

## Java Basics

(lectures programs)

Sun OnLine Documentations

---

### Everything is an Object

➤ First Example: HelloDate Run Applet

```
public class HelloDate {  
    public static void main(String[] args) {  
        System.out.println ("Hello  
        CS476/CS576 students, it's: ");  
        System.out.println (new Date() );  
    }  
}
```

➔ To Compile:

```
% make  
OR  
% javac HelloDate.java
```

➔ To Run:

```
% java HelloDate
```

---

## Controlling Program Flow

- ✓ **Operators:** *precedence & assignment.*
- ✓ **Decision:** *if & switch.*
- ✓ **Looping:** *while & for.*

➤ **Example:** [VowelsAndConsonants](#) [Run Applet](#)

```
public class VowelsAndConsonants {
    public static void main (String[] args) {

        for (int i = 0; i < 100; i++) {
            char c = (char)(Math.random() * 26 + 'a');
            System.out.print(c + ": ");
            switch (c) {
                case 'a':
                case 'e':
                case 'i':
                case 'o':
                case 'u':

                    System.out.println("vowel"); break;
                case 'y':
                case 'w':

                    System.out.println( "Sometimes a
vowel"); break;
                default:

                    System.out.println("consonant");

            }}}
    }
```

---

## Initialization & Cleanup

➤ **Example 1: ArrayNew Run Applet**

```
public class ArrayNew {
    static Random rand = new Random();
    public static void main(String[] args) {
        int[] a;
        a = new int[rand.nextInt(20)];
        System.out.println("length of a = " +
            a.length);
        for (int i = 0; i < a.length; i++)
            System.out.println("a[" + i
                + "] = " + a[i]);
    }
}
```

➤ **Example 2: Garbage Run Applet**

```
class Chair {
    static boolean f = false;
    static int created = 0;
    int i;
    Chair() {
        i = ++created;
        if ( (i % 10000) == 0)
            System.out.print( ".");
    }
    public void finalize() {
        f = true;
    }
}

public class Garbage {
    public static void main(String[] args) {
        while (!Chair.f) {
            new Chair();
        }
        System.out.println (
            "\nTotal Chairs Created = " +
            Chair.created ");
    }
}
```

---

## The Java IO System

➤ **Example:** [IOStreamDemo.java](#)

```
public class IOStreamDemo {
    public static void main (String[] args)
        throws IOException {

        // Finding file length
        File file = new File(args[0]);
        long length = file.length();
        System.out.println("file length is: " + length);

        // Transfer bytes from in to out
        String infile = args[0];
        String outfile = args[1];
        InputStream fin = new FileInputStream(infile);
        OutputStream fout = new
FileOutputStream(outfile);

        byte[] buf = new byte[1024];
        int len;
        while ((len = fin.read (buf)) > 0) {
            fout.write (buf, 0, len);
        }
        fin.close();
        fout.close();

        // Reading standard input:
        BufferedReader stdin = new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter a line:");
        System.out.println(stdin.readLine());

        // Reading input by lines:
```

```

    BufferedReader in = new BufferedReader(new
FileReader(infile));
    String s, s2 = new String();
    while((s = in.readLine()) != null)
        s2 += s + "\n";
    in.close();

// File output
    try {
        BufferedReader in2 = new BufferedReader(new
StringReader(s2));
        PrintWriter out1 = new PrintWriter(new
BufferedWriter
                                (new FileWriter
("IODemo.out")));
        int lineCount = 1;
        while((s = in2.readLine()) != null )
            out1.println (lineCount++ + ":" + s);
        out1.close();
    } catch(EOFException e) {
        System.err.println("End of stream");
    }
}
}

```

**To execute:**

% java IOStreamDemo InputFile OutFile

and look at: IODemo.out and OutFile

**Collections of Objects**

➤ **Vector: CatsAndDogs** **Run applet**

```
class Cat {
    private int catNumber;
    Cat (int i) {
        catNumber = i;
    }
    void print() {
        System.out.println("Cat number " + catNumber);
    }
}

class Dog {
    private int dogNumber;
    Dog (int i) {
        dogNumber = i;
    }
    void print() {
        System.out.println("Dog number " + dogNumber);
    }
}

public class CatsAndDogs {

    public static void main (String[] args) {
        ArrayList cats = new ArrayList();

        for (int i = 0; i < 7; i++)
            cats.add (new Cat(i));
        // Not a problem to add a dog to cats:
        cats.add(new Dog(7));

        Iterator e = cats.iterator();
        while (e.hasNext())
            ((Cat)e.next()).print();
        // Dog is detected only at run-time
    }
}
```

➤ **Hashtable: Statistics** **Run applet**

```
class Counter {
    int i = 1;
```

```

    public String toString() {
        return Integer.toString(i);
    }
}

public class Statistics {
    public static void main (String[] args) {
        HashMap hm = new HashMap();

        for (int i = 0; i < 1000; i++) {
            // Produce a number between 0 and 9:
            Integer r =
                new Integer((int)(Math.random() * 10));
            if (hm.containsKey(r))
                ((Counter) hm.get(r)).i++;
            else
                hm.put (r, new Counter());
        }
        System.out.println (hm);

        for(int i = 0; i < 10; i++) {
            Integer r = new Integer((int)(i));
            if(hm.containsKey(r)){
                Counter c = (Counter)hm.get(r);
                System.out.println (r + " -> " + c + "\n");
            }
        }
    }
}

```

---

## Reusing Classes

Composition: Car Run applet

```

class Engine {
    public void start() {}
    public void rev() {}
    public void stop() {}
}

```

```

class Wheel {
    public void inflate(int psi) {
        System.out.println("inflated to:" + psi + "\n");
    }
}

class Window {
    public void rollup() {
        System.out.println("Rolled Up\n");
    }
    public void rolldown() {
        System.out.println("Rolled Down\n");
    }
}

class Door {
    public Window window = new Window();
    public void open() {}
    public void close() {}
}

public class Car {
    public Engine engine = new Engine();
    public Wheel[] wheel = new Wheel[4];
    public Door left = new Door(),
               right = new Door(); // 2-door
    public Car() {
        for (int i = 0; i < 4; i++)
            wheel[i] = new Wheel();
    }

    public static void main (String[] args) {
        Car car = new Car();
        car.left.window.rollup();
        car.wheel[0].inflate(72);
    }
}

```

---

**Polymorphism**

➤ **Example: Shapes2 Run applet**

```
class Shape {
    void draw() {
        System.out.println("Default.draw()");
    }
    void erase() {
        System.out.println ("Default.erase()");
    }
}

class Circle extends Shape {
    void draw() {
        System.out.println ("Circle calling super.draw():->
");
        super.draw();
    }
}

class Square extends Shape {
    void draw() {
        System.out.println("Square.draw()");
    }
    void erase() {
        System.out.println("Square.erase()");
    }
}

class Triangle extends Shape {
    void draw() {
        System.out.println("Triangle.draw()");
    }
    void erase() {
        System.out.println("Triangle.erase()");
    }
}

public class Shapes2 {

    public static Shape randShape() {
        switch((int)(Math.random() * 3)) {
            default:
            case 0: return new Circle();
```

```

        case 1: return new Square();
        case 2: return new Triangle();
    }
}

public static void main(String[] args) {
    Shape[] s = new Shape[9];
    // Fill up the array with shapes:
    for (int i = 0; i < s.length; i++)
        s[i] = randShape();
    // Make polymorphic method calls:
    for (int i = 0; i < s.length; i++){
        s[i].draw();
        s[i].erase();
    }
}
}

```

---

## Detecting Types

### ➤ Example 1: Shapes Run applet

```

class Shape {
    void draw() {
        System.out.println(this + ".draw()");
    }
}

class Circle extends Shape {
    public String toString() { return "Circle"; }
}

class Square extends Shape {
    public String toString() { return "Square"; }
}

class Triangle extends Shape {
    public String toString() { return "Triangle"; }
}

```

```

}

public class Shapes {
    public static void main (String[] args) {
        List s = new ArrayList();
        s.add (new Circle());
        s.add (new Square());
        s.add (new Triangle());

        Iterator e = s.iterator();

        while (e.hasNext())
            ( (Shape) e.next() ).draw();
    }
}

```

➤ **Example 2: CatsAndDogs2 Run applet**

```

class Pet {
    void print() {
    }
}

class Cat2 extends Pet{
    private int catNumber;
    Cat2 (int i) {
        catNumber = i;
    }
    void print() {
        System.out.println("Cat number " + catNumber);
    }
}

class Dog2 extends Pet{
    private int dogNumber;
    Dog2 (int i) {
        dogNumber = i;
    }
    void print() {
        System.out.println("Dog number " + dogNumber);
    }
}

public class CatsAndDogs2 {

```

```
public static void main (String[] args) {
    ArrayList cats = new ArrayList();

    for (int i = 0; i < 7; i++)
        cats.add (new Cat2 (i));
    // Not a problem to add a dog to cats:
    cats.add(new Dog2 (7));

    Iterator e = cats.iterator();
    while (e.hasNext())
        ((Pet)e.next()).print();
    // Cat and Dog are detected only at run-time
}
}
```

---