Data Structures and Algorithms(CSE2001)

MODULE 1

Introduction to Data Structure and Linear Data Structure

PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013
PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

Introduction

What are Data Structures?

- Data structures are containers for storing and organizing data in a computer's memory.
- They define how data is stored, accessed, and manipulated.
- A data structure is a way to store and organize data in order to facilitate access and modifications. No single data structure works well for all purposes, so it is important to know the strengths and limitations of several of them.

Different types of Data Structures

Depending on the organization of the elements, data structures are classified into two types:

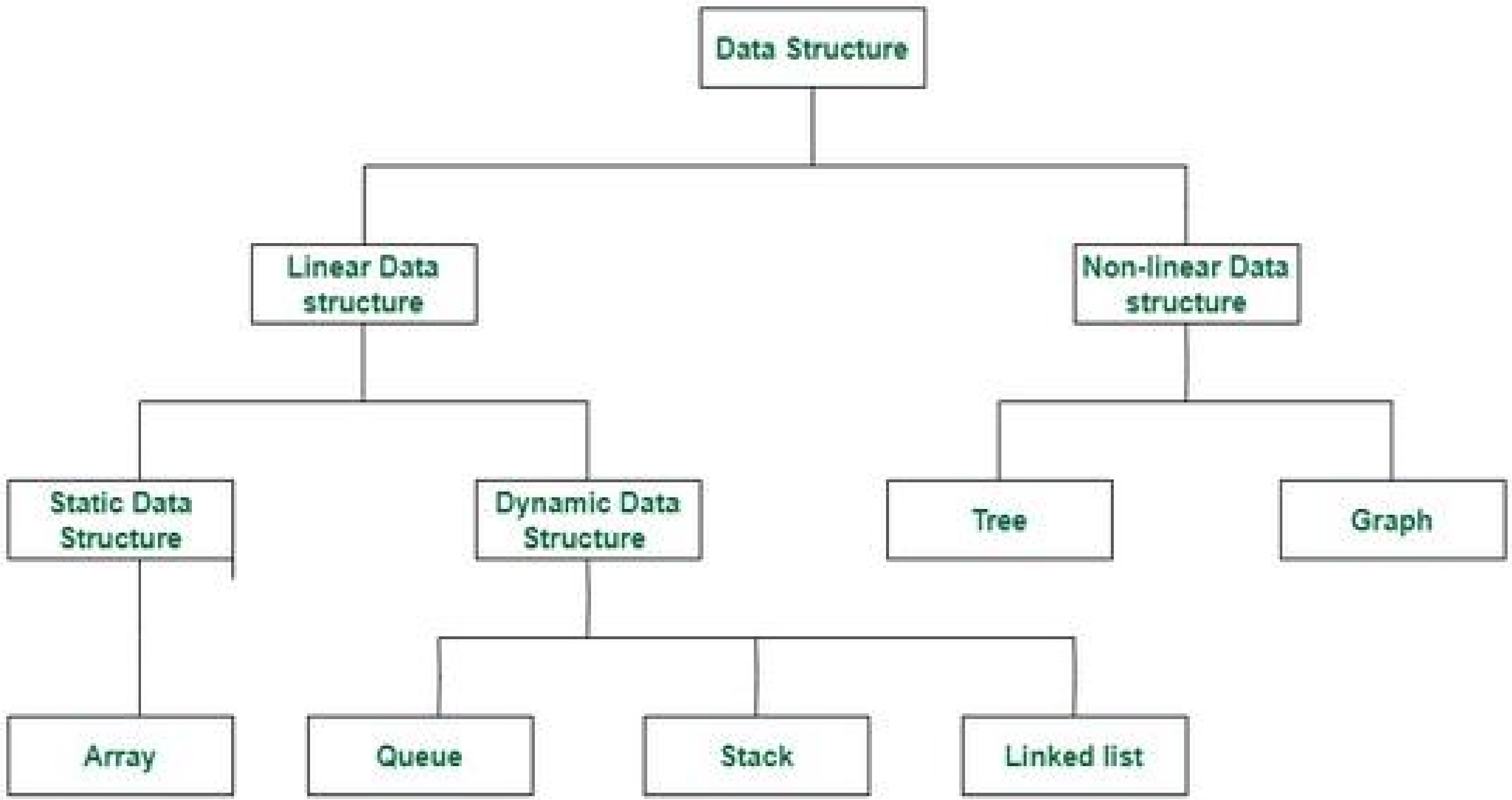
- Linear data structures: Elements are accessed in a sequential order but it is not compulsory to store all elements sequentially (say, Linked Lists).

  Examples: Linked Lists, Stacks and Queues.

- Non – linear data structures: Elements of this data structure are stored/accessed in a non-linear order.

  Examples: Trees and graphs.

Classification of Data Structure



Data Structure

Linear Data structure

Non-linear Data structure

Static Data Structure

Dynamic Data Structure

Tree

Graph

Array

Queue

Stack

Linked list

Abstract Data Type(ADT)

- **Abstract data type:** is a collection of data and the operations that can be performed on the data. In the extreme, the data that comprise the ADT are not directly accessible from the outside. When implemented in this manner, the only way to get at the data (or modify it) is through the defined operations on the ADT.

Abstract Data Type

- Computer programs also need to organize their data.
  - Examples: a list, a stack, a dictionary, and etc..
- An **abstract data type,** or **ADT**:
  - A specification that describes **a data set** and the **operations** on that data.
  - Each **ADT** specifies what **data** is stored and what the **operations** on the data do.
  - An **ADT** does not indicate how to store the data or how to implement the operations

Overview of Java

Introduction

- Java is a computer programming language.
- Java was created based on the C & C++. Java uses **C syntax** and many of the **object oriented features** are taken from **C++**.
- Before **Java** was invented there were other languages like COBOL, FORTRAN, C, C++, Small Talk etc., These languages had few disadvantages which were corrected in Java
- **Java** also **innovated many new features to solve the fundamental problems** which the previous languages could not solve.

Features of Java

- Simple
- Object Oriented
- Robust
- Multi Threading
- Architecture Neutral
- Interpreted and High Performance
- Distributed
- Dynamic

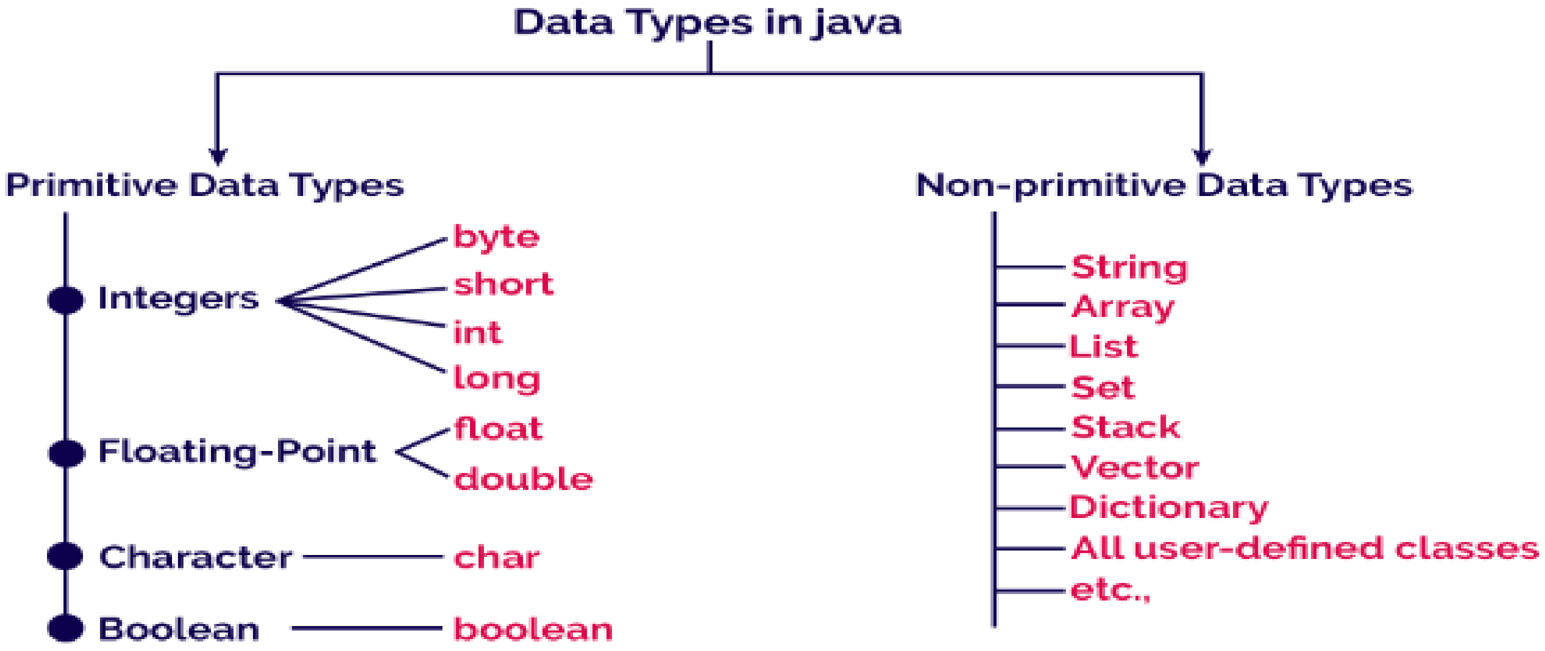
Java is an Object-Oriented Language. Java supports the following fundamental concepts –

- **Polymorphism**
- **Inheritance**
- **Encapsulation**
- **Abstraction**
- **Classes**
- **Objects**
- **Instance**
- **Method**
- **Message Passing**

BASICS OF JAVA

Java Data types

- Primitive Data Types
- Non-primitive Data Types



Primitive Data types

Base Types:

- boolean a boolean value: true or false
- char 16-bit Unicode character
- byte 8-bit signed two’s complement integer
- short 16-bit signed two’s complement integer
- int 32-bit signed two’s complement integer
- long 64-bit signed two’s complement integer
- float 32-bit floating-point number (IEEE 754-1985)
- double 64-bit floating-point number (IEEE 754-1985)

Declarations and initializations of several base-type variables.

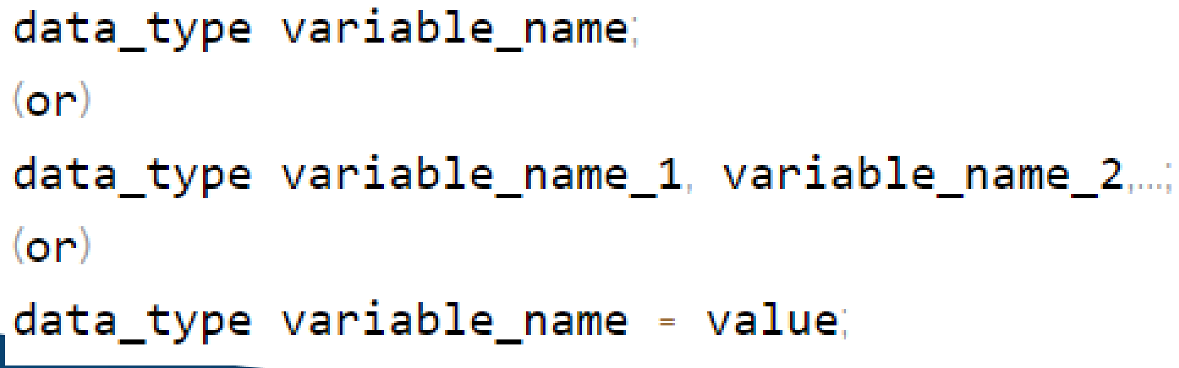
```
boolean flag = true;
boolean verbose, debug; // two variables declared, but not yet initialized
char grade = 'A';
byte b = 12;
short s = 24;
int i, j, k = 257; // three variables declared; only k initialized
long l = 890L; // note the use of ”L” here
float pi = 3.1416F; // note the use of ”F” here
double e = 2.71828, a = 6.022e23; // both variables are initialized
```

Java Variables

In java programming language, variables are classiffied as follows.

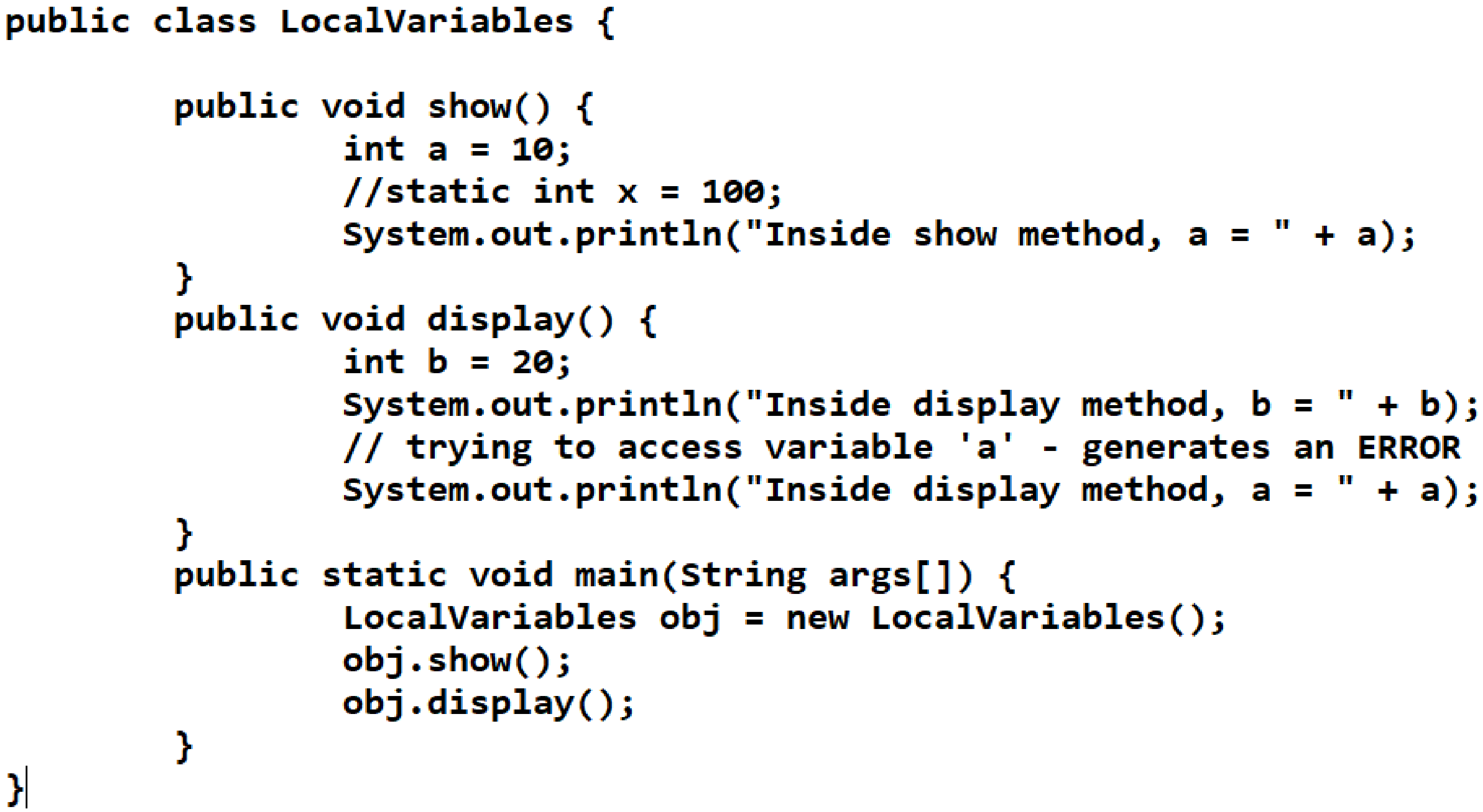
- **Local variables**
- **Instance variables or Member variables or Global variables**
- **Static variables or Class variables**
- **Final variables**

Syntax to declare variables:



```
data_type variable_name;
(or)
data_type variable_name_1, variable_name_2,...;
(or)
data_type variable_name = value;
```

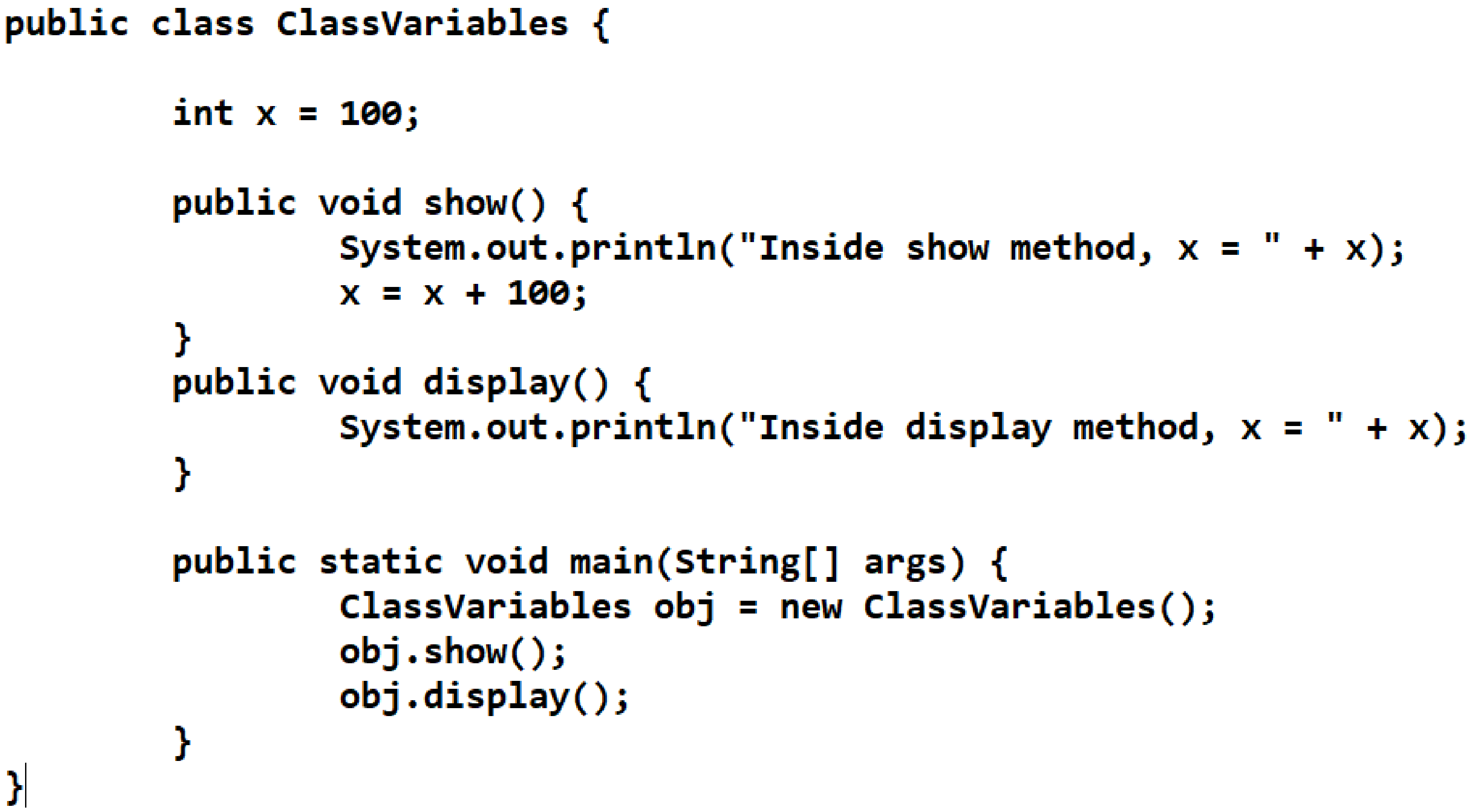
Java code to demonstrate local variables visibility



```java
public class LocalVariables {

        public void show() {
                int a = 10;
                //static int x = 100;
                System.out.println("Inside show method, a = " + a);
        }
        public void display() {
                int b = 20;
                System.out.println("Inside display method, b = " + b);
                // trying to access variable 'a' - generates an ERROR
                System.out.println("Inside display method, a = " + a);
        }
        public static void main(String args[]) {
                LocalVariables obj = new LocalVariables();
                obj.show();
                obj.display();
        }
}
```

Java program to illustrate instance or global variable.



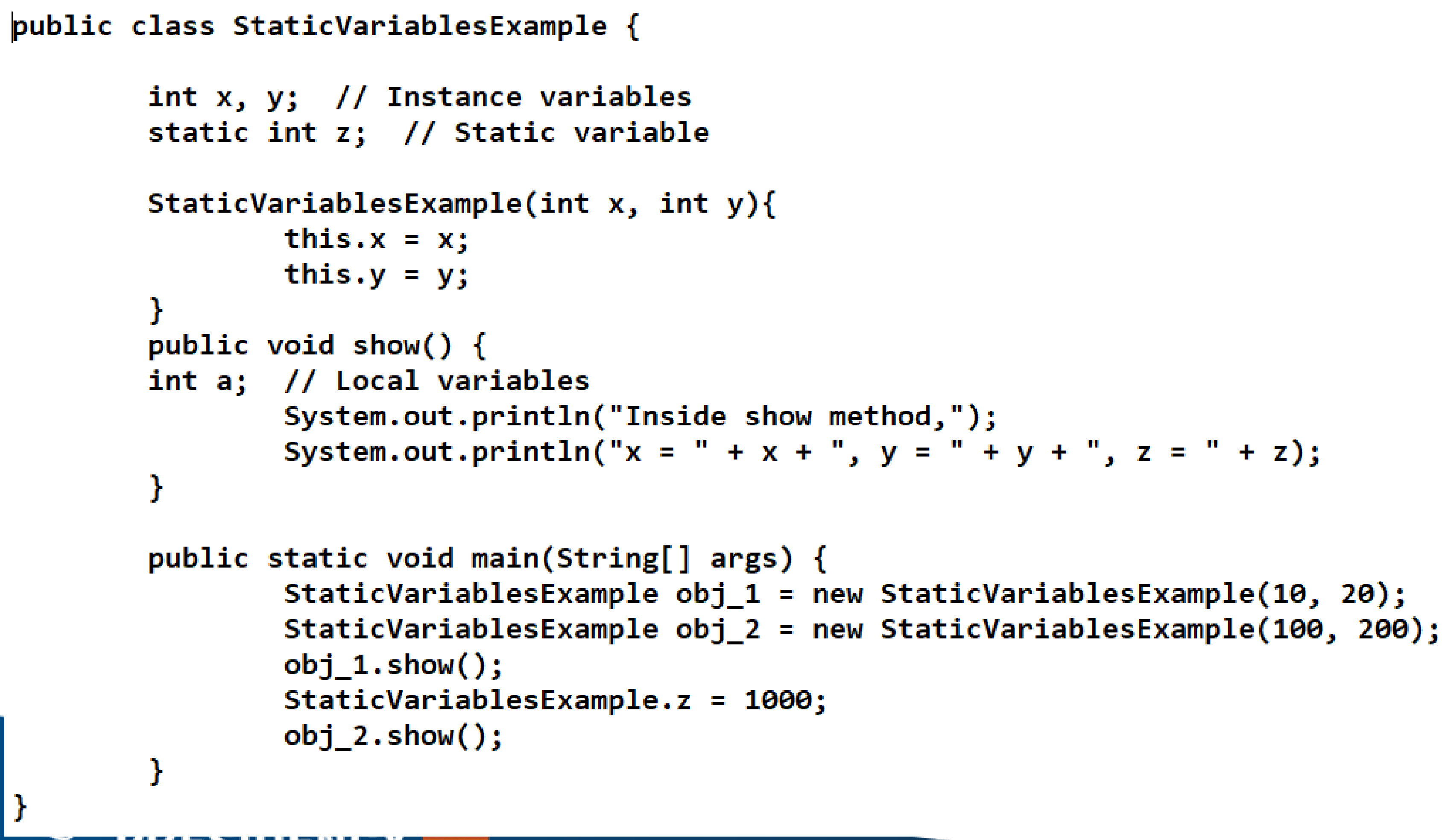
```java
public class ClassVariables {

        int x = 100;

        public void show() {
                System.out.println("Inside show method, x = " + x);
                x = x + 100;
        }
        public void display() {
                System.out.println("Inside display method, x = " + x);
        }

        public static void main(String[] args) {
                ClassVariables obj = new ClassVariables();
                obj.show();
                obj.display();
        }
}
```

Java program to illustrate static variable



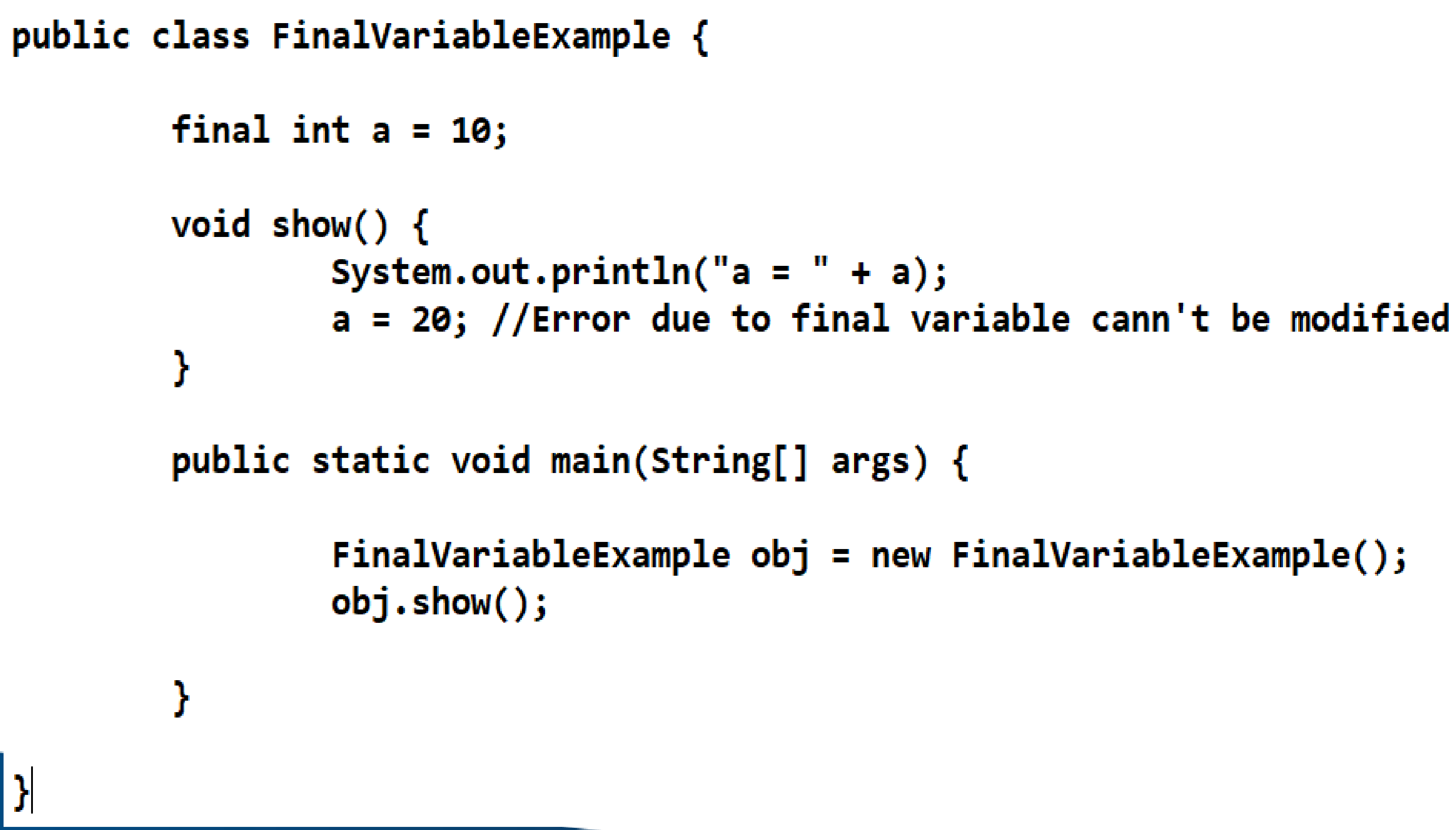
```
public class StaticVariablesExample {

        int x, y;  // Instance variables
        static int z;  // Static variable

        StaticVariablesExample(int x, int y){
                this.x = x;
                this.y = y;
        }
        public void show() {
        int a;  // Local variables
                System.out.println("Inside show method,");
                System.out.println("x = " + x + ", y = " + y + ", z = " + z);
        }

        public static void main(String[] args) {
                StaticVariablesExample obj_1 = new StaticVariablesExample(10, 20);
                StaticVariablesExample obj_2 = new StaticVariablesExample(100, 200);
                obj_1.show();
                StaticVariablesExample.z = 1000;
                obj_2.show();
        }
}
```

Java program to illustrate final variable



```
public class FinalVariableExample {

        final int a = 10;

        void show() {
                System.out.println("a = " + a);
                a = 20; //Error due to final variable cann't be modified
        }

        public static void main(String[] args) {

                FinalVariableExample obj = new FinalVariableExample();
                obj.show();

        }

}
```

Classes in Java

Introduction

- The entire concept of Object Oriented Programming is not about objects, its about class. Because we use classes to create objects…..
- The class describes what an object will be. It's a blue print of an object and it is the description and definition of an object.

How to Create a class?

- To create a class first, we need to find the following....

Name - What is it ? type

Student, Employee, BankAccount, Document, Car, Box.....

Attributes - What describes it ? Properties, data

name, salary, balance, size, color, width.....

Behavior - What can it do ? operations

read, play, search, save, create, print, close, open, deposit.....

Example class

- Creating a Bank Account class

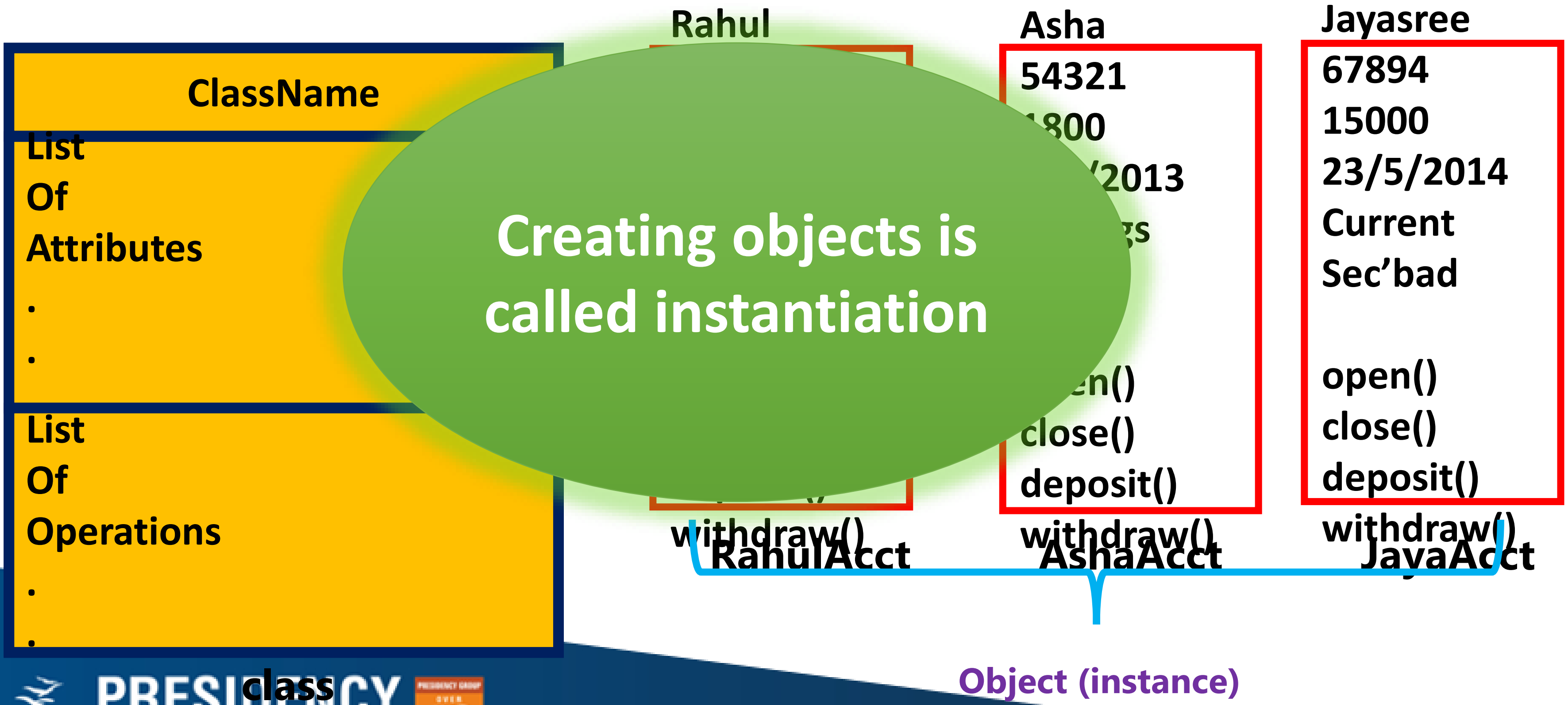
Name - BankAccount

Attributes - name, AccNo, balance, dateOpened, accountType, branch

Behavior - open( ), close( ), deposit( ), withdraw( )

Example class

- A class is represented in a rectangle box with 3 rows..



ClassName

List
Of
Attributes
.
.

List
Of
Operations
.
.

class

Rahul

withdraw()
RahulAcct

Asha
54321
800
/2013
gs

en()
close()
deposit()
withdraw()
AshaAcct

Jayasree
67894
15000
23/5/2014
Current
Sec'bad

open()
close()
deposit()
withdraw()
JayaAcct

Object (instance)

Creating objects is called instantiation

Creating object of a class

```
{class  BankAccount
         String  name ;
         int  accountNumber;
         double   balance ;
         String   dateOpened;
         String   accountType;
         String  branch;

         void  open( ){….}
         void  close( ){….}
         void  deposit( ){….}
         void   withdraw( ){….}
 }
```

Syntax for object creation

```
ClassName   objectName = new  ClassName( );
```

Example for object creation

```
BankAccount   rahulAcct = new  BankAccount( );
```

```
BankAccount   ashaAcct = new  BankAccount( );
```

```
BankAccount   jayaAcct = new  BankAccount( );
```

ARRAYS in JAVA

- Array is collection of related data items
- Creating an array
  - Declare an array
  - Create memory location
  - Putting values to memory locations

Declaring an Array Variable

- Do not have to create an array while declaring array variable
  - <*type*> [ ] variable_name;
  - *double[ ] myList;*
  - *double myList[ ];*
- Both syntaxes are equivalent
- No memory allocation at this point

Defining an Array

Define an array as follows:

- variable_name=new <*type*>[arraySize];
- Number = new int[5];
- Mylist = new int[10];

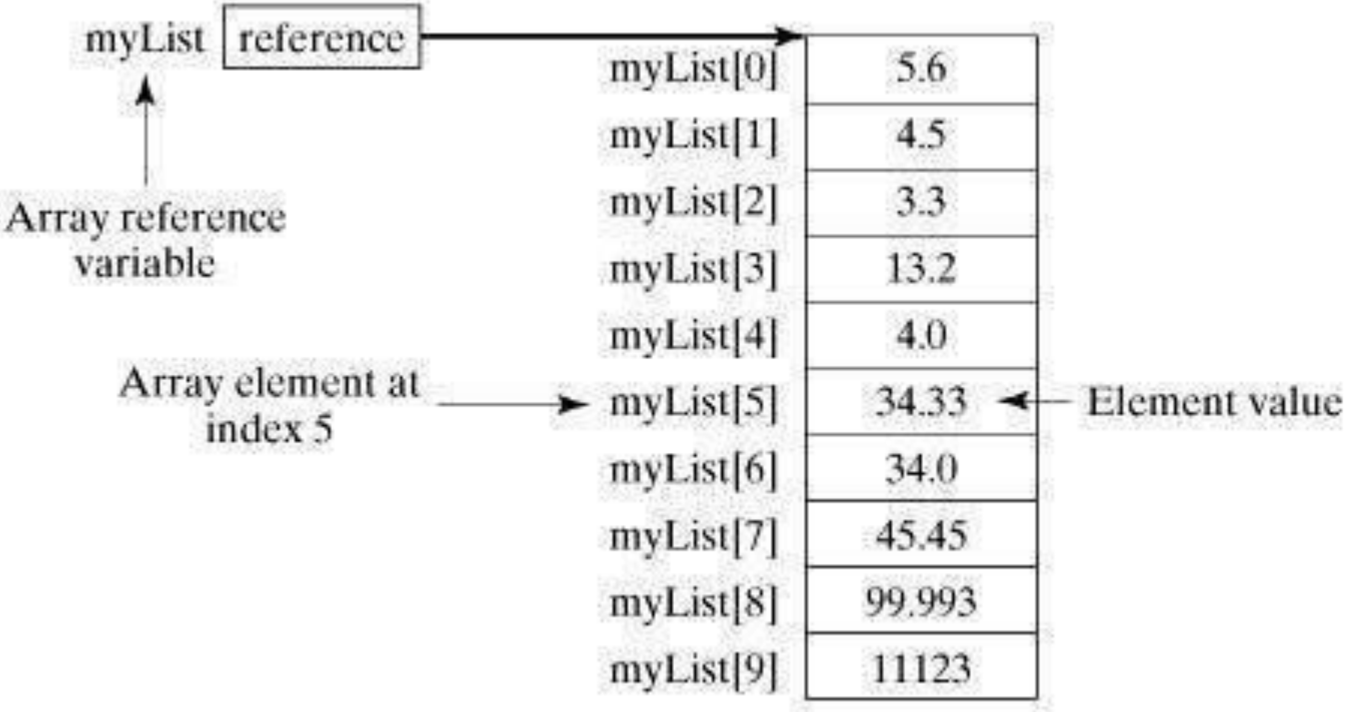
It creates an array using new dataType[arraySize];

- It assigns the reference of the newly created array to the variable variable_name.
- dataType arrayname[ ] = {list of values};
- int a [ ]={1,2,3,4,5,6,7};
- Array index starts from 0 to arraySize-1;
- int is of 4 bytes, total space=4*10=40 bytes

Declaring and defining in the same statement:

- *double* [ ] mylist = new *double*[10];

Creating arrays cntd...



myList[0]	5.6
myList[1]	4.5
myList[2]	3.3
myList[3]	13.2
myList[4]	4.0
myList[5]	34.33
myList[6]	34.0
myList[7]	45.45
myList[8]	99.993
myList[9]	11123

Example: Ms. White's Test Score Analyzer

Problem:

Ms. White needs to analyze students' test performance. She needs a program to display a name and let her enter the test score. Then it should compute and display the average. Finally it should give a report with names, scores, and deviation from the average.