Lecture 7

Picking Balls From an Urn

The problem: An urn has n (n = 10) balls numbered from 0 to 9. A ball is selected at random, its' is number noted, it is set aside, and another ball is selected from the remaining 9 balls. The process continues until all 10 balls have been selected from the urn.

You are to simulate the process by making a record of the balls chosen, in the sequence in which they are chosen. Typical output for a run would be:

```
Command Prompt

2:\11Pic20\Lecture_7>java Test

There were 10 balls in the urn.
10 were chosen. The order in which they appeared was:

    7 2 0 4 3 1 5 8
    9 6

```

That is, the first ball chosen the one numbered 0, the next was numbered 3, ... , the last was the one numbered 4.

More generally, the urn is to have n balls, and k are to be selected. The program in this case would record the k balls that were chosen:

```
Command Prompt

There were 20 balls in the urn.
5 were chosen. The order in which they appeared was:

    6 9 19 10 3

```

The selection process will be simulated by a class called:

```
public class PickFromUrn
{
    int[] b;
    int n;

    public PickFromUrn(int N)
    {
        n = N;
        int[] a = new int[n];
        b = new int[n]; // 1
    }

    // Simulate the selection process
}
```
for (int i = 0; i < n; i++) a[i] = i; // 2

for (int k = 0; k < n; k++)
{
    int m = (int)((n-k)*Math.random());
    b[k] = a[m];

    for (int j = m; j < n-k-1; j++)
        a[j] = a[j+1];
}
} // end PickFromUrn()

} // end class PickFromUrn

In this class the integer array int[] b; will be the array that holds the selections.

The code marked in blue is the "constructor" of this class. The name of the constructor PickFromUrn() is the name of the class PickFromUrn, along with the parameter list (int N). The parameter list is empty in this case.

In the constructor we begin (at //1) by instantiating two integer arrays with

    n = N;
    int[] a = new int[n];
    b = new int[n];

Then (at //2) we initialize the array a[] by setting a[i] = i:

    for (int i = 0; i < n; i++) a[i] = i;

We are then in a position to simulate the selection of balls from the urn. The general idea (with n = 8) is this;

| a[4] | b[0] = a[m]                 | a[m] = a[m+1] | a[5] |
| a[1] |                             | .           | a[2] |
| a[0] |                             | .           | a[1] |
|       |                             | a[0] = a[0] | a[0] |

The first column represents the initialized array, with a[i] = i for all i.

In the second column we choose a random integer between 0 and 7, and then define the first element of the b array by b[0] = a[m].

In the third column we leave the array elements a[0] thorough a[m-1] alone, and shift down the elements a[m+1] through a[7].

The fourth column represents the state of the arrays at the end of the first loop: b[0] has been defined; the original array a[] has been redefined to reflect the selection of the first ball.
Continuing, we repeat with the modified a[] array to define b[1]:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a[5]</td>
<td></td>
<td>a[m+1] = a[m+2]</td>
<td>b[1]</td>
</tr>
<tr>
<td>a[1]</td>
<td></td>
<td></td>
<td>a[2]</td>
</tr>
<tr>
<td>a[0]</td>
<td></td>
<td></td>
<td>a[1]</td>
</tr>
</tbody>
</table>

The process proceeds inductively. The code for this is:

```java
for (int k = 0; k < n; k++)
{
    // choose b[k]:
    int m = (int)((n-k)*Math.random());
    b[k] = a[m];

    // shift down:
    for (int j = m; j < n-k-1; j++)
        a[j] = a[j+1];
}
```

Next we add a print method `PrintOut(int L)` to the class `PickFromUrn`:

```java
public class PickFromUrn
{
    int[] b;
    int n;

    public PickFromUrn(int N)
    {
        n = N;
        int[] a = new int[n];
        b = new int[n];

        for (int i = 0; i < n; i++)
            a[i] = i;

        for (int k = 0; k < n; k++)
        {
            int m = (int)((n-k)*Math.random());
            b[k] = a[m];

            for (int j = m; j < n-k-1; j++)
                a[j] = a[j+1];
        }
    }

    public void PrintOut(int L)
    {
        String blank = " ";
        int m = Math.min(n,L);
```
System.out.print("\n There were " +n);
System.out.println(" balls in the urn.");
System.out.println(" + m + " were chosen. The order in ");
System.out.println(" which they appeared was:\n");
System.out.print(blank);

for (int i = 0; i < m; i++)
{
    System.out.print(b[i]+" ");
    if (i%8 == 7)
    {
        System.out.println();
        System.out.print(blank);
    }
}
System.out.println("\n");
} // end PrintOut(int L)

} // end class PickFromUrn

All the code we have written so far is in one class, the class
that can be outlined as

    public class PickFromUrn
    {
        int[] b;
        int n;

        public PickFromUrn(int N)
        {
        } // end PickFromUrn(int N)

        public void PrintOut(int L)
        {
        } // end PrintOut(int L)

    } // end class PickFromUrn

It is a class with one constructor PickFromUrn(int N), and one
method public void PrintOut(int L). The class variables int[] b and act
int n, as we shall see, as "global" variables. Note that there is no
"main()" method.

To run the this class we write an application Test. The file will
be written in a separate file ; the file will have to be kept in the
same directory as the PickFromUrn.java file.

The application Test an appealing simplicity: In (//1) we create
two instances of the class. One urn contains 10 balls, the other. Then,
in (//2) we print out all the balls from the first urn and the first 5
selected from the second:

public class Test
{ public static void main(String args[]) {
    PickFromUrn p = new PickFromUrn(10);
    PickFromUrn q = new PickFromUrn(20); // 1
    p.PrintOut(p.n);
    q.PrintOut(5); // 2
}
} // end main(String args[])
}// end class Test

As for terminology, in (//1), p and q are "references" to the class PickFromUrn. Part (//2) shows how to utilize the PrintOut() method of the different instances of the class.

The output of Test is:

```
Z:\11Pic20\Lecture_7>java Test

There were 10 balls in the urn. 10 were chosen. The order in which they appeared was:

    4 7 6 1 2 5 3 9
    8 0

There were 20 balls in the urn. 5 were chosen. The order in which they appeared was:

    4 16 1 13 0
```

Z:\11Pic20\Lecture_7>
Overloading Constructors

A class can have several constructors. The name of the constructor is the same as the class in each case; the parameter list is different in each case.

To illustrate, we shall add a constructor that has an array as a parameter. The motivation is that up to this point we have been working with the balls marked from 1 to \( n \). If you go back and examine the lines where the consecutive balls are marked

```java
for (int k = 0; k < n; k++)
{
    int m = (int)((n-k)*Math.random());
    b[k] = a[m];
    for (int j = m; j < n-k-1; j++)
        a[j] = a[j+1];
}
```

you see that we only work with the integers \( j \) in \( a[j] \); we do nothing to the value of the array component \( a[j] \).

Thus if we were to pass a array of distinct elements to the class \texttt{PickFromUrn} we could use the algorithm of the previous paragraph to make random selections from the array. So, we add a constructor that handles an array:

```java
public class PickFromUrn
{
    int[] b;
    int n, l;

    public PickFromUrn(int N)
    {
        n = N;
        int[] a = new int[n];
        b = new int[n];
        for (int i = 0; i < n; i++)
            a[i] = i;
        for (int k = 0; k<n; k++)
        {
            int m = (int)((n-k)*Math.random());
            b[k] = a[m];
            for (int j = m; j < n-k-1; j++)
                a[j] = a[j+1];
        }
    }
}
```

```java
public PickFromUrn(int[] A )
{
```

```java```
n = A.length;

int[] a = new int[n];
b = new int[n];

for (int i = 0; i < n; i++)
    a[i] = A[i];

for (int k = 0; k < n; k++)
{
    int m = (int)((n-k)*Math.random());
b[k] = a[m];
    for (int j = m; j < n-k-1; j++)
        a[j] = a[j+1];
}

} // end PickFromUrn(int[] A )

} // end class PickFromUrn

To test this out type in an array of distinct elements in
the method main(), and then instantiate:

public class Test
{
    public static void main(String args[])
    {
        PickFromUrn p = new PickFromUrn(8);
PickFromUrn q = new PickFromUrn(10);

int[] pr =new int[7];
pr[0] =2; pr[1] = 3; pr[2] = 5; pr[3] = 7;

PickFromUrn r = new PickFromUrn(pr);
p.PrintOut(p.n);
q.PrintOut(q.n);
r.PrintOut(r.n);

} // end main(String args[])

} // end class Test

The output is:
Arrays of Classes

We illustrate how to work with an array of classes.

public class Test
{
    public static void main(String args[])
    {
        PickFromUrn[] p;

        p = new PickFromUrn[3]; // array of 3
        int A[] = {5, 7, 8, 18, 20, 30};
        p[0] = new PickFromUrn(A);
        p[1] = new PickFromUrn(20);
        p[2] = new PickFromUrn(32);

        p[0].PrintOut(p[0].n);
        p[1].PrintOut(12);
        p[2].PrintOut(32);

    }// end main(String args[])
The output is:

![Command Prompt output]

There were 6 balls in the urn. 6 were chosen. The order in which they appeared was:

7 8 5 30 20 18

There were 20 balls in the urn. 12 were chosen. The order in which they appeared was:

17 15 4 2 19 13 8 14
11 9 1 3

There were 32 balls in the urn. 32 were chosen. The order in which they appeared was:

12 17 23 7 15 26 30 11
6 29 31 9 5 4 22 13
10 3 28 14 0 16 21 8
18 25 1 27 29 2 19 24
Lab 4. You are to write an application which will take a message and permute all its words.

The original message must contain at least 14 words. It can be anything you like; type all the letters in the lower case.

The general structure of your program should be one file. An outline would be:

```java
public class Message
{
    public static void main(String args[])
    {
    } // end main()
}
} // end class Message

class Permute
{
    String[] b;
    int n;

    public Permute(String[] word)
    {
    } // Permute(String[] word)

    public void PrintOut(int L)
    {
    } // end PrintOut(int L)
}

} // end class Permute

In short, the application will be in one file named Message.java. The file will contain two classes public class Message and class Permute. The class will contain a constructor Permute(String[] word) and a print routine PrintOut(int L).

In a typical run the application will print out the original message, a permuted version of all the words in the message, and another shorter version.
Specifications:

1) The message can be anything you like. It must have at least 14 words, which should be typed in lower case letters.

2) The various messages must be offset with a margin of 12 blanks, as above.

3) The shortened 8 word permutation must come in the third message must be different from the first 8 words in the second message.

Note: You will have to construct the array of words defining the message in the "main()" part of the application. In addition, it will be easier to split the printouts into two parts: one in the "main()" part to print out the original message, and one as a method PrintOut(int L) in the Permute class for the remaining messages.