

Lecture 11

static variables

Two general types of variables might appear in a class, instance variables and static variables. The first type, instance variables are characterized by the fact that the values they assume are different for different instances of the class.

If a variable is declared to be static there is only one such variable and it belongs to all instantiations of the class. In fact, such variables exist before any instantiation of the class. Similarly, methods can be declared to be static. A static method belongs to all instantiations of the class, and exists before any instantiation of the class takes place.

We illustrate these points with the class Disk which, in each instance, will contain the artist and name of a CD disk:

```
public class Disk
{
    private String artist;
    private String title;
    private static int count;

    public Disk(String a, String t)
    {
        artist = a;
        title = t;
        ++count;
        System.out.print("artist = " + artist);
        System.out.println(", title = " + title);
    }

    public String getArtist()
    {
        return artist;
    }

    public String getTitle()
    {
        return title;
    }

    public static int getCount()
    {
        return count;
    }

    protected void finalize()
    {
        --count;
        System.out.print("Disk object finalizer:" + artist);
        System.out.println(title);
    }
}

} // end class Disk
```

There are two instance variables and one static variable in the class:

```
private String artist;
private String title;
private static int count;
```

The purpose of the `static` variable `count` is to keep a record of how many instances of the class `Disk` exist at any time in the program that follows.

The constructor

```
public Disk(String a, String t)
{
    artist = a;
    title = t;
    ++count;
    System.out.print("artist = " + artist);
    System.out.println(", title = " + title);
}
```

makes a record of the artist and title on each disk and increments the static variable `count` when an instance of `Disk()` is created.

Note that we also have a static method

```
public static int getCount()
{
    return count;
}
```

The method

```
protected void finalize()
{
    --count;
    System.out.print("Disk object finalizer:" + artist);
    System.out.println(title);
}
```

will be used to decrement the variable `count` when we eliminate a disk from our program. The word `protected` in the title is another access modifier, such as `public` or `private`. It will be discussed later.

The method `finalize()` will work in conjunction with the "garbage collector". See the comments after the last version of `DiskTest`.

The simplest version of `DiskTest` is

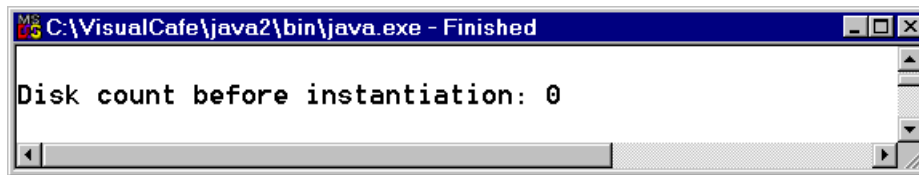
```
public class DiskTest
{
    public static void main(String args[])
    {
        String output = "\nDisk count before instantiation: ";
        System.out.println(output+ Disk.getCount() + "\n" );

        }// main()
} // end class DiskTest
```

As for file details: The text for `Disk.java` and `DiskTest.java` are in separate files; both files are in the same directory.

The point to notice here is that we are calling on a method of the class (in `Disk.getCount()`) without instantiating the class `Disk`.

When this version of `DiskTest` is run the output is:



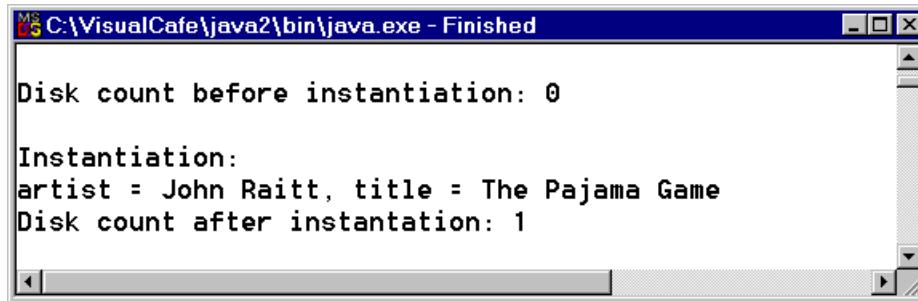
In sum, the class `Disk` has been compiled but it not been instantiated. A call to its static method `getCount()` yields the default value 0 of the static variable `count`.

To continue, we add one disk:

```
public class DiskTest
{
    public static void main(String args[])
    {
        String output = "\nDisk count before instantiation: ";
        System.out.println(output+ Disk.getCount()+"\n" );

        System.out.println("Instantiation:");
        Disk d1 = new Disk("John Raitt", "The Pajama Game");
        output = "Disk count after instantiation: ";
        System.out.println(output+ Disk.getCount() + "\n");
    }
}
```

The output reflects the addition:



```
C:\VisualCafe\java2\bin\java.exe - Finished
Disk count before instantiation: 0
Instantiation:
artist = John Raitt, title = The Pajama Game
Disk count after instantiation: 1
```

Continuing, we add two more disks and then eliminate the first by setting `d1 = null` in (//1):

```
public class DiskTest
{
    public static void main(String args[])
    {
        String output = "\nDisk count before instantiation: ";
        System.out.println(output+ Disk.getCount()+"\n" );

        System.out.println("Instantiation:");
        Disk d1 = new Disk("John Raitt", "The Pajama Game");
        output = "Disk count after instantiation: ";
        System.out.println(output+ Disk.getCount() +"\n");

        System.out.println("Instantiation:");
        Disk d2 = new Disk("Lola Beltran", "12 Exitos Rancheros");
        output = "Disk count after second instantiation: ";
        System.out.println(output+ Disk.getCount() +"\n");

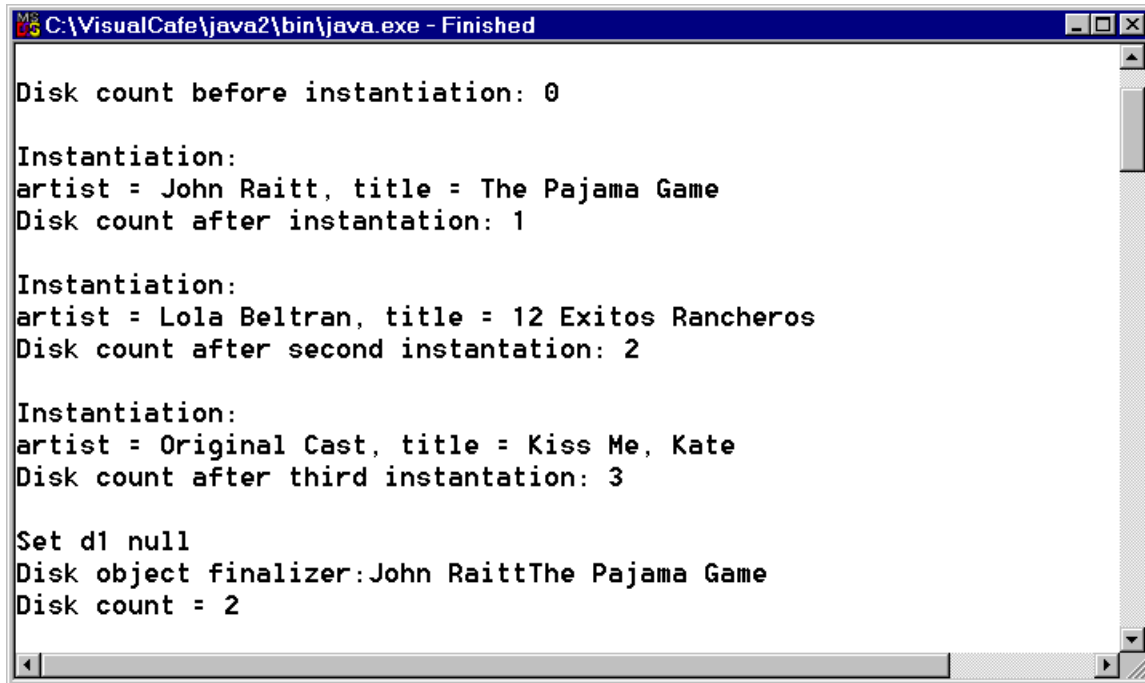
        System.out.println("Instantiation:");
        Disk d3 = new Disk("Original Cast", "Kiss Me, Kate");
        output = "Disk count after third instantiation: ";
        System.out.println(output+ Disk.getCount() +"\n");

        System.out.println("Set d1 null");
        d1 = null; //1
        System.gc(); //2
        System.out.println("Disk count = " +Disk.getCount()+"\n");
    }
}
```

The call `System.gc()` in (//2) is a call to the "garbage collector". The instruction `d1 = null` freed the resources that were used for `d1`. Eventually, the garbage collection would operate perform the necessary housekeeping chores. However, it is not possible to tell when it would operate, so the call `System.gc()` insures that the collection is done "now".

The method `finalize()` of the class `Disk` comes into play at this point. When the garbage collector runs it makes a call on `finalize()`. (Java provides an empty, default `finalize()` for programs without one) Checking the code for our `finalize()` we see that it decrements the count variable.

The output shows that the count is increased by one each time a CD is added, and is decreased by one after the disk d1 is set to null:



```
C:\VisualCafe\java2\bin\java.exe - Finished

Disk count before instantiation: 0

Instantiation:
artist = John Raitt, title = The Pajama Game
Disk count after instantiation: 1

Instantiation:
artist = Lola Beltran, title = 12 Exitos Rancheros
Disk count after second instantiation: 2

Instantiation:
artist = Original Cast, title = Kiss Me, Kate
Disk count after third instantiation: 3

Set d1 null
Disk object finalizer: John RaittThe Pajama Game
Disk count = 2
```

To summarize: only one copy of the static variable count exists at any time; its' value is the same for all instances of the class.

Initialization Blocks

We continue our study of static variables. We begin with a static array and initialize the array values in a static block. Note that the values assigned to the array elements are random integers between 0 and 99.

A point of syntax must be mentioned: the variables that are assigned values in a static block (//2) must be static variables (//1).

```
public class TestInitialize
{
    static int[] values = new int[10];    //1

    static
    {
        for (int i = 0; i < values.length; i++)
            values[i] = (int)(100*Math.random());    //2
    }

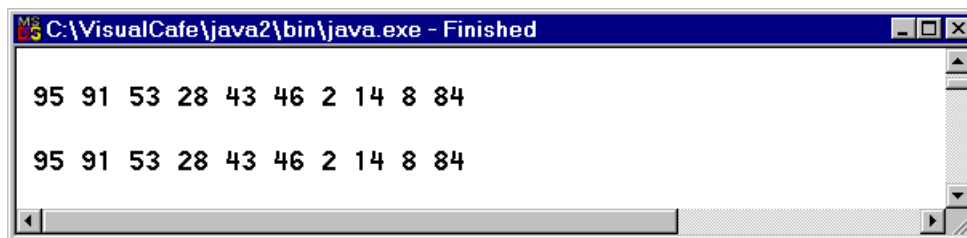
    void listValues()
    {
        for (int i = 0; i < values.length; i++)
            System.out.print(" " + values[i]);
            System.out.println();
    }

    public static void main(String args[])
    {
        System.out.println();
        TestInitialize example1 = new TestInitialize();
        example1.listValues();

        System.out.println();
        TestInitialize example2 = new TestInitialize();
        example2.listValues();
    }
}

} // end class TestInitialize
```

We create two instances of the class `TestInitialize` and print out the initial values of the array. They are the same in both instances:

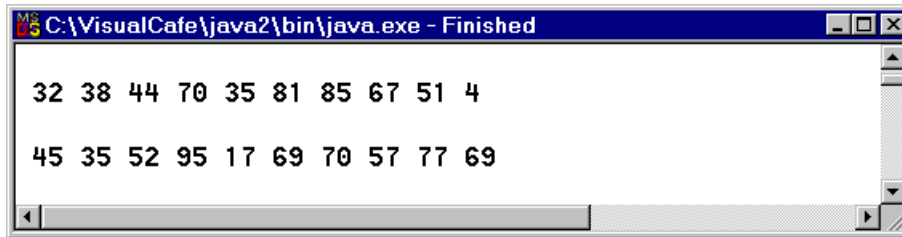


```
C:\VisualCafe\java2\bin\java.exe - Finished
95 91 53 28 43 46 2 14 8 84
95 91 53 28 43 46 2 14 8 84
```

Thus, the static block has been run only once.

If we modify the code in the program by removing the word `static`

from the block in (//2), leave everything else as before, and run the program a we find that the initialization block was run twice:



```
C:\VisualCafe\java2\bin\java.exe - Finished
32 38 44 70 35 81 85 67 51 4
45 35 52 95 17 69 70 57 77 69
```

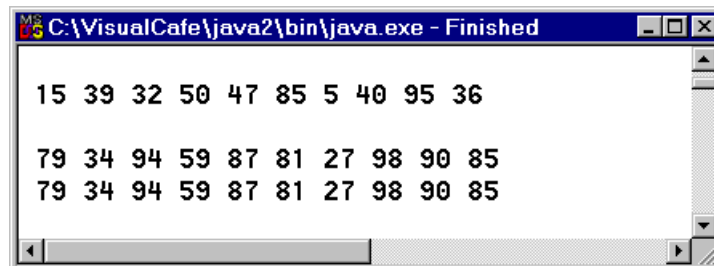
This, however is not the total story because the array in (//1) is a static array. To investigate further, add the line in blue below to the `main()` method. That is print out the values of `example1` after `example2` has been instantiated.

```
public static void main(String args[])
{
    System.out.println();
    TestInitialize example1 = new TestInitialize();
    example1.listValues();

    System.out.println();
    TestInitialize example2 = new TestInitialize();
    example2.listValues();

    example1.listValues();
}
```

When the program is run again the result is:



```
C:\VisualCafe\java2\bin\java.exe - Finished
15 39 32 50 47 85 5 40 95 36
79 34 94 59 87 81 27 98 90 85
79 34 94 59 87 81 27 98 90 85
```

In short, when `example1` is instantiated the values created are 15 39 .. 95 36. When `example2` is instantiated the values created are 79 34 ..90 85. However, since the array values[] is static only one copy of its' values exist. The values of `example1` have been overwritten and replaced by those of `example2`.

Example: Working with private variables in a Class

In this example the Widget manufacturing company has twelve salesman. The company wishes to build a record of the daily sales of each salesman for five days, monday thru friday, and have the computer compute each salesman's weekly earnings. The record for each will be kept in a class called `Salesman`:

```

import java.text.*;

public class Salesman
{
    private String name;
    private double[] DailySales = new double[5];
    private static double[] DailyMax = new double[5];
    private double earnings;

    DecimalFormat precision2 = new DecimalFormat("0.00");

    public Salesman(String Name)
    { name = Name; }

    public String getName()
    { return name;}

    public void setDailySales(int i, double Sales)
    { DailySales[i] = Sales;}

    public double getDailySales(int i)
    { return DailySales[i];}

    public void setDailyMax(int i, double M)
    { DailyMax[i] = M;}

    public double getDailyMax(int i)
    { return DailyMax[i]; }

    public void setEarnings()
    {
        double wages = 0;
        for (int i = 0 ; i < 5; i++)
            wages = wages + 0.08*DailySales[i];
        earnings = wages;
    }

    public double getEarnings()
    {
        return earnings;
    }

    public String toString()
    {
        String out = "";
        out += " name:  " + name + "\n";

        for (int k = 0; k < 5; k++)
            out += " " +precision2.format(DailySales[k])+ " ";
        out += "earned " + precision2.format(earnings);
        return out;
    }
}
} // end class Salesman

```

The records for the salesman will be processed by a **HomeOffice** application:

```
import javax.swing.*;
import java.text.*;

public class HomeOffice
{
    public static void main(String args[])
    {
        String Staff[] = {"Abel", "Gauss", "Galois",
                        "Newton", "L'Hospital", "DeMorgan",
                        "Linnik", "Klein", "Murphy",
                        "Dimaggio", "Williams", "Banks"};

        int N = Staff.length;

        Salesman[] sm = new Salesman[N];

        for (int i = 0; i < N; i++)
            sm[i] = new Salesman(Staff[i]);

        for (int k = 0; k < 5; k++)    // k <-> to the days
        {
            double sales= 0;
            double Max = 0;
            for (int j = 0; j < N; j++) // j <-> salesmen
            {
                sales =2000 +400*Math.random();
                if (sales > Max) Max = sales;
                sm[j].setDailySales(k, sales);
            }
            sm[k].setDailyMax(k, Max);
        } // end k loop

        for (int i = 0; i < N; i++)
            sm[i].setEarnings();

        System.out.print("\n Weekly Summary:");
        String summary = " daily sales, weekly earnings\n";
        System.out.println(summary);

        for (int i = 0; i < N; i++)
            System.out.println( sm[i].toString() );

        DecimalFormat precision = new DecimalFormat("0.00");
        String dmax = " ";

        for (int k = 0; k < 5; k++)
        {
            dmax +=
                precision.format(sm[0].getDailyMax(k))+ "  ";
        }

        System.out.println("\n daily maximums");
    }
}
```

```

        System.out.println(dmax);

        System.exit(0);

    }// end main(String args[])
}// end class HomeOffice

```

Command Prompt

```

Weekly Summary: daily sales, weekly earnings

name: Abel
2307.98  2202.95  2127.59  2365.43  2267.13  earned 901.69
name: Gauss
2166.48  2230.07  2277.75  2069.24  2133.34  earned 870.15
name: Galois
2382.30  2271.73  2036.22  2305.39  2069.99  earned 885.25
name: Newton
2075.65  2363.43  2113.74  2310.09  2260.95  earned 889.91
name: L'Hospital
2091.19  2028.17  2309.13  2203.35  2037.84  earned 853.57
name: DeMorgan
2137.61  2389.29  2304.53  2235.88  2360.79  earned 914.25
name: Linnik
2163.03  2140.07  2034.46  2058.15  2392.32  earned 863.04
name: Klein
2052.03  2255.49  2146.79  2392.45  2387.15  earned 898.71
name: Murphy
2386.99  2389.45  2362.73  2192.18  2253.09  earned 926.76
name: Dimaggio
2385.62  2341.71  2117.75  2135.01  2067.81  earned 883.83
name: Williams
2295.13  2171.15  2271.46  2331.05  2317.92  earned 910.94
name: Banks
2354.69  2135.69  2148.31  2108.46  2123.71  earned 869.67

daily maximums
2386.99  2389.45  2362.73  2392.45  2392.32

```

Lab 5. A professor has decided to build a java application keep his records for one of his classes. Your assignment is to complete the outline of the project. Your application is to output the one given at the end.

There will be two java classes to carry out the job. The first will be a record for an individual student. Its' structure will reflect the fact that he will give three exams, two hour exams and a three hour final. At the time he sets this up he does not know what the maximum score will be on each class, so he includes a static array to hold these maximum. Finally, he wishes to have the computer calculate the weighted average for the course, so he will define a method for that in the class.

```
import java.text.*;

public class Student
{
    private String name;
    private double score[] = new double[3];
    private static double maximum[] = new double[3];
    private double WeightedAverage;

    DecimalFormat precision2 = new DecimalFormat("0.00");

    public Student(String Name)
    {
        name = Name;
    }

    public String getName()
    {
    }

    public void setScore(int ExamNumber, double Score)
    {
        score[ExamNumber] = Score;
    }

    public double getScore(int ExamNumber)
    {
    }

    public void setMaximum(int ExamNumber, double Maximum)
    {
    }

    public double getMax(int ExamNumber)
    {
    }

    public void setWeightedAverage()
    {
        // the exams will be wighted as follows:
        // 25% for each hour exam; 50% for the final
    }
}
```

```

    public double getWeightedAverage()
    {
    }

    public String toString()
    {
    }

} // end class Student

```

```
import javax.swing.*;
```

The Prof then builds an application called Prof to interface with the student records. He was lucky to have but 12 students in his class, so he can type in their names as below.

Next, he wants to make sure his application will run so he types in some score for the first two hour exams and the final. For each exam, he enters 13 numbers. The last of the thirteen is the maximum score for the exam. Thus, the first exam has a max of 120, the second a max of 100, and the final a max of 110. As for the first 12 numbers, the first belongs to the first student, the second to the second, etc.

Finally, he has become bored with output to the DOS window, so he goes to his references and finds the code printed in blue near the end of the main() method that enables him to construct a message window with scrollbars

```

public class Prof
{

    public static void main(String args[])
    {
        String Name[] = {"aveazul", "averoja", "bigbird",
                        "bigfoot", "birdseye", "canary",
                        "duckweb", "everywing", "fasthawk",
                        "greenbill", "hornyowl", "jaybird"};

        int M = Name.length;

        double exam1[] = {90,80,70, 65,94,42,
                        99,72,89, 67,44,81, 120};

        double exam2[] = {45,84,91, 75,64,92,
                        94,87,78, 72,54,80, 100};

        double exam3[] = {92,76,81, 80,74,82,
                        84,80,70, 81,65,75, 110};

        // your job is to get this data into the individual
        // student records, and then be able to access it
    }
}

```

```

String summary = "";
for (int i = 0; i < M; i++)
    summary += " i = " + i + "    "+s[i].toString();

JTextArea  outputArea = new JTextArea(30, 10);
JScrollPane scroller = new JScrollPane(outputArea);
outputArea.setText(summary);
JOptionPane.showMessageDialog(null,scroller);

    System.exit(0);
} // end main(String args[])

} // end class Prof

```

The output is:

