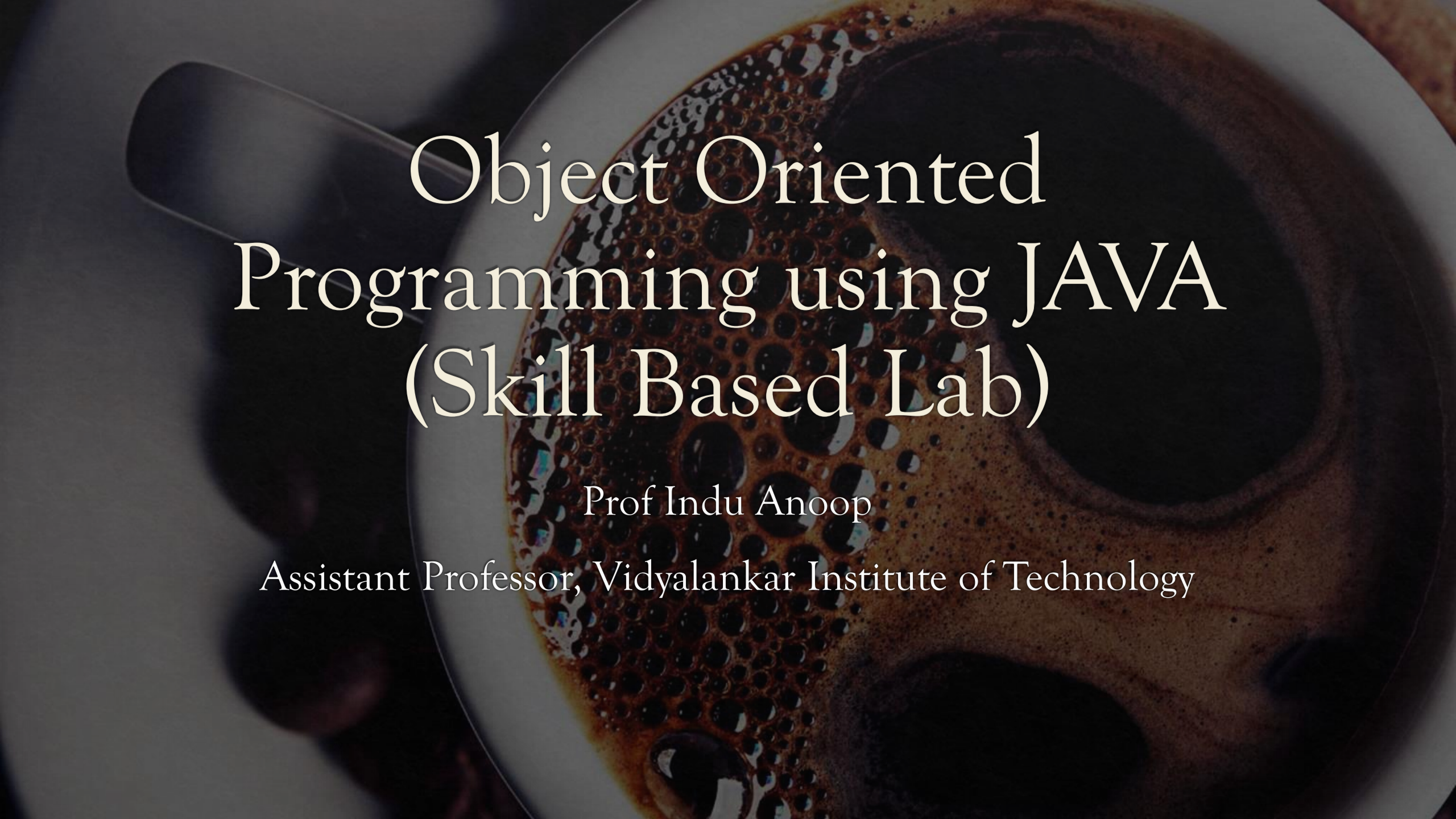


Object Oriented Programming using JAVA (Skill Based Lab)

Prof Indu Anoop

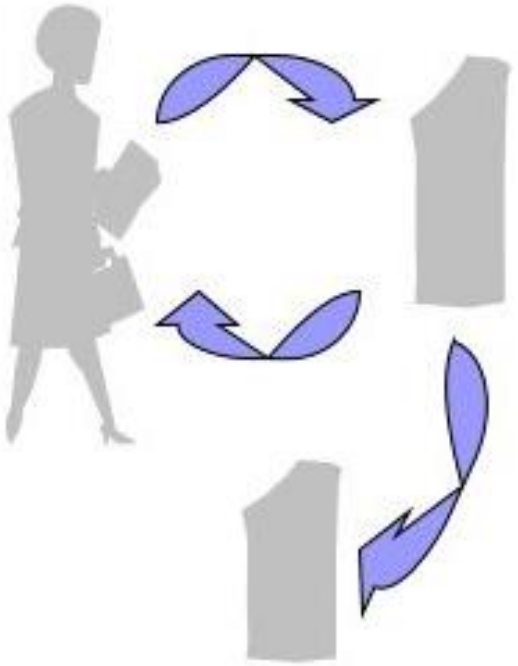
Assistant Professor, Vidyalankar Institute of Technology

What is the Difference

Srl.No	Parameter	POP	OOP
1	Definition	POP stands for Procedural Oriented Programming.	OOP stands for Object Oriented Programming.
2	Approach	POP follows top down approach.	OOP follows bottom up approach.
3	Division	A program is divided into functions and they interactions.	A program is divided to objects and their interactions.
4	Inheritance support	Inheritance is not supported.	Inheritance is supported.
5	Access control	No access modifiers are supported.	Access control is supported via access modifiers.
6	Data Hiding	No data hiding present. Data is globally accessible.	Encapsulation is used to hide data.
7	Example	C, Pascal	C++, Java

Actual Real life Difference (OOP and POP)

- Procedural



Withdraw, deposit, transfer

- Object Oriented

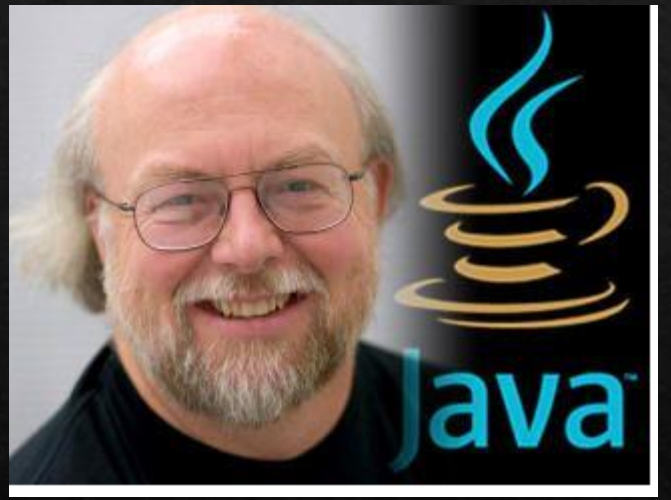
Customer, money, account

Definition of Object Oriented Programming?

- OOP is a programming approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand
- Thus, OOP treats **data** as the critical element in the program which can be accessed only by **methods** associated with that **object**

What is Java? (Brief History)

- Language Designer: James Gosling in 1991
- Developers: Sun Microsystems -> Bought by Oracle in 2010
- The first public implementation was **Java** 1.0 in 1995.
- What we will be using is Java 1.8 (as per syllabus)
- Latest version is Java 16
- Java Name : Coffee Beans in Island called Java in Indonesia



How to install Java?

Steps to install Java

- System Environment Configuration After Installation
- Copy Path to the bin folder inside JDK

  C:\Program Files (x86)\Java\jdk1.8.0_101\bin

- Enter this path inside system Environment variable

Basic Concepts of OOP (Used during Programming)

1.Object

2.Class

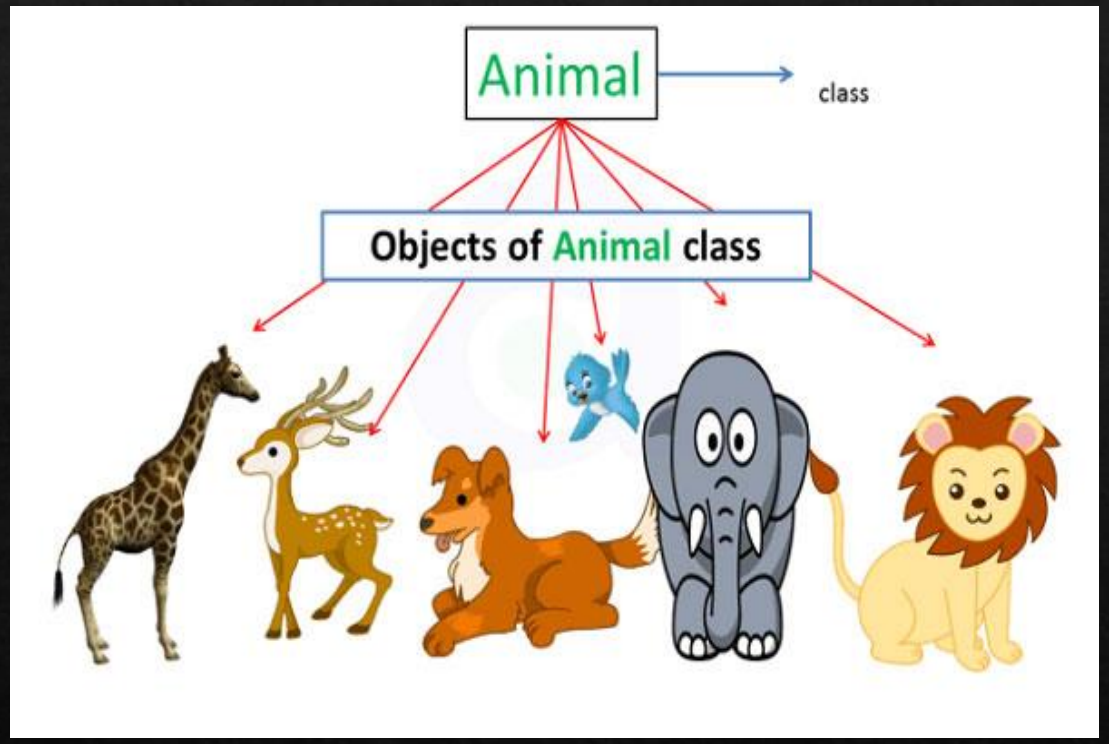
3.Data Abstraction

4.Encapsulation(Data Hiding)

5.Inheritance.

6.Polymorphism

1. Object



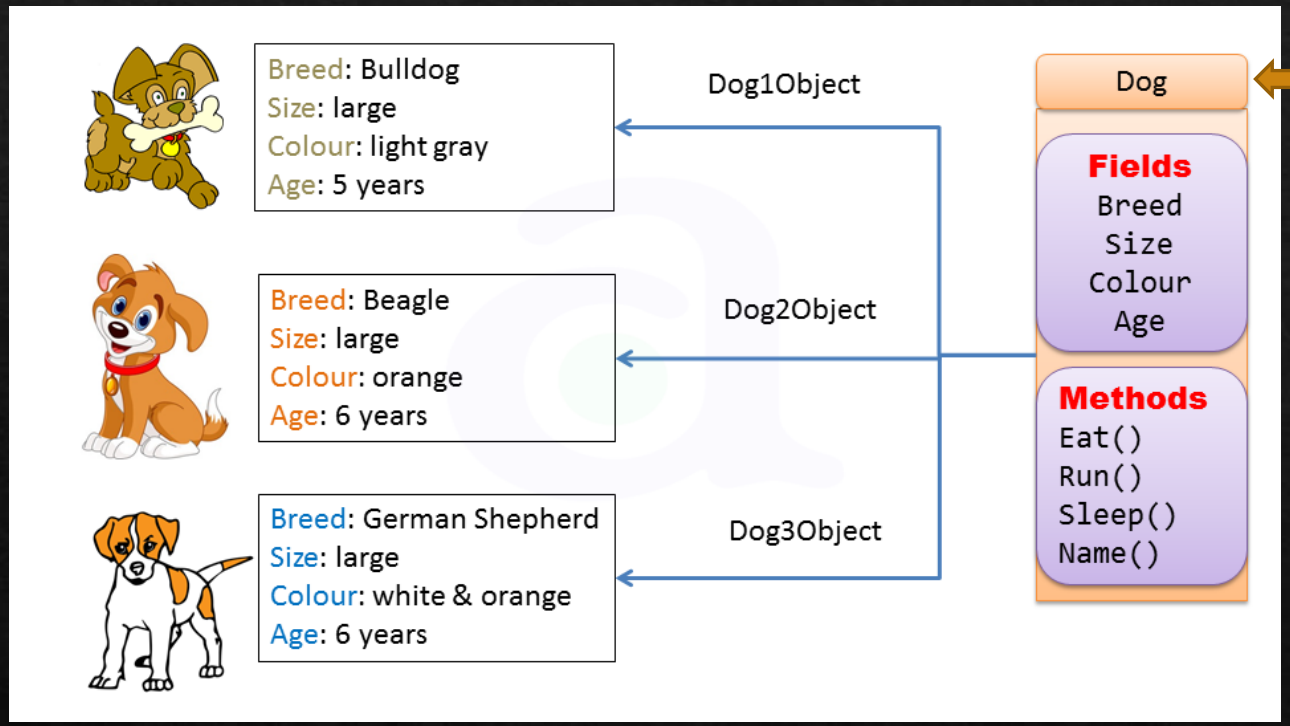
- Object are defined as the instance(representation) of a class.
- Example table, chair, are all instance of the class Furniture.
- Object of a class will have same attribute and function(method) which are defined in that class.
- The only difference between objects of same class will be in values of attribute.
- Each Object Entity have unique identity and behavior

For BANKING SYSTEM → Objects are Customer, Account etc.

Person as an Object

Attributes	State	Identity	Behavior	Responsibility
empid	empid =10	empid =10	Swipe Card	To be present in the office on time
name	name= "Dhruv"		Fill Timesheet	Do allotted work
gender	gender ="male"		Wear ICard	Identity with organization
age	age=27		Calculate salary	Get salary
address	address = "Pune"		Print details	Provide personal information
phone	phone = 9823013451			
basic salary	basicSalary = 9000			

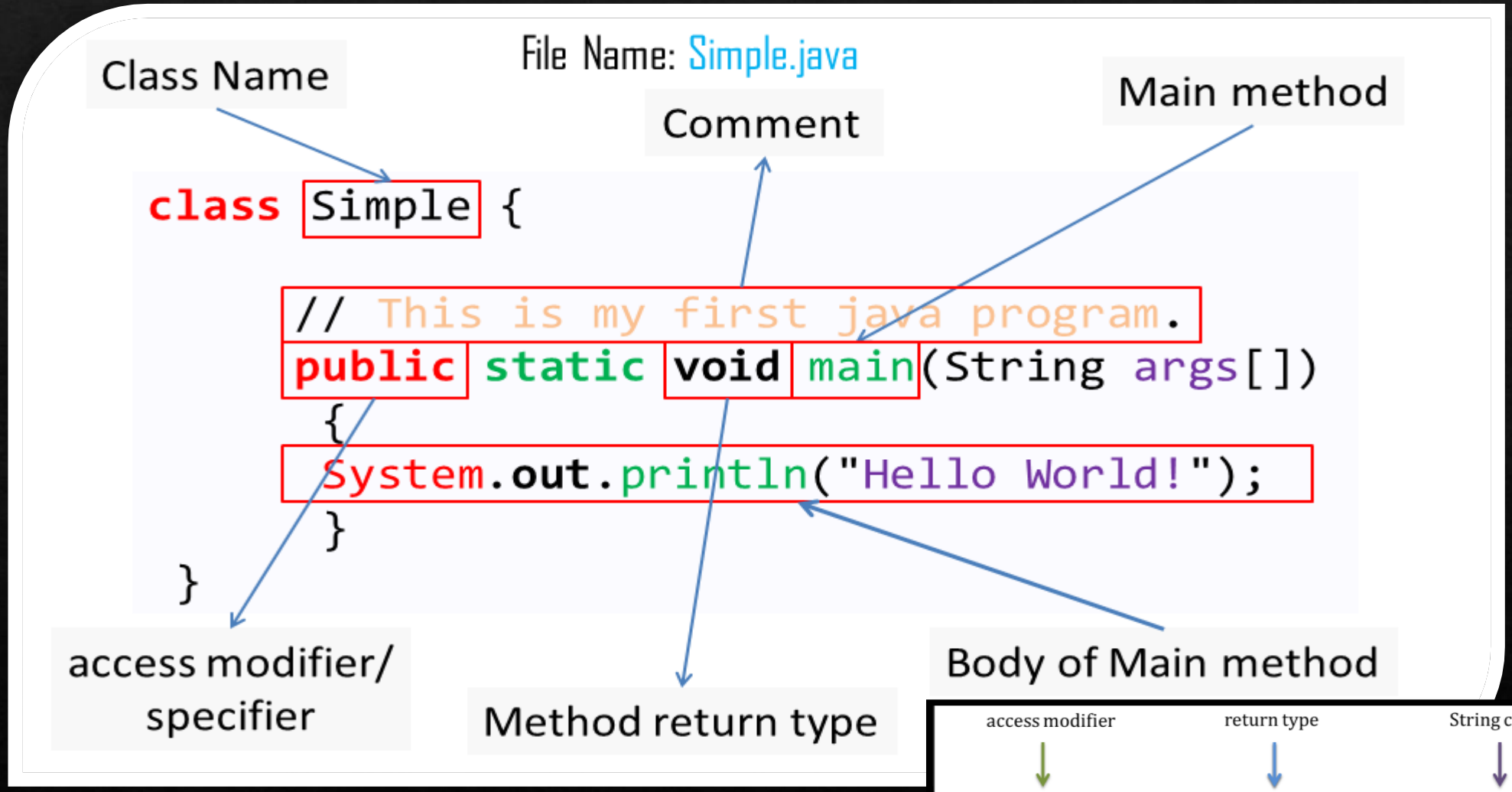
2. Class

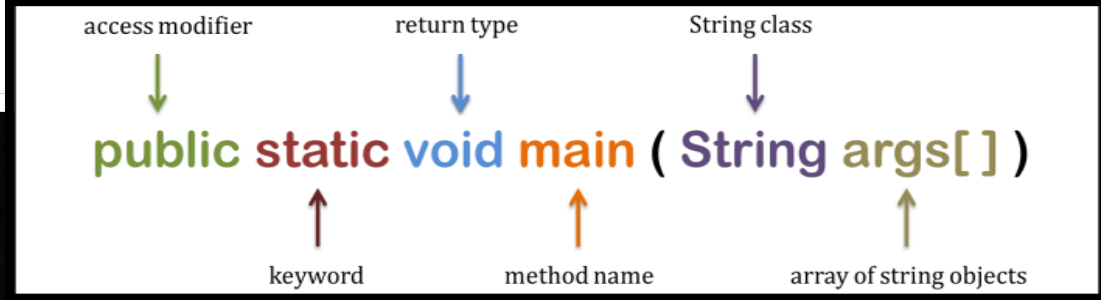


- A class is defined as a collection of similar objects
- It acts as the blueprint for an object
- An object is not created by just defining class, object need to be created individually.
- Classes are logical in nature mean they do not have physical existence.
- Example : Furniture, employee, student don't exist physically

Let's see how a simple Java Class looks like....

File Name: Simple.java





Java Class Syntax

```
class  Class_Name
{
 variable_Declaration;

method_Declaration(){}


public static void
main(String args[])
{
        . . .
}

}//end of class
```

By Convention:

class name starts with Capital case
e.g. class **M**yDate

Variable (attribute or property) name starts with small case
e.g. int **d**ate

Method(behavior) name starts with 1st word small case then next words 1st letter Capital case called 'camel**C**ase'
e.g. void **p**rint**D**ate(){}

Writing Java Class

```
/* This is Comment */
// This is another comment

class MyDate
{
    private int day,month,year;

    public void initDate()
    {
       day=30;
       month=6;
       year=2011;
    }
    public void printDate()
    {
      System.out.println("Date is :"+ day+"/"+month+"/"
       +year);
    }
```

Create an Object using new Keyword

```
    public static void main(String args[])
    {
         MyDate d;
         d=new MyDate();

              // OR

         MyDate d=new MyDate();

              //method invocation
          d.initDate();
          d.printDate();


    }
}
```

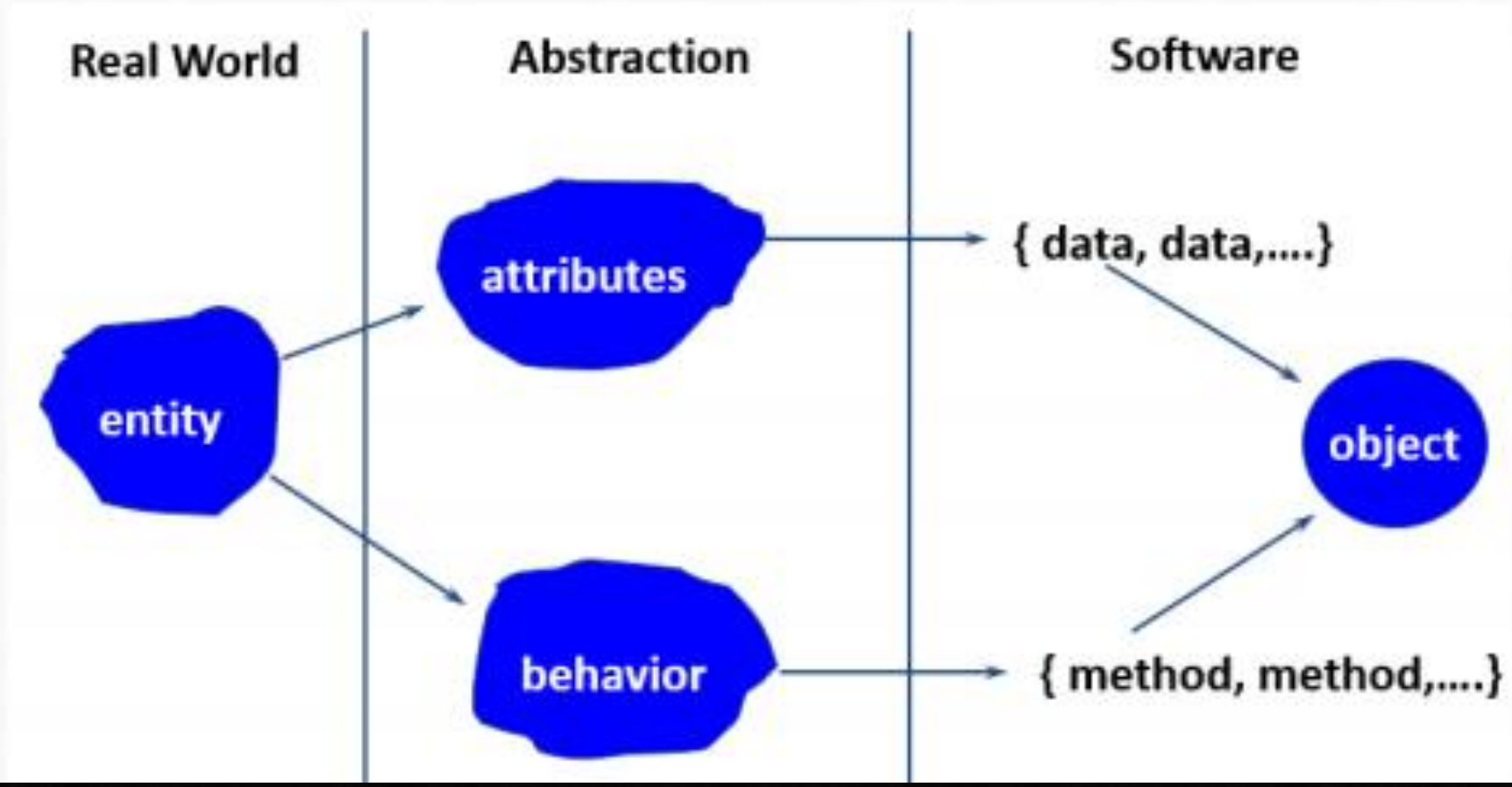
3.Abstraction

- Abstraction is the process of identifying the key aspects of an entity and ignoring the rest.
- Only those aspects are selected that are important to the current problem scenario.
- Example : Abstraction of a person object
  - Enumerate attributes of a "person object" that needs to be created for developing a database.
    - useful for social survey
    - useful for health care industry
    - useful for payroll system

Abstraction of a `Person` Object

Social Survey	Health Care	Payroll System
name	name	name
age	age	age
marital status	---	---
religion	---	---
income group	---	---
address	address	address
occupation	occupation	occupation
---	blood group	---
---	weight	---
---	previous record	---
---	---	basic salary
---	---	department
---	---	qualification

Abstraction



Encapsulation

- The process of binding data members and member functions into a single unit called class. It achieves data hiding and implementation hiding

- It occurs at Implementation Level

Inheritance

- Classification helps in handling complexity.
- Inheritance is the process by which one object can acquire the properties of another object.
  - Broad category is formed and then sub-categories are formed.
- "is – a" kind of hierarchy.

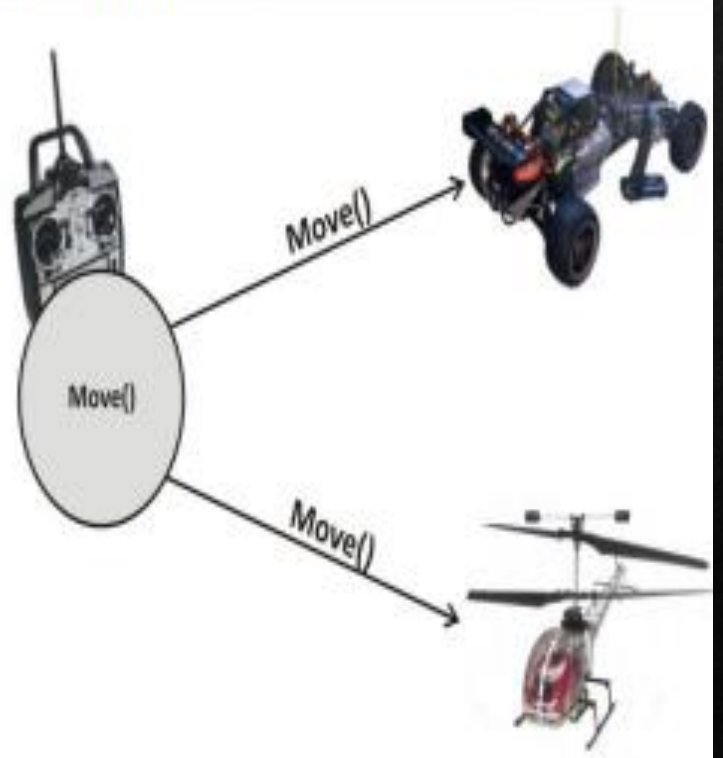
Examples of Inheritance

- Inheritance is used:
  - When you want to force the new type to be the same type as the base class.
  - Computer system is an electronic device.
  - Car is a vehicle.

Polymorphism

- The ability of different types of related objects to respond to the same message in their own ways is called polymorphism
- Polymorphism helps us to :
  - Design extensible software; as new objects can be

added to the design without rewriting existing procedures.



Let's Recollect...

- What are the features of Java?
- How Java is secure?
- Why Java is called platform independent ?
- What is a Java Virtual Machine?
- Explain execution flow of Java Application.
- What is a '`class`'?

Reserved Words

Datatype (8)	Flow Ctrl (11)	Modifiers (11)	Exception (6)	Class (6)	Object (4)	Return (1)	Unused (2)	Reserved Literals(3)	Enum (1)
byte	if	public	try	class	new	void	goto	true	enum (1.5 v)
short	else	private	catch	package	instanceof		const	false	
int	switch	protected	finally	import	super			null	
long	case	static	throw	extends	this				
float	default	final	throws	implements					
double	while	abstract	assert (1.4 v)	interface					
char	do	synchroniz-ed							

Let's see which is valid/invalid keyword

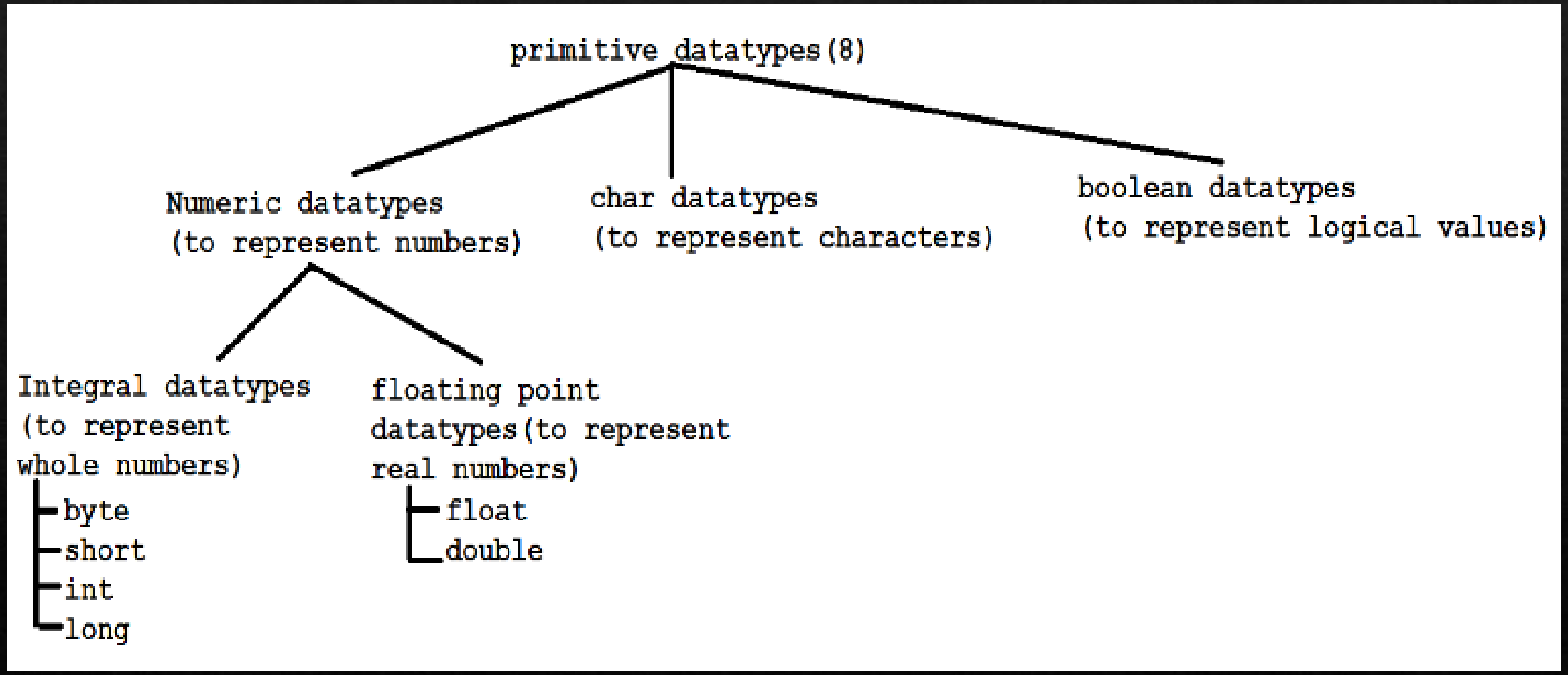
1.	public	(valid)
2.	static	(valid)
3.	void	(valid)
4.	main	(invalid)
5.	String	(invalid)
6.	args	(invalid)

Datatypes: Every variable has a type

Datatype	Size	Range	Corresponding Wrapper class	Default Value	Example
byte	1 byte	-2^7 to 2^7-1 (-128 to 127)	Byte	0	byte b=10; handle data in terms of streams either from file or network
short	2 bytes	-2^15 to 2^15-1 (-32768 to 32767)	Short	0	short s=130; Used for 8086 processors (16 bit) but outdated
int	4 bytes	-2^31 to 2^31 -1	Integer	0	int i=130; most commonly used

long	8 bytes	-2^63 to 2^63 -1	Long		long l=f.length() ; //f is a file
					when int not enough to hold big values

float	4 bytes	-3.4e38 to 3.4e38	Float	0.0	For 5-6 decimal places accuracy Single precision
double	8 bytes	-1.7e308 to 1.7e308	Double	0.0	For 14-15 decimal places accuracy Double precision
boolean	Not applicable	Not applicable (but allowed values true|false)	Boolean	false	
char	2 bytes	0-65535	Character	0 (represents blank space)	Java is Unicode based (no of Unicode char >256 and <=65536) so 1 byte not enough


primitive datatypes(8)
Numeric datatypes
(to represent numbers)
char datatypes
(to represent characters)
boolean datatypes
(to represent logical values)
Integral datatypes
(to represent
whole numbers)
byte
short
int
long
floating point
datatypes(to represent
real numbers)
float
double

Control Statements

1. If Statement

Syntax

```
if(condition)
    statement;
```

What is the condition?

condition is a Boolean expression (true or false.)

Operator	Meaning
<	Less than
>	Greater than
==	Equal to

Example

```
if(num < 100)
System.out.println("num is less than 100");
```

Control Statements

2. If-else Statement

Syntax

```
if(condition)
    statement1;
else
     statement2;
```

What is the condition?

condition is a Boolean expression (true or false.)

Example

```
if(num== 100)
System.out.println(“You got full marks");
else
System.out.println(“You missed out on the top score”);
```

Control Statements

3. Else if

Syntax

```
if(condition)
    statement1;
else if(condition)
    statement2;
else
    statement3;
```

What is the condition?

condition is a Boolean expression (true or false.)

Example

```
if(num < 50)
System.out.println("num is less than 50");
elseif (num < 80)
System.out.println("num is between 50 and 80");
else
System.out.println("num is greater than 80");
```

Control Statements

3. Nested if-else

Syntax

```
if(condition) {
    if(condition)
        statement1
    else
        statement2
}
else
    statement3;
```

Example

```
if( num > 60) {
        if( num > 75)
        System.out.println("Distinction");
        else
        System.out.println("First Class")
}
else
System.out.println("Second Class");
```

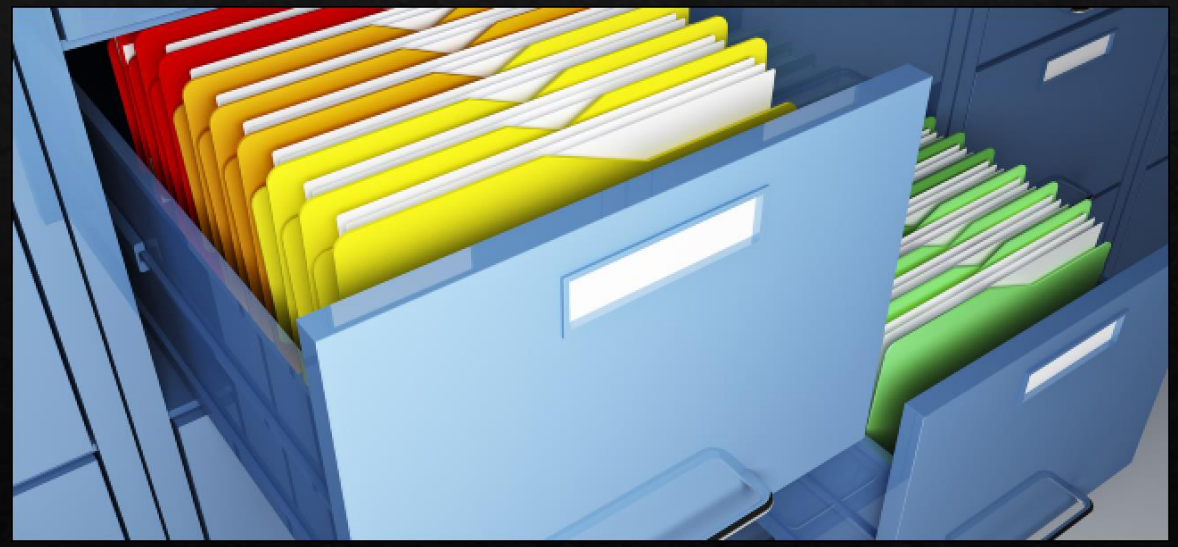
Control Statements

4. Switch Statement: Complexity of program increases as number of alternatives increases, so we need a multiway decision statement. A switch statement tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statements associated with that case is executed.

Syntax

```
switch(expression) {
        case value1: block1
                     break;
        case value2: block2
                     break;
        ...
        ...
        default: default-block
                 break;
}
```

Metaphor :Case of Files



So where is switch statement useful??

- Making a Menu Based Program
  - Eg :
    - 1.Menu1
    - 2.Menu2
    - 3.Menu3
    - 4.Menu4

```
Get choice input  from user
Switch(choice)
{
    case 1: System.out.println("You have chosen to view files 1"); break;
    case 2: System.out.println ("You have chosen to view files 2"); break;
   ...
   default: System.out.println("Invalid Choice");
}
```

Iteration/Looping Statements

1. For Loop

Syntax

```
for(initialization; condition; iteration)
    statement;
```

Example

```
for(int i=0;i<6;i++) {
System.out.println("Counter: " +i);
}
```

Output

```
counter: 1
counter: 2
counter: 3
counter: 4
counter: 5
```

Iteration Statements

2. While Loop : repeats a statement or block while its controlling expression is true

Syntax

```
while(condition) {
    //body of loop
}
```

Example

```
int n=5;
while(n>0) {
System.out.println("Counter: " +n);
n- - ;
}
```

Output

```
counter: 5
counter: 4
counter: 3
counter: 2
counter: 1
```

Iteration Statements

3. do-While Loop : always executes its body at least once , because it's conditional expression is at the bottom of the loop. Executes body of loop first and then checks condition, if true, then repeats otherwise terminates

Syntax

```
do{
//body of loop
} while (condition)
```

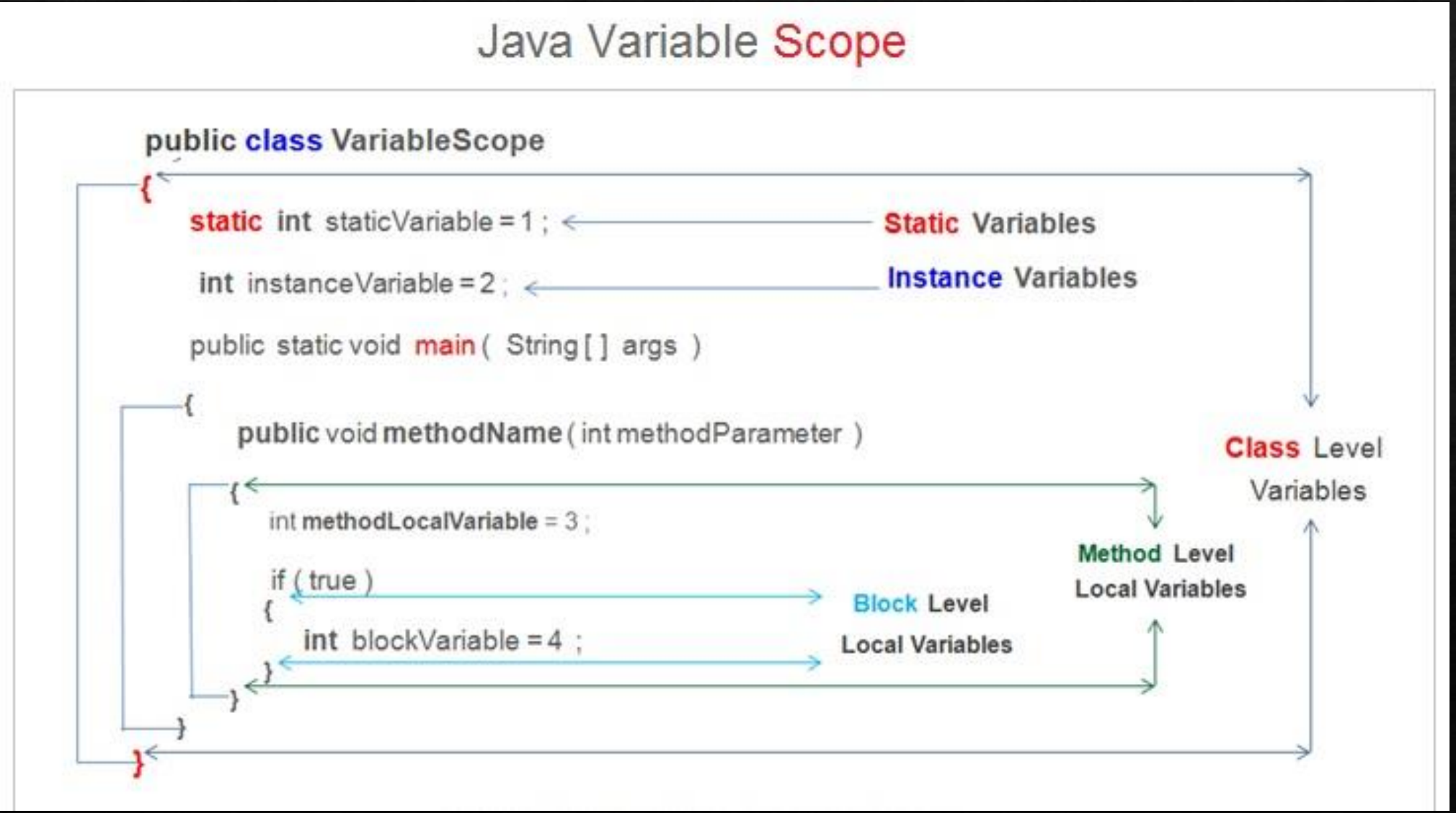
Example

```
int n=5;
while(n>0) {
System.out.println("Counter: " +n);
n- - ;
}
```

Output

```
counter: 1
counter: 2
counter: 3
counter: 4
counter: 5
```

Scope of Variables



Scope of Variables

Instance Variable: Associated with objects and declared inside the class

Class Variable: Global to a class and belong to entire set of objects that class creates [only one memory location is created for each variable]

Local variable: Variables declared and used inside methods [Not available to use outside method definition]

Arithmetic Operators

a - b =10

a + b = 18

a * b = 56

a / b =3 (decimal part truncation)

a % b = 2 (remainder of integer division)

Mixed Mode Operators

15 /10.0 produces the result 1.5

15/10 produces the result 1

Relational Operators

Operator	Meaning
<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Is equal to
!=	Is not equal to
+=	Summation and Assignment

Let's solve (Exp1: Check assignment section of teams for experiment submission)

- **Problem Statement1 :** Create a java program to grade students >=75 Distinction, 60-75:First Class,50-60 : Second class, <50 : Fail
- **Problem Statement 2:** Write a program using vector to store list of 'n' number of students and perform following operation on it:
    - Insert a new name in vector
    - Delete a name from vector
    - Display contents of vector
    - Exit

ArrayList

- The limitation with array is that it has a fixed length so if it is full you cannot add any more elements to it, likewise if there are number of elements gets removed from it the memory consumption would be the same as it doesn't shrink.
- ArrayList can dynamically grow and shrink after addition and removal of elements

Vector

- The Vector class implements a growable array of objects. Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.

```
Vector vec = new Vector();
```

- It creates an empty Vector with the default initial capacity of 10. It means the Vector will be re-sized when the 11th elements needs to be inserted into the Vector. Note: By default vector doubles its size. i.e. In this case the Vector size would remain 10 till 10 insertions and once we try to insert the 11th element It would become 20 (double of default capacity 10).

When to use ArrayList and when to use vector?

It totally depends on the requirement. If there is a need to perform **"thread-safe"** operation the vector is your best bet as it ensures that only one thread access the collection at a time.

Performance: Synchronized operations consumes more time compared to non-synchronized ones so if there is no need for thread safe operation, ArrayList is a better choice as performance will be improved because of the concurrent processes.