive data types are referred to by value.

public class referenceTest {
    public static void main(String[] args) {
        int A, B;
        A = 4;
        B = A; /* copy */
        System.out.println(String.format("Initial: A=%d, B=%d",A,B));
        System.out.println("Change B's value to 9");
        B = 9;
        System.out.println(String.format("Final: A=%d, B=%d",A,B));
    }
}

Arrays of primitives (referenceTest2.java)

Arrays (even of primitives) are objects, and are "by reference".

public class referenceTest2 {
    public static void main(String[] args) {
        int[] myInts, myInts2;
        myInts = new int[4];
        myInts2 = myInts;
        System.out.println(String.format("Init: mI[0]=%d, mI2[0]=%d",myInts[0],myInts2[0]));
        System.out.println("Change myInts2[0]'s value to 88");
        myInts2[0] = 88;
        System.out.println(String.format("Final: mI[0]=%d, mI2[0]=%d",myInts[0],myInts2[0]));
    }
}

primitives (referenceTest3.java)

Passing arguments to a method

public class referenceTest3 {
    public static void main(String[] args) {
        int A, B;
        A = 4;
        myIntArr = new int[1];
        myIntArr[0] = 8;
        myIntArr[0] = 8;
        referenceTest3.myTest(myIntArr,myInt);
        System.out.println(myIntArr[0]);
        System.out.println(myInt);
    }

    public static void myTest(int[] a, int b) {
        a[0] = 9;
        b = 9;
    }
}

primitives (LoanClass.java)

This illustrates a class definition and methods.

public class LoanClass {
    private double L;
    private double R;
    private double M;
    private int N;
    private String dprop;
    public LoanClass(String d) { L=1;R=0;M=1;N=1;dprop = d;}
    public double getPrinciple() { return L }
    public double[] getBalance() { }
    public void setPrinciple(double Lnew) {
        update();
    }
    private void update() {
        if (dprop.equals("Principle")) {
            L = getL(R,M,N); } }
    private double getL(double R, double M, int N) {
        /* formula */
    }
}

The instance variables. In Matlab, these would be the fields of the underlying struct object. Here, they are declared private. Only the methods have access to the fields of the struct. Only the methods have access to the fields of the struct.
The instance variables, in Matlab, these would be the fields of the underlying struct object.

Here, they are declared private. Only the methods have access to these variables. In Matlab, the "privacy" of automatic, only the methods have access to the fields of the struct.

The instance variables.

In Matlab, these would be the fields of the underlying struct object.

Here, they are declared private. Only the methods have access to these variables. In Matlab, the "privacy" of automatic, only the methods have access to the fields of the struct.

The instance variables.

In Matlab, these would be the fields of the underlying struct object.

Here, they are declared private. Only the methods have access to these variables. In Matlab, the "privacy" of automatic, only the methods have access to the fields of the struct.

The instance variables.

In Matlab, these would be the fields of the underlying struct object.

Here, they are declared private. Only the methods have access to these variables. In Matlab, the "privacy" of automatic, only the methods have access to the fields of the struct.

The instance variables.

In Matlab, these would be the fields of the underlying struct object.

Here, they are declared private. Only the methods have access to these variables. In Matlab, the "privacy" of automatic, only the methods have access to the fields of the struct.

The instance variables.

In Matlab, these would be the fields of the underlying struct object.

Here, they are declared private. Only the methods have access to these variables. In Matlab, the "privacy" of automatic, only the methods have access to the fields of the struct.

The instance variables.

In Matlab, these would be the fields of the underlying struct object.

Here, they are declared private. Only the methods have access to these variables. In Matlab, the "privacy" of automatic, only the methods have access to the fields of the struct.

The instance variables.

In Matlab, these would be the fields of the underlying struct object.

Here, they are declared private. Only the methods have access to these variables. In Matlab, the "privacy" of automatic, only the methods have access to the fields of the struct.
C vs Java, Matlab vs Java

No pointers
Static class members (367)
Primitives are "by-value"
Objects are by reference
object.Method(arg)
Constructors, many same constructors 342,
Instance variables are available in methods, (fields are accessible in methods)
ClassName.Method(args)
Keyword extends indicates inheritance
All extend Object
    toString method
    equals, finalize (GC)
    clone
Packaging (336)
Import package (337)