Recursion
Recursion

There are two kinds of repetitive technique:

- Iterative
- Recursive

Iteration

- Iteration use loops (for, while, or do).
- Iteration often provide a straightforward and efficient way to implement a repetitive process.
- Iterative solution is complex. Discovering or verifying such solution is not simple task. In these cases recursion is an elegant alternative.

By: S. Hassan Adelyar
Recursion

- Recursion
  - Recursion is an **important** problem solving **strategy**.
  - Simple solution to difficult problems.
  - **Programming technique** in which a **method call itself**.
  - In each call the argument become **smaller** so the problem become **simpler**.
  - **Recursive method** calls itself. When it calls itself, it does so to solve a **smaller problem**.

By: S. Hassan Adelyar
Decimal To Binary

Recursion

```java
public static void main(String[] args){
    performop(b, a);
}

public static void performop(int n, int s) {
    if (n > 0 ) {
        performop (n/s, s);
        System.out.print(n % s);
    }
}
```

By: S. Hassan Adelyar
Factorial

Recursion

\[ N! = n*(n-1)*(n-2)*\ldots*3*2*1 \]

Public static int recursivefactorial(int n)
{
    if (n == 0) return 1;
    else return n*recursivefactorial(n-1);
}
Recursion

class factofN {
    static int a;
    static int b = 7;
    public static void main(String [] args) {
        a = recursivefactorial(b);
        System.out.println(a);
    }

    public static int recursivefactorial(int n) {
        if (n == 0) return 1;
        else return n * recursivefactorial(n - 1);
    }
}

By: S. Hassan Adelyar
Counting the sum of 1+2+3………..+n:

Recursion

class sumofnum {
    static int a;
    static int b = 10;
    public static void main(String[] args) {
        a = sumof(b);
        System.out.println(a);
    }
    public static int sumof(int n) {
        int sum;
        if (n == 1)
            sum = 1;
        else
            sum = sumof(n-1) + n;
        return sum;
    }
}

By: S. Hassan Adelyar
Write each digit vertically

Recursion

class vertnum {
static int n = 3652;
public static void main(String[] args) {
  writevertical(n);
}
public static void writevertical(int n) {
  if (n < 10) System.out.println(n);
  else {
    writevertical(n/10);
    System.out.println(n % 10);
  }
}
}
Fibonacci Sequence

Recursion

The fibonacci sequence is: 1,1,2,3,5,8,13,21 ..... 

Long fib( int n) {
{
If (n<2) return n;
Return fib(n-1) + fib(n-2);
}

By: S. Hassan Adelyar
public class FibonacciNumber {
    public static void main (String [] args) {
        int firstFibNum = 3;
        int secondFibNum = 5;
        int nth = 6;
        System.out.println("The Fibonacci number at position" + nth + "is:");
        System.out.println(" + rFibNum(firstFibNum, secondFibNum, nth));
    }
}
Recursion

```java
public static int rFibNum(int a, int b, int n){
    if(n == 1)
        return a;
    else if(n==2)
        return b;
    else
        return rFibNum(a, b, n-1) + rFibNum(a, b, n-2);
}
```
Recursion

import java.io.*;
public class FibonacciNumber {
    static BufferedReader keyboard = new BufferedReader (new InputStreamReader (System.in));
    public static void main (String [] args) throws IOException {
        int firstFibNum;
        int secondFibNum;
        int nth;
        System.out.print("Enter the first Fibonacci Number: ");
        firstFibNum = Integer.parseInt(keyboard.readLine());
        System.out.println();
        System.out.print("Enter the second Fibonacci Number: ");
        secondFibNum = Integer.parseInt(keyboard.readLine());
        System.out.println();
        System.out.print("Enter the third Fibonacci Number: ");
        nth = Integer.parseInt(keyboard.readLine());
        System.out.println();
        By: S. Hassan Adelyar
Recursion

```
System.out.print("Enter the position of the desired" + "Fibonacci number: ");
nth = Integer.parseInt(keyboard.readLine());
System.out.println();
System.out.println("The Fibonacci number at position" + nth + "is: "+ rFibNum(firstFibNum, secondFibNum, nth));
```
Recursion

public static int rFibNum(int a, int b, int n){
    if(n == 1)
        return a;
    else if(n==2)
        return b;
    else
        return rFibNum(a, b, n-1) + rFibNum(a, b, n-2);
}

By: S. Hassan Adelyar
Fractals

Recursion

import java.applet.Applet;
import java.awt. *

public class Fractal extends Applet {
    private Image display;
    private Graphics drawingArea;

By: S. Hassan Adelyar
Recursion

public void init() {
    int height = getSize().height;
    int width = getSize().width;
    display = createImage(width, height);
    drawingArea = display.getGraphics();
    randomFractal(0, height/2, width, height/2, drawingArea);
}

public void paint(Graphics g) {
    g.drawImage(display, 0, 0, null);
}
Recursion

public static void randomFractal
    (
        int leftX,
        int leftY,
        int rightX,
        int rightY,
        Graphics drawingArea
    )

By: S. Hassan Adelyar
Recursion

```java
{  
    final int STOP = 4;
    int midX, midY;
    int delta;
    if((rightX - leftX) <= STOP)
        drawingArea.drawLine(leftX, leftY, rightX, rightY);
    else {
        midX = (leftX + rightX) / 2;
        midY = (leftY + rightY) / 2;
        delta = (int)((Math.random() - 0.5) * (rightX - leftX));
        midY += delta;
        randomFractal(leftX, leftY, midX, midY, drawingArea);
        randomFractal(midX, midY, rightX, rightY, drawingArea);
    }
}
```

By: S. Hassan Adelyar
The Towers of Hanoi

Recursion

- An ancient puzzle. There are 3 towers and there are 7 disks on tower A. The disks have different diameter.
- The goal is to transfer all the disks from tower A to tower C according to the following rules:
  - Carry only one disk at a time.
  - Never put a big disk on the small disk.
  - You can use tower B temporarily.
- If the sub-tree has an odd number of disks, start by moving the topmost disk directly to the tower where you want the sub-tree to go.

By: S. Hassan Adelyar
Recursion

class towerapp {

    static int ndisks = 3;
    public static void main (String[] args) {
        dotowers(ndisks, 'A', 'B', 'C');
    }

    public static void dotowers(int topn, char from, char inter, char to) {
        if(topn == 1)
            System.out.println("Disk 1 from " + from + " to " + to);
        else {
            dotowers(topn-1, from, to, inter);
            System.out.println("Disk " + topn + "from " + from + "to " + to);
            dotowers(topn-1, inter, from, to);
        }
    }
}

By: S. Hassan Adelyar
Recursion

When tracing the code, the stack has the following status:

<table>
<thead>
<tr>
<th>N</th>
<th>From</th>
<th>Inter</th>
<th>To</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>2, 4</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>C</td>
<td>B</td>
<td>2, 4</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>2, 4</td>
</tr>
</tbody>
</table>
Traversing a Maze

Recursion

- Suppose the following two facts about a maze:
- Somewhere is a panel which contain the secret of the universe.
- You can keep this panel if you can enter the maze, find the panel, and return to the maze’s entrance.

- The maze is built on a rectangular grid. At each point of the grid, there are four directions to move: north, east, south, or west. Some directions may be blocked by an impenetrable wall.
- You accept to enter the maze but only with the help of your portable computer and a method that we will write to guide you into the maze and back out.

By: S. Hassan Adelyar