Hari Sahasrabuddhe Lecture Series on Inflections in Computing

Department of Computer Science & Engineering

IIT Kanpur

Talks:

Jan 20 @ Infosys Bangalore,

Jan 29 @ IITK

Title: The Power of Abstraction

Abstract: Abstraction is at the center of much work in Computer Science. It encompasses finding the right interface for a system as well as finding an effective design for a system implementation. Furthermore, abstraction is the basis for program construction, allowing programs to be built in a modular fashion. This talk will discuss how the abstraction mechanisms we use today came to be, how they are supported in programming languages, and some possible areas for future research.

Professor Barbara Liskov is the Ford Professor of Engineering in the MIT School of Engineering's Electrical Engineering and Computer Science department and an Institute Professor at the Massachusetts Institute of Technology. She leads the Programming Methodology Group at MIT, with a current research focus in Byzantine fault tolerance and distributed computing.

The contributions of Professor Liskov include the design and implementation of **CLU**, a programming language that significantly influenced the development of object-oriented programming; **Argus**, the first high-level language to support implementation of distributed programs; and **Thor**, an object-oriented database system. With Jeannette Wing, she developed a particular definition of subtyping, commonly known as the Liskov substitution principle.

```
//package
Crypto;

import
java.io.*;

public class
GF256
{
public static
final int
FieldSize = 256;
// size of the
field GF[256]
public static
final int
mulGroupSize =
255; // size of
multiplicative
group GF^*[256]
public static
final int
noPrimePowers =
128; // size of
multiplicative
group Z^*_{255}
public static
final byte ZERO
= 0; // element
0
public static
final byte ONE =
1; // element 1
public static
final byte
GENERATOR = 3;
// generator of
the
multiplicative
group

/*
Multiplication
and
exponentiation
in the field is
done by a table
lookup.
 * The two
tables are
present in a
file, and need
to be stored in
the class.
 * The following
two arrays are
used for this
purpose.
 */
private static
byte[][]
multTable = new
byte[FieldSize][
FieldSize];
private static
byte[][]
expTable = new
byte[FieldSize][
FieldSize];

/* Sometimes it
is required to
move from the
normal
representation
of an element
 * of GF^*[256]
to generator
power
representation.
The following
two arrays
 * allow
conversion in
both directions.
 */
private static
byte[]
powerToNormal =
new
byte[FieldSize];
private static
byte[]
normalToPower =
new
byte[FieldSize];
private static
byte[]
primePowers =
new
byte[noPrimePowe
rs];
private static
byte[]
invPrimePowers =
new
byte[noPrimePowe
rs];
else { // large
N
for (i = 0; i <
N; I++)
for (j = 0; j <
N; j++)
M[i][j] = (i ==
j ? ONE : ZERO);
} // end of
class definition
```