

Welcome to **JSPIDERS BASAVANAGUDI**

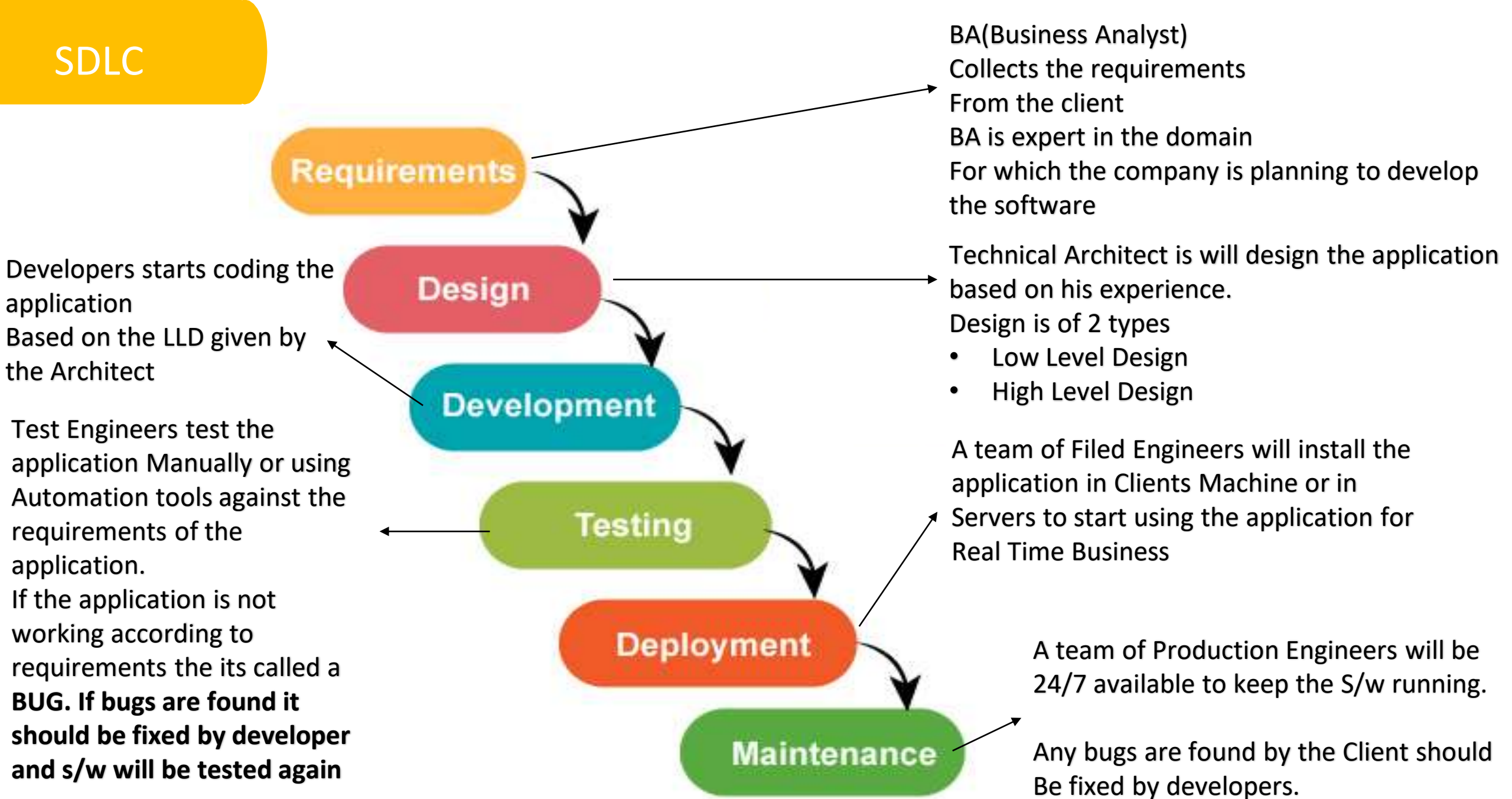
Software Development Introduction



BASAVANAGUD

- Software : A software is a group programs which is developed to solve a business problem.
- Program : It is a set of instructions which are developed to perform a particular task.

SDLC



Types of Software

- Standalone Apps
- Client/Server Apps
- Web Application

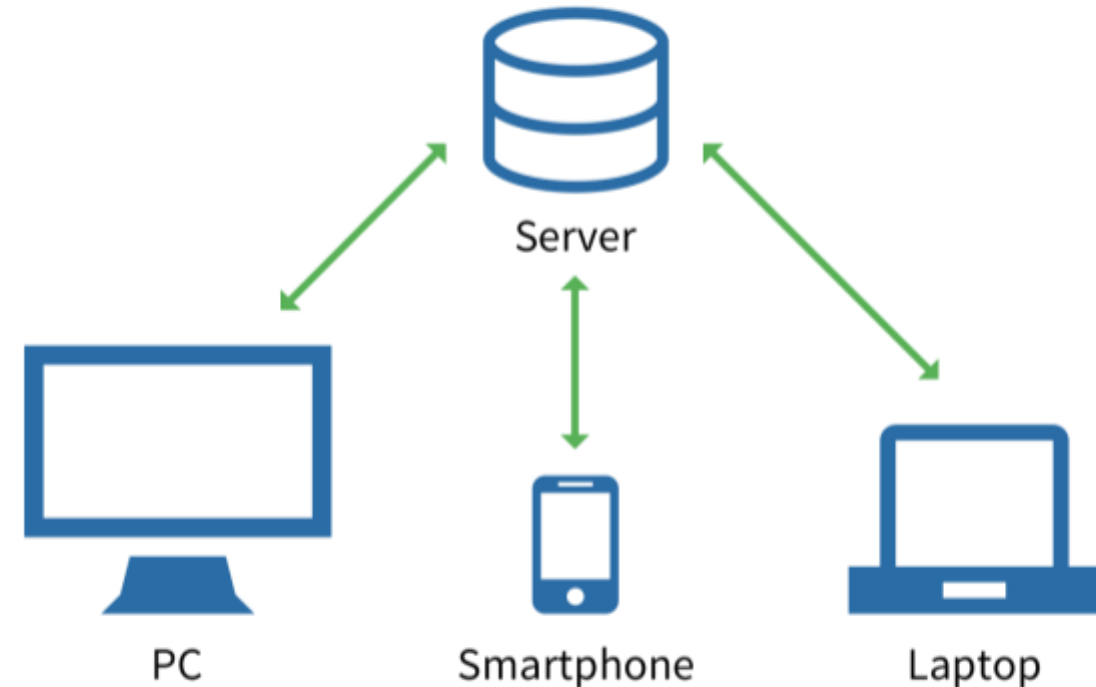
Standalone Apps

- Any application that runs without any network is called as standalone apps.
- Ex : Notepad, Calculator , Calendar etc..

Client/Server Apps

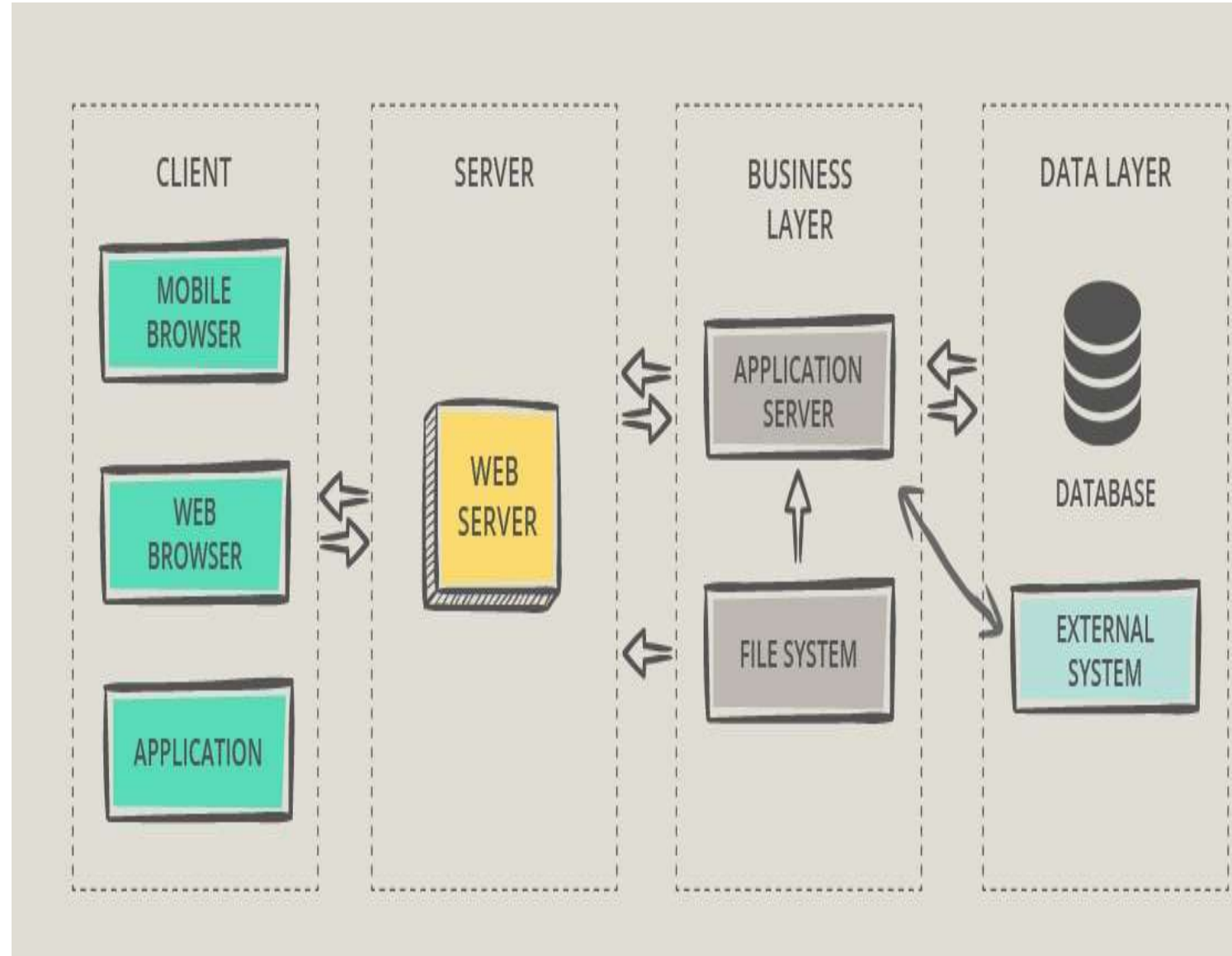
- Any application that requires network for its execution is called as client/server apps.
- The software developed to access the services of the Server is called as **Client**.
- Client is always installed in user machine(Computer/Mobile Devices)
- The software and hardware that receives the request from the client and process it to provide the service is called as **Server**
- Ex : whatsapp, messenger , skype

Client-Server Model



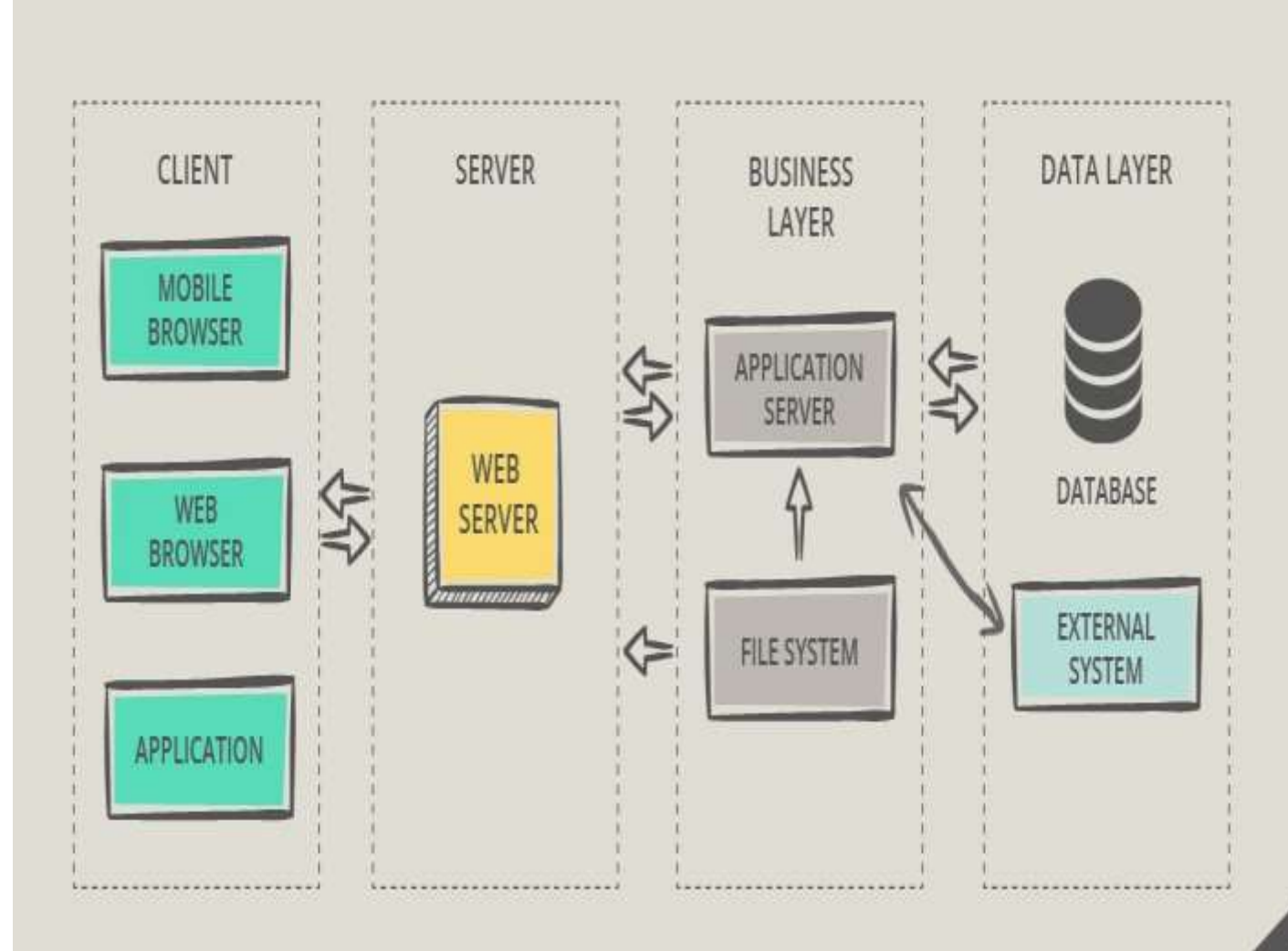
Web-Applications

- Any application that requires web browser for its execution is called as web application.
- Web App will have 3 layers.
 1. Presentation Layer
 2. Application Layer
 3. Data Layer



Web-Applications

- **Presentation layer** consists of web pages which is the front end of web application and it is developed by using HTML,CSS,JS etc..
- **Application layer** consists business logics which process the inputs from front end and provides necessary service by executing the corresponding programs in APP server.
- **Data layer** consists all the business data that is required to run the application such as User details, Account information, Settings data etc.



Duration : 2hrs

Trainer : Mr.Madhu Sundar

Subject : CORE JAVA

Program and Programming Language Fundamentals

The logo features a large, stylized orange letter 'J' followed by the word 'Spiders' in a bold, black, italicized sans-serif font.

Training & Development Center

BASAVANAGUDI

Program

- Program : It is a set of instructions which are developed to perform a particular task.
- Program Language : It is a language to communicate with a computer from a programmer.

Converts high level
language to Assembly level
language

```
a=10  
b=10  
c=a+b  
Print c
```

Compiler

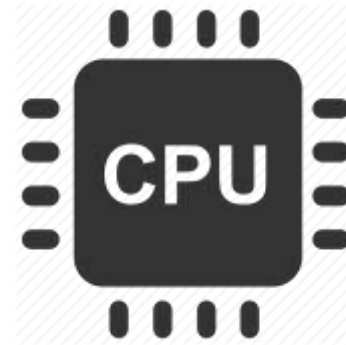
```
Take a  
Take b  
Store 10,a  
Store 10,b  
Take c  
Add a,b  
Store c  
Print c
```

Assembler

low level language or binary
language

0 1 0 1 0 1 0 1

ON/OFF



High level language
C, C++,java ,python ,JS, SQL

Assembly level language
8086 8085

Converts assembly level language to
low level language or binary language

Birth of Java

- Assembly level language created by the assembler can be executed only on the same computer which has same OS and CPU in which program was developed and this makes all S/W developed using this language dependent on the platform.
- Sun Microsystems a software development company had the vision of developing a programming language which was completely platform independent.
- In 1990 a team of Researchers led by **James Gosling** starts research to develop a platform independent programming language.
- In the year 1995 official version of JAVA was first released to the market



JDK(Java Development Kit)



Demo.java

Create java program using
Code Editors : Sublime
text, Notepad++, Editplus
etc..

IDE: NetBeans, Eclipse ,
IntelliJ , Jcreator

Javac
Java compiler

* Check the syntax of the
program.

if syntax mistake

throw Compile Time Error

else

**create convert java code to
bytecode and save it .class
file.**

Bytecode

Demo.class

JVM

Java Virtual
Machine

- Read a line of code.
- Understand a line of code
- Execute a line of code

if code is not
understandable

**throw Run Time Error
or Exception**

else

execute the code

Installation of JDK and Sublime Text



Learning objectives

- How to set-up JDK for java development.
- Learn how to compile and execute java programs
- Understanding structure of java program.

Keywords, Identifiers and Variables



Keywords

- Keywords are those reserved words which will have a pre-defined meaning defined in the programming language.
- Keyword should be written only in lowercase.

[https://docs.oracle.com/javase/tutorial/java/nutsandbolts/ keywords.html](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/keywords.html)

Identifiers

- Identifiers are the names given for an entity in a program
- In java identifiers are used to identify Class, Methods and Variables.

RULES OF IDENTIFIERS.

- ✓ Identifiers can be alpha-numeric.
- ✓ Identifiers **should not** start with numeric .
- ✓ \$ and _ are the only 2 special characters that can be used with Identifiers.
- ✓ Identifiers can start with \$ or _
- ✓ Single _ is an invalid identifier in java.
- ✓ Keywords **can not** be used as identifiers.

Variables

- A named memory location which is used to store values for the program is called as variable.

To use a variable we have to follow 3 steps

- ✓ Declaration
- ✓ Initialization
- ✓ Utilization

Variables

- Declaration is a statement which is written to specify the type of data to be stored in the given variable.

syntax : datatype varName;

DataType	Capacity	Default value
byte	8-bits or 1-byte	0
short	16-bits or 2-bytes	0
int	32-bits or 4-bytes	0
long	64-bits or 8-bytes	0l
float	32-bits or 4-bytes	0.0f
double	64-bits or 8-bytes	0.0
char	16-bits or 2-bytes	Space
boolean	8-bits or 1-byte	false

Initialization

- Initialization : It is a statement which is written to store the data within a variable using assignment operator(=).

syntax : varName = value;

Utilization

- Utilization : It is one or group of statements which is written to use the value in the variable to perform the operation.

Operators



Training & Development Center

BASAVANAGUDI

Operators

- Operator performs operations on operands and produce results.

Types of operators

- ✓ Arithmetic Operators
+ - * / %
- ✓ Increment and Decrement Operators
++ --
- ✓ Comparison Operators or Relational Operators
< > == != <= >=
- ✓ Logical Operators && || ^
- ✓ Bitwise Operators & | ^

Priority	Operator	Operation
1	() [] -> . ::	Function call, scope, array/member access
2	! ~ - + * & sizeof type cast ++ --	(most) unary operators, sizeof and type casts (right to left)
3	* / % MOD	Multiplication, division, modulo
4	+ -	Addition and subtraction
5	<< >>	Bitwise shift left and right
6	< <= > >=	Comparisons: less-than and greater-than
7	== !=	Comparisons: equal and not equal
8	&	Bitwise AND
9	^	Bitwise exclusive OR (XOR)
10		Bitwise inclusive (normal) OR
11	&&	Logical AND
12		Logical OR
13	? :	Conditional expression (ternary)
14	= += - = *= /= %= &= = ^= <<= >>=	Assignment operators (right to left)
15	,	Comma operator

Increment Operator

- Increment operator will increase the value of a variable by one unit;
- Pre-Increment Post-Increment
 - *increment *substitute
 - *substitute *operation
 - *operation *increment
- Both Pre-Increment and Post-Increment Operators will show same results if they are used without any expressions or java statements.
- Whenever the increment Operator is used with a character variable the unicode value of the character will be increased and its corresponding character will be stored in the given variable.
- Boolean variables cant be used with increment operator.
- Values cant be directly used with increment operator

Comparison Operators

- Comparison operators will compare given 2 values and returns a Boolean result

Logical Operators

- Logical operators are used to combine multiple Boolean conditions as ONE single condition.
- Logical operators will always return a Boolean result
- Different types of logical operators are
 - Logical AND (&&)
 - Logical OR(||)
 - Logical XOR(^)

Logical AND &&

- Logical AND && operators will return TRUE if and only if the result of both the Boolean condition's result is TRUE and in all other cases it returns FALSE
- TRUTH table for LOGICAL AND (&&)

CONDITION 1	CONDITION 2	RESULT
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

- Logical OR || operator will return FALSE if and only if the result of both the Boolean condition's result is FALSE and in all other cases it returns TRUE
- TRUTH table for LOGICAL OR (||)

CONDITION 1	CONDITION 2	RESULT
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Logical XOR ^

- Logical XOR ^ operator will return FALSE if and only if the result of both the Boolean condition's result is FALSE or TRUE and in all other cases it returns TRUE
- TRUTH table for Logical XOR ^

CONDITION 1	CONDITION 2	RESULT
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Bitwise Operator

- Bitwise operator will perform bitwise operations on every bit of given value and produce results as 0 or 1

25

2|12-1

2|6-0

2|3-0

2|1-1

Result : 11001

0011001

1100100(&)

0000000

$0*2^6 + 0*2^5 +$

$0*2^4 + 0*2^3 +$

$0*2^2 + 0*2^1 +$

$0*2^0 =$

RESULT : 0

0011001

1100100(|)

1111101

0011001

1100100(^)

1111101

100

2|50-0

2|25-0

2|12-1

2|6-0

2|3-0

2|1-1

Result : 1100100

Concatenation Operator

- concatenation oprt(+) : Is used to join a String value with any other data.
- Between a string and any other type if you use + oprt it will join the string and given value creating a new String.
- Ex : "hello"+"world"
- "hello"+10

Methods/ Functions



methods

- Methods are used to eliminate duplicate lines of code in the program.
- Methods are named blocks of codes which will perform a specific task.

- Syntax : **access** **access** **return** **name(argument list)**

```
    specifier modifiers type  
    {  
        stmt..  
        stmt..  
        return;  
    }
```

- Access modifier : static.
- Access specifier : public, protected, pkg-level, private.
- Return type : depends on the data type of the value returned from the method

- If methods returns any value then the returned value should be stored in a variable matching the return type of the method.
- A method which is calling another method is known as calling method.
- A method which is called by another method is known as called method.
- return statement is used to transfer the control and the values from called method to calling method.

- If methods do not return any value then its return type should be declared as **void**.
- If return type of method is void then, programmer can skip writing the return statement.

FLOW CONTROL STATEMENTS



Flow control statements

- Flow control statements are used to control the execution flow of the program at the runtime.
- Flow control statements are of 2 types
 - Decision Making Statements
 - Looping Statements

Decision Making Statements

- Decision Making Statements are used to execute one or group of statements based on a Boolean condition.
- Boolean cond / Boolean exp : a Comparison statement written using Comparison opts and logical opts is called Boolean exp/cond.
- Types of Decision making statements are
 - IF
 - If-else
 - if-else-if
 - Switch-case

if - statement

- it is a type of decision making where one or group of statements written within the if body will be executed if and only if result of Boolean condition is true .

Syntax : if(Boolean cond)
 {
 stmt..
 stmt..
 }

if - else statement

- It is a type of decision making statements where statements of if body are executed if result of boolean condition is TRUE else the statements of else body are executed .

Syntax : if(Boolean cond)

```
{  
    stmt..  
    stmt.  
}  
else  
{  
    stmt  
    stmt  
}
```


if – else - if statement

- It is a type of decision making statements which is used to whenever there are multiple Boolean conditions to be evaluated to execute different set of statements.

Syntax : if(Boolean cond1)

```
{  
    stmt..  
    stmt.
```

```
}
```

else if(Boolean cond2)

```
{
```

```
    stmt
```

```
    stmt
```

```
}
```

else

```
{
```

```
    stmt
```

```
    stmt
```

```
}
```

switch case - statement

- It is a type of decision making statements which is used whenever there are only **equals conditions** to be evaluated to execute different set of statements.

Syntax : switch(choice)
{
 case choice 1 : stmts
 break;
 case choice 2 : stmts
 break
 default : stmts
}

Looping Statements

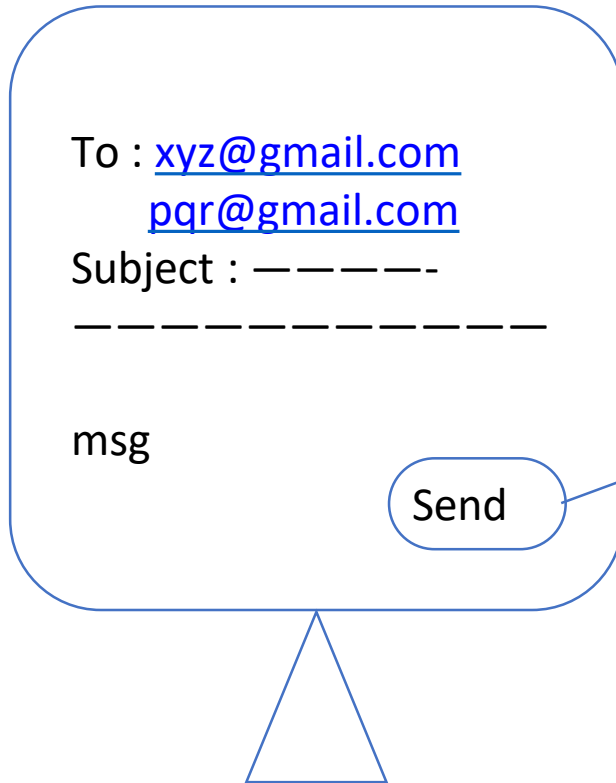
- Looping Statements are used to execute one or multiple Statements repeatedly.
- In java there are 4 looping statements
 - For loop
 - While loop
 - do-while loop
 - for-each loop

Looping Statements

To : xyz@gmail.com
pqr@gmail.com
Subject : -----

msg

Send

A diagram of a computer monitor. The screen displays an email form with fields for 'To', 'Subject', and 'msg'. The 'To' field contains two email addresses: 'xyz@gmail.com' and 'pqr@gmail.com'. The 'Subject' field is followed by a dashed line. The 'msg' field is empty. A 'Send' button is located at the bottom right of the form. A blue line connects the 'Send' button to the code block on the right.

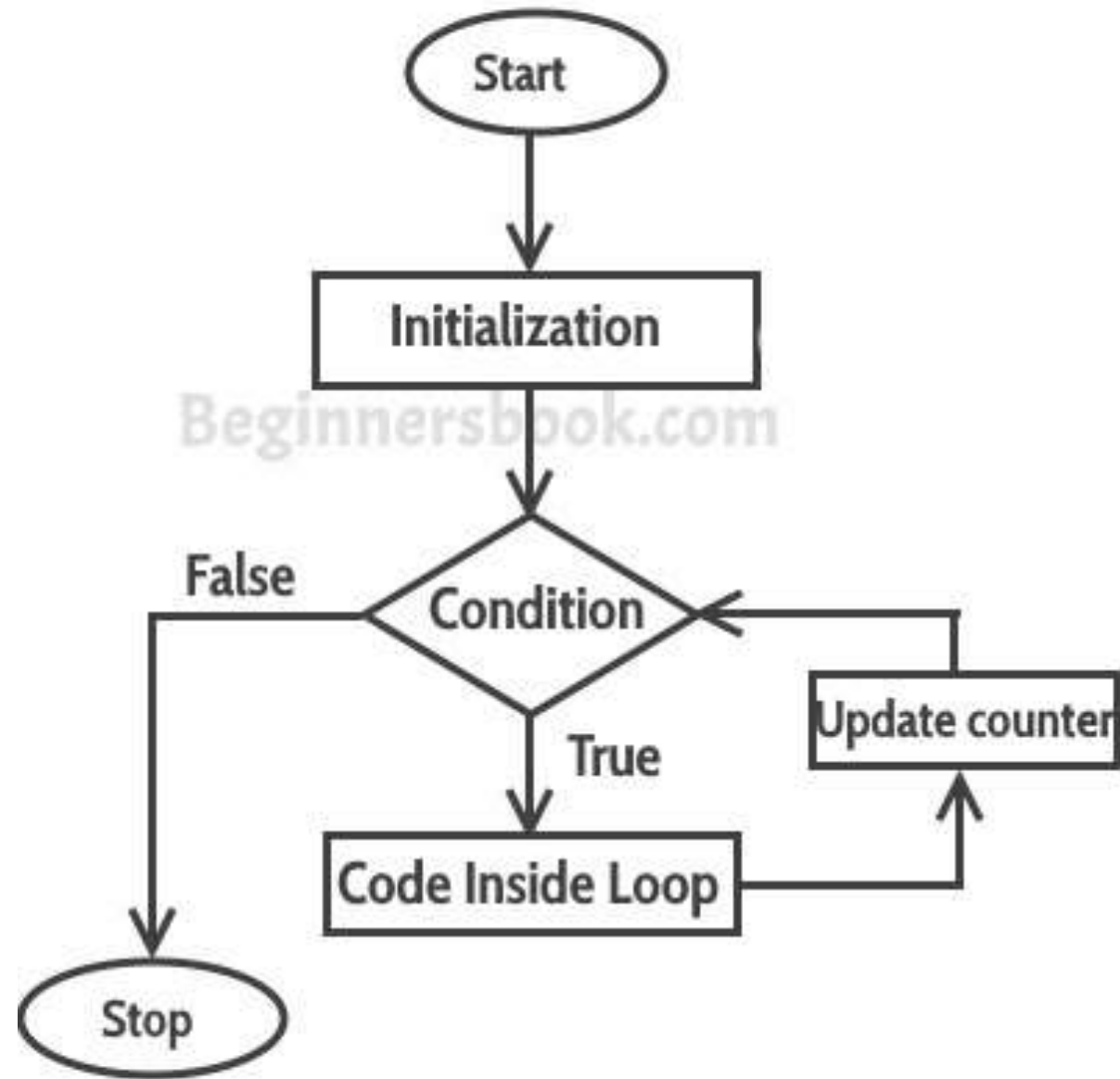
```
class Email
{
    sendmail(int num)
    {
        loop(num) num = no. of To email ids
        {
            --
            --
        }
    }
}
```

For loop

- for-loop is used whenever logical start and logical end is very well defined.

- syntax :

```
        start      check did you reach the end
for(initialization; stop_condition; counter)
{
    stmts..
    stmts...
}
```

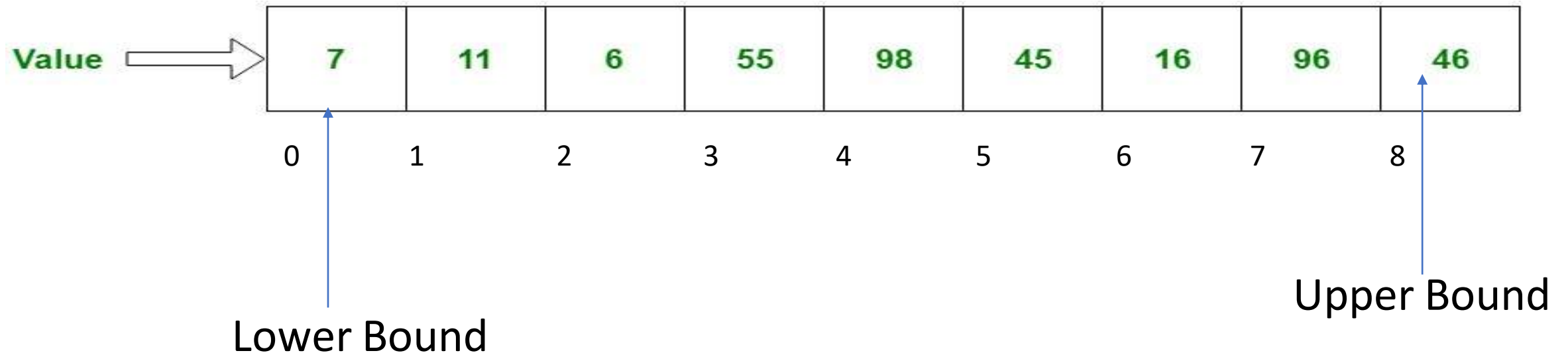


ARRAYS



Arrays

- Array is a group of same type of data which has a fixed length and index to identify every bucket uniquely.



Array Length = 9

Arrays

- Array Declaration
- syntax : `datatype[] arrayVarName;`
ex : `int[] arr;`
- syntax : `datatype arrayVarName[];`
ex : `int arr[];`
[] -> subscript
- Array Creation
- syntax : `arrayVarName = new datatype[size];`
- ex : `arr = new int[10];`

- Array Declaration and Creation

- syntax : datatype[] arrayVarName = new datatype[size];

- ex : int[] arr = new int[10];

- Once the array is created all the buckets will be filled with default values depending on the data type of the array.

- length is an in-built variable which contains the count of no. of buckets in the given array.

- Array Declaration and Creation

- syntax : datatype[] arrayVarName = {v1,v2,v3...};

- ex : int[] arr = {20,40,12};

Arrays

- first index $\rightarrow 0$ (lower bound)
- last index $\rightarrow \text{arr.length} - 1 = 9 - 1 = 8$ (upper bound)
- mid index $\rightarrow (\text{arr.length} - 1) / 2 = (9 - 1) / 2 = 4$ (middle bound)

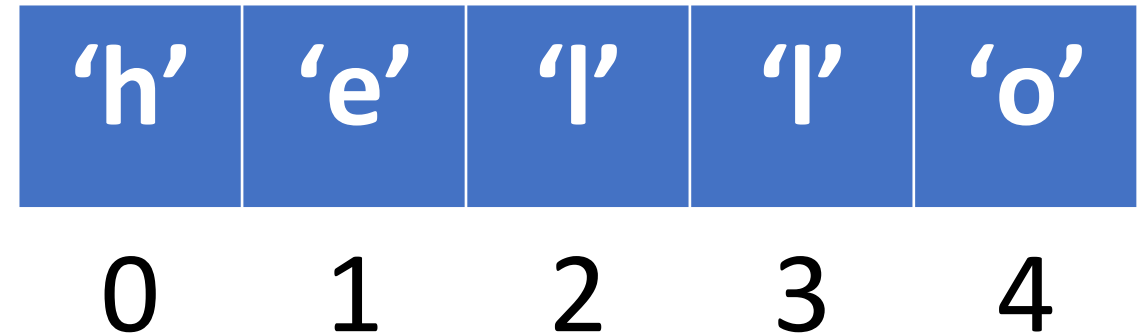
Strings



Strings

- String value is a group of characters which is written in the double quotes.
- We can access the characters or perform any operations on the string only by using the methods of String.
- Internally String value is stored as a character Array by the JVM.

String str = "hello"



Methods of String

- `length()` : It returns the count of number of characters present in the given String.
- `charAt(index)` : It returns the character at the given index from the given String.
- `indexOf(char)` : It returns the index of the given character from the given String. It returns -ve value if the character do not exist.
- `indexOf(char, startIndex)` : It returns the index of the given character from the given String by starting the search from the given `startIndex`.
- `equals(string)` : It compares characters of given two strings and returns true if they are exactly same else it returns false.

Methods of String

- `equalsIgnoreCase(string)` : It compares characters of given two strings by ignoring the case and returns true if they are exactly same else it returns false.
- `substring(start, end-1)` : It creates a String from the given start index and given end-1 index no. of characters and returns the String.

- **NOTE**

- `length` and `length()`
- `length` -> variable used in array
- `length()` -> method of String

Object Oriented Programming System



Training & Development Center
BASAVANAGUDI

Section 2 Object Oriented Programming Syllabus

- Classes and Objects

- - Object creation
 - - Reference variable
 - - Global and local variables
 - - Constructors
 - - Composition
 - - Inheritance *****
 - - Method Overloading
 - - Method Overriding
 - - Abstract classes
 - - Interfaces
 - - Typecasting
 - - Polymorphism
- Abstraction
 - Java packages
 - Access Specifiers
 - Encapsulation

CLASSES and Objects



Class

- Class is blueprint of an object.
- A class will have 2 types of members
 1. Data members
 2. Function members
- Data members are those variables which are declared within the body of the class.
- Function members are those methods/functions which are declared within the body of the class.
- Members of the class are of 2 types
 1. static
 2. non-static

Static Members

- Any data member or function member of the class declared by using static keyword is called as STATIC MEMBER.

Present in different class

- Static member present in different class can be accessed by using class name with member name.
Ex : **ClassName**.memberName

Present in same class

- Static member present in same class can be accessed just by using member name.

Non-Static

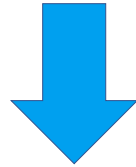
- Any data member or function member of the class declared by **without** using static keyword is called as NON-STATIC MEMBER.

Present in different class

- Non-Static member present in a different class can be accessed within any method of a different class only by creating the object of the class.

- Object of a class can be created by using following syntax :

new ClassName();



- Creates a new object in the memory of the given class and returns its address
- Constructor Call
Copy all the non-static members of the class to the new object created

Present in same class

- Non-Static member present in a class can be accessed within a **static method** only by creating the object of the class.
- Non-Static member present in a class can be accessed within a non-static method directly by using membername.

ACCESS MODIFIER LOCATION	STATIC	NON- STATIC
SAME CLASS	Just by using the member name	Static method : only by creating the object of the class. Non-Static method : Just by using the member name.
DIFFERENT CLASS	Using classname with the member name. ClassName.membername	only by creating the object of the class

Static Members

- Static members will have only one copy in the memory.
- If the value of a data member in a class is not changing from object other object then those data members should be declared as **static**.
- If a function member / method in a class is using only static members of the class then those function member / method should be declared as **static**.

NON-STATIC MEMBERS

- Non-Static members will have multiple copy in the memory.
- If the value of a data member in a class is changing from object other object then those data members should be declared as non-static.
- If a function member / method in a class is using at least one non-static members of the class then those function member / method should be declared as non-static.

Reference Variables

- It is a type of variable which is used to store address of an object.
- Within a reference variables we can't store primitive values.
- Within a primitive variables we can't store store address of an object.
- Within one reference variables we can store ONLY ONE OBJECT ADDRESS.
- Multiple reference variables can point to same object.
- If Multiple reference variables points to same object then, changes done on the data of the object by one reference variable will impact other reference variables.
- If two reference variables points to different objects then, changes done on the data of one object by one reference variable will NOT impact other reference variables.

Global variables and Local variables



Global variables

- Any variable which is declared within the body of the class is called Global variable.
- Global variables can be accessed by any method present in same class.
- Global variables will be initialized by the compiler with the default values if the programmer do not initialize them.
- Global variables are nothing but DATA MEMEBERS of the class.

Local variables

- Any variable which is declared with the method declaration and within the method definition is called as local variable.
- Local variables can be accessed only by the method in which it is declared.
- Local variables should be initialized by the programmer with the values explicitly.

Constructors



Training & Development Center

BASAVANAGUDI

Constructors

- Constructors are special type of methods which have same name as the class name..
- Constructors are executed whenever the object of that corresponding class is created.
- Every class must and should have Constructor.
- If the programmer do not write any constructor, then compiler will write default constructor implicitly.
- If the programmer writes constructor it can be of two type
 1. Zero argument constructor
 2. Parameterized constructor
- Constructors cannot be declared as static or final.
- If you specify return type for the constructor, then it will be considered as a normal method.
- If the programmer writes any constructor explicitly, then compiler will not write any constructor implicitly.

App.

- Constructors are used to Initialize the data members of the class.

Constructors

- This keyword
- this keyword is used to differentiate between local variables and Global variables within a method / constructor whenever they have same names.

Relations between classes or Objects



Class diagrams

- A Class diagram is pictorial representation of class.
- Class diagrams helps understanding the relations between classes in much better way.
- 2 classes are related based on the dependency of class object over the another class object.
- 2 classes are related based by two types relations
 1. Has-A relation
 2. Is-A relation(inheritance)
- **Has-A relation**
 - Has-A relation is a type of relation where 2 classes are related based on the dependency of existence of one class object over the another class object.
 - Has-A relation is implemented by creating GLOBAL STATIC REFERENCE variable pointing to the object of another class.

Inheritance



Training & Development Center

BASAVANAGUDI

Inheritance

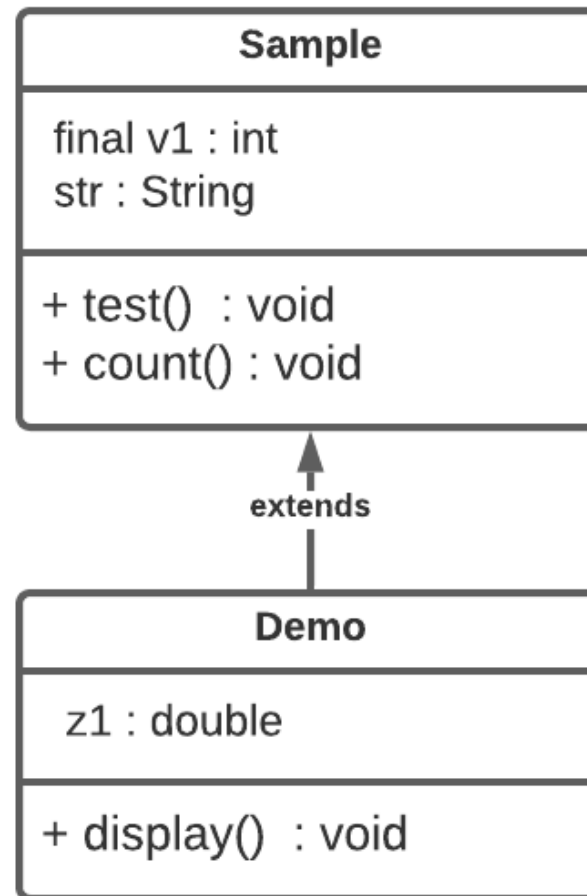
- One class acquiring the properties of another class is called inheritance.
- The class from where properties are inherited is called as superclass or base class or parent class.
- The class which is inheriting properties is called as sub class or derived class or child class.
- In java inheritance is achieved by using extends keyword.
- Using superclass object we can access ONLY properties of superclass
- Using subclass object we can access properties of both subclass and superclass
- final classes CANNOT be inherited.
- final data members and function members of the class can be inherited.
- constructors of the class CANNOT be inherited.
- private data members and function members of the class CANNOT be inherited.
- Using subclass object we can access both static and non-static properties of both subclass and superclass

- Different types of inheritance are
 1. Single inheritance
 2. Multilevel inheritance
 3. Multiple inheritance
 4. Hierarchical inheritance
 5. Hybrid inheritance

Single inheritance

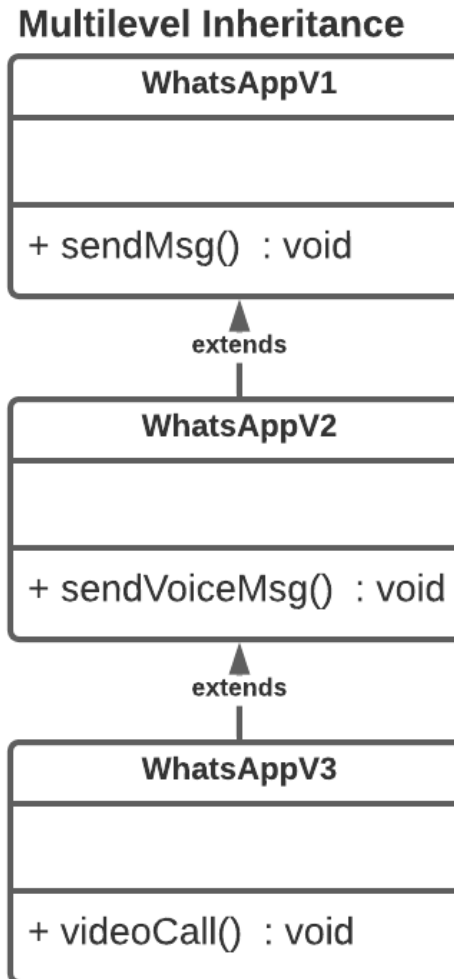
- one Subclass inheriting from one Single Superclass is called as Single Inheritance.

Single inheritance



Multilevel Inheritance

- Subclass inheriting the properties of superclass and that superclass inheriting the properties from another superclass is called as Multilevel inheritance.

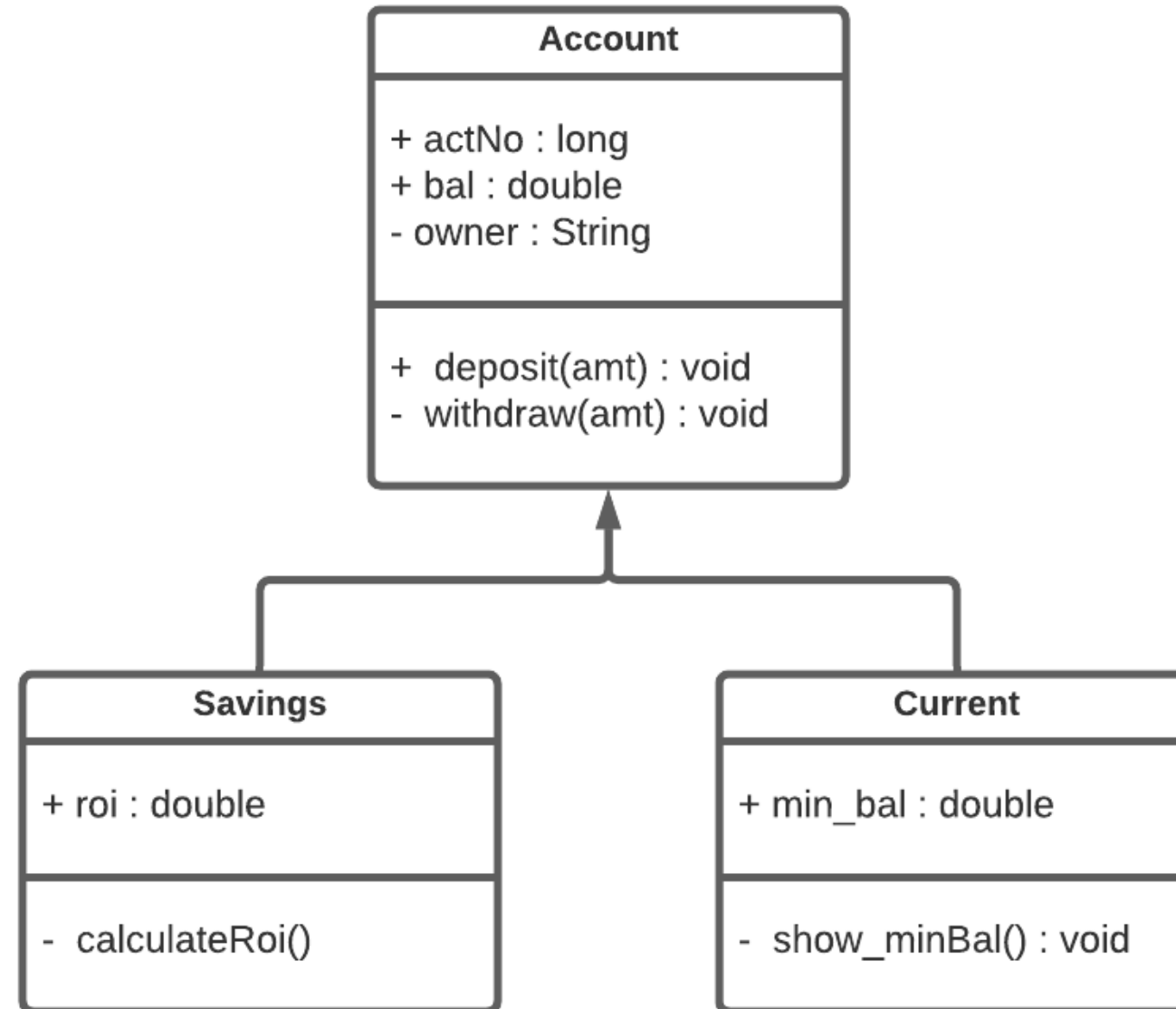


Multiple inheritance

- Subclass inheriting the properties from 2 or more superclass is called as Multiple inheritance.
- Java don't support Multiple inheritance.

Hierarchical inheritance

One superclass inherited/extended by two or more subclasses is called as Hierarchical inheritance.



Generalization

- Declaring common methods and variables of all subclasses in one common superclass.

Specialization

- Declaring methods and variables specifically for one subclass.

Hybrid Inheritance

- It is the combination of Different types inheritance.

- `super();` (super statement)
- `super();` creates the object of superclass whenever object of subclass is created to inherit the properties.
- `super();` can be written explicitly by the programmer or implicitly by the compiler at the compile time.
- If the superclass contains only parameterized constructors then the programmer should write `super();` explicitly and pass the required arguments.
- `super();` must be written ONLY within the constructor body.
- Always superclass object should be created first and then properties should be inherited to subclass object and hence `super();` should be always written at the first line of constructor body.
- Multiple `super();` within the same constructor body is NOT allowed. because writing Multiple `super();` may lead to creation of multiple superclass objects which is not required.

- Object class is the super Most class in java.
- Each and every class directly or indirectly inherits from object class .
- **Why java do not support Multiple Inheritance?**OR Explain Diamond problem in java?**
- It creates an ambiguity/confusion for the complier to choose the path from where properties of object class should be copied to subclass.
- To call multiple Constructors of multiple Super classes we should write multiple `super();` which is not supported in java.
- If both super classes have methods with same name and same arguments, then it creates an ambiguity for the complier to choose which method should be called for the execution when it called using subclass object.

❖ **Constructor chaining :**

- It is the process of subclass constructor calling its superclass constructor and superclass constructor calling object class constructor.

Method Overloading



Training & Development Center

BASAVANAGUDI

- Developing multiple methods within the same class with the same name which may differ in
 1. Number of arguments
 2. Data type of arguments
 3. Order of argumentsis called as Method Overloading.
- Ex: println() is the best example for Method Overloading.
- **Advantage:** It is easy to remember One method which may perform similar operation with different arguments.
- **Disadvantage:** If proper documentation are not maintained for every overloaded method it will be difficult for the new developer to understand behavior of every method.

Constructor Overloading



Training & Development Center

BASAVANAGUDI

- Developing multiple constructors within the same class with the same name which differ in
 1. No. of arguments or
 2. data type of arguments or
 3. order of argumentsis constructor overloading

Method Overriding



Training & Development Center

BASAVANAGUDI

- Subclass inheriting method of superclass and changing the method definition according subclass specification without changing method declaration is called as method Overriding.
- Inheritance is mandatory for method Overriding.
- @ -> annotation
- @Override (Override annotation) it compares the given method declaration of subclass with every other method declaration present in superclass and throws an error if there is no matching declaration found.
- Final, Private and Static methods cannot be Overridden.
- If Subclass and Superclass contains STATIC methods with same name and same arguments then it is called as Method Hiding.

❖ **super keyword**

- super keyword is special in-built reference variable which is created by JVM and it will point to immediate Parent class object.
- super keyword is used within the subclass methods in order to access properties of its immediate superclass.

Abstract class and Abstract methods



Abstract

- incomplete
- A method which has only method declaration and no method definition is called as Abstract method.
- Abstract method must be declared by using abstract keyword.
- If a class contains at least one abstract method, then the class must be declared as abstract class.
- It is impossible to create object of an Abstract class.
- We can create reference variables of abstract class type.
- A method which has both method declaration and method definition is called as concrete method.
- If a class contains only concrete methods, then the class is called as concrete class.
- If a class inherits from an abstract class then the subclass must override all the abstract methods present in super class else the class must be declared as abstract.

Abstract

- abstract methods cannot be declared as
 - ☐ final
 - ☐ static
 - ☐ private
- within an abstract class we can write
 1. only abstract methods or
 2. only concrete methods or
 3. both abstract and concrete methods
- Non-static non-private members of abstract class can be accessed by creating the object of its subclass only.
- Static non-private members of abstract class can be accessed by using the class name.
- If Generalized methods of all the subclasses have same definitions then those methods should declared as concrete methods in Generalized class.
- If Generalized methods of all the subclasses have different definitions then those methods should declared as abstract methods in Generalized class.

- **Application of Abstract class:**
- By using abstract classes we can ensure there are no missing method implementations the subclass.
- Abstract acts a blueprint to create concrete class.
- final class cannot be inherited
 - final methods can be inherited but cannot be overridden
 - final variables can be inherited but cannot be re-intialized

Interface



Training & Development Center

BASAVANAGUDI

Interface

- Interface is a type where methods are by default abstract and variables are by default static and final.
- It is impossible to create object of an interface.
- We can create reference variables of interface type.
- The class can inherit the properties of an Interface only by using Implements keyword.
- The interface from where properties are inherited is called as Super Interface
- The class which implements the interface is called as Implementation Class
- If a class implements an interface then the implementation class must override all the abstract methods present in super Interface else the class must be declared as abstract class.
- An interface cannot implements another interface.
- An interface should only extends another interface.
- An interface can NEVER inherit or extends from a class not even from Object class

- A class can implements any number of interfaces.
- A class can extends and implements at the same time.
- A class can extends ONLY one Superclass but it can implements any number of interfaces at the same time.
- **Application**
- If we have to develop a class using Two Super Types then we should use interfaces.
- **Note:** Abstract class cannot be used here because, a class cannot extends from Two Different super classes at the same time.
- From JDK 8 interfaces now on supports concrete methods.
- concrete methods of interface should be declared only by using 2 keywords
 1. static - for static methods
 2. default - for non-static methods
- if you want a concrete method within the interface which should NOT be overridden in implementation class then method shall be declared as static method.

Interface

- if you want a concrete method within the interface which you want to be overridden in implementation classes then method shall be declared as default method.
- static methods of interface can be accessed by using interface name with dot(.) operator.
- default methods of interface can be accessed by using Object of its implementation class ONLY.
- If two interfaces contains static methods with the same name and same arguments then , we differentiate them with help of Interface Name.
- If two interfaces contains default methods with the same name and same arguments then , it should be overridden in the implementation class.

TypeCasting



Training & Development Center

BASAVANAGUDI

TypeCasting

- Converting one Type to another Type is called is TypeCasting.
- TypeCasting is of 2 types
 - Primitive Casting
 - Derived Casting
- Converting one Primitive data type value to another Primitive data type value is called as Primitive Casting.
- Primitive Casting is of 2 types
 - Widening
 - Narrowing
- Widening : Converting lower data type value to Higher Data type value
- Widening is done implicitly by the compiler.
- Narrowing : Converting higher data type value to lower Data type value.
- Narrowing should be done explicitly by the programmer by writting
- casting statement.
Syntax : (datatype) val/var;

- If a method is having Primitive data type argument then for the same method we can pass values which are lower data type compared to given method argument
- If there are 2 overloaded methods one with lower data type argument and other with higher data type argument, if you pass a lower data type value and call the method , compiler will always choose method with lower data type argument
- Even though int and float have same capacity the data is represented in a different format.
- Compared to float and int , float is higher data type and int is lower data type
- Even though long and double have same capacity the data is represented in a different format.
- Compared to double and long , double is higher data type and long is lower data type.
- If you store a character value within an integer variable then its unicode value will be stored in the given integer variable.

TypeCasting

- Converting one object reference type to another object reference type is called as Derived casting.
- Derived casting is of 2 types
 1. Upcasting
 2. Downcasting
- Converting subclass object reference to superclass object reference is called Upcasting
- Upcasting is achieved by storing the address of subclass object into superclass reference variable
- Any superclass reference variable pointing to subclass object is called as Upcasted reference.
- Using an Upcasted reference we can access only properties of superclass in the subclass object and it is impossible to access subclass properties.
- Converting Upcasted object reference back to subclass object reference is called Downcasting.
- Using an Downcasted reference we can access properties of superclass and subclass.

- NOTE : INHERITANCE IS MANDATORY FOR DERIVED CASTING
- **what is ClassCastException?**
- Whenever we try downcast an upcasted reference to a different class reference which do not contains the properties of given object, then jvm will throw ClassCastException
- If a method is having Superclass reference argument then for the same method we can pass all of its Subclass references compared to given method argument
- If there are 2 overloaded methods one with Super class reference argument and other with Subclass reference argument, if you pass Subclass reference and call the method , compiler will always choose method with Subclass reference argument
- **instanceof** operator compares given object reference with the given class returns true if it contains the given class properties else it returns false..
- ***** If you want to access Parent class properties which is common to all Sub-class then, there is no need of performing any downcasting.**

Polymorphism



Training & Development Center

BASAVANAGUDI

Polymorphism

- One entity showing different behaviors at different places is called Polymorphism
- Polymorphism is of two types
 1. Compile time Polymorphism
 2. Runtime Polymorphism

Compile Time Polymorphism

- Binding the method declaration to method definition by the compiler at the compile time based on the arguments is called as compile time Polymorphism.
- Since the binding is done before the execution it is called as early binding.
- Once the binding is done it cannot be changed at the Runtime and hence it is also called as Static Binding
- Method Overloading is the best example for Compile time Polymorphism

- Binding the method declaration to method definition by the JVM at the run time based on the objects is called as run time Polymorphism
- Since the binding is done during the execution it is called as late binding
- Once the binding is done it can be changed at the Runtime and hence it is also called as dynamic Binding.
- Method Overriding is the best example for Run time Polymorphism
- **Golden java rule**
- ***** Using an upcasted reference if you call an overridden method then you always get the implementation of subclass.

Abstraction



Training & Development Center

BASAVANAGUDI

Abstraction

- Abstraction - Hiding
- Abstract - incomplete
- Hiding the implementation details of a class and exposing only the services or behaviours is called Abstraction
- Abstraction is achieved by creating 3 layers
 - - Object Implementation layer
 - - Object creation layer
 - - Object utilization layer
- Generalize all the subclass methods and declare them as abstract methods in a common Super Class or Super Interface and create Object Implementation layer
- Create Object creation layer by Creating a class that will create the objects of implementation class and upcast it to Super Class or Super Interface reference.
- Within the Object utilization layer utilize upcasted reference returned by Object creation layer to call the Generalized methods.

Abstraction

- By using Abstraction we can achieve loose coupling where changes done in object implementation layer will not have any impact on object utilization layer

Java Packages

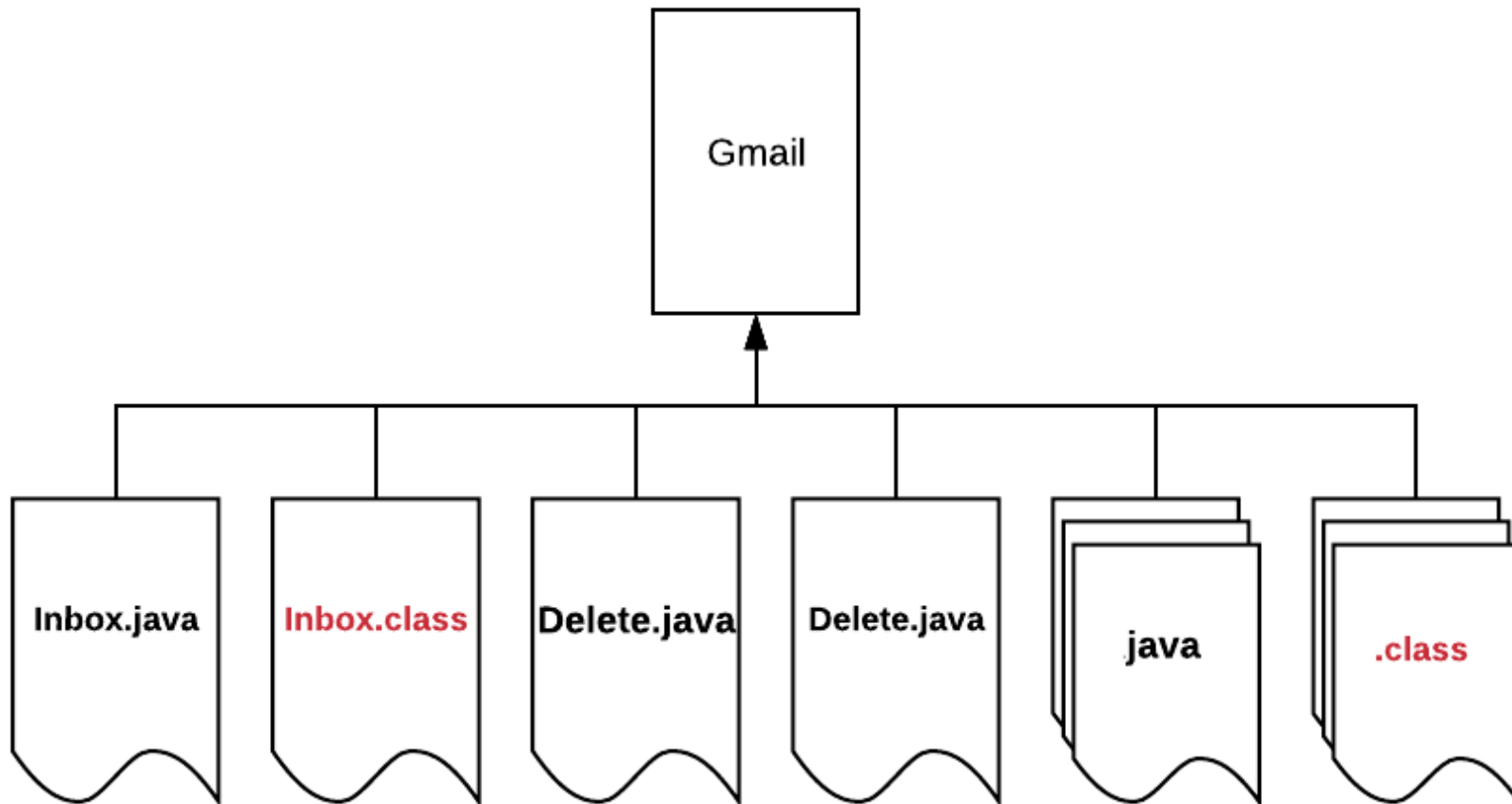


Training & Development Center

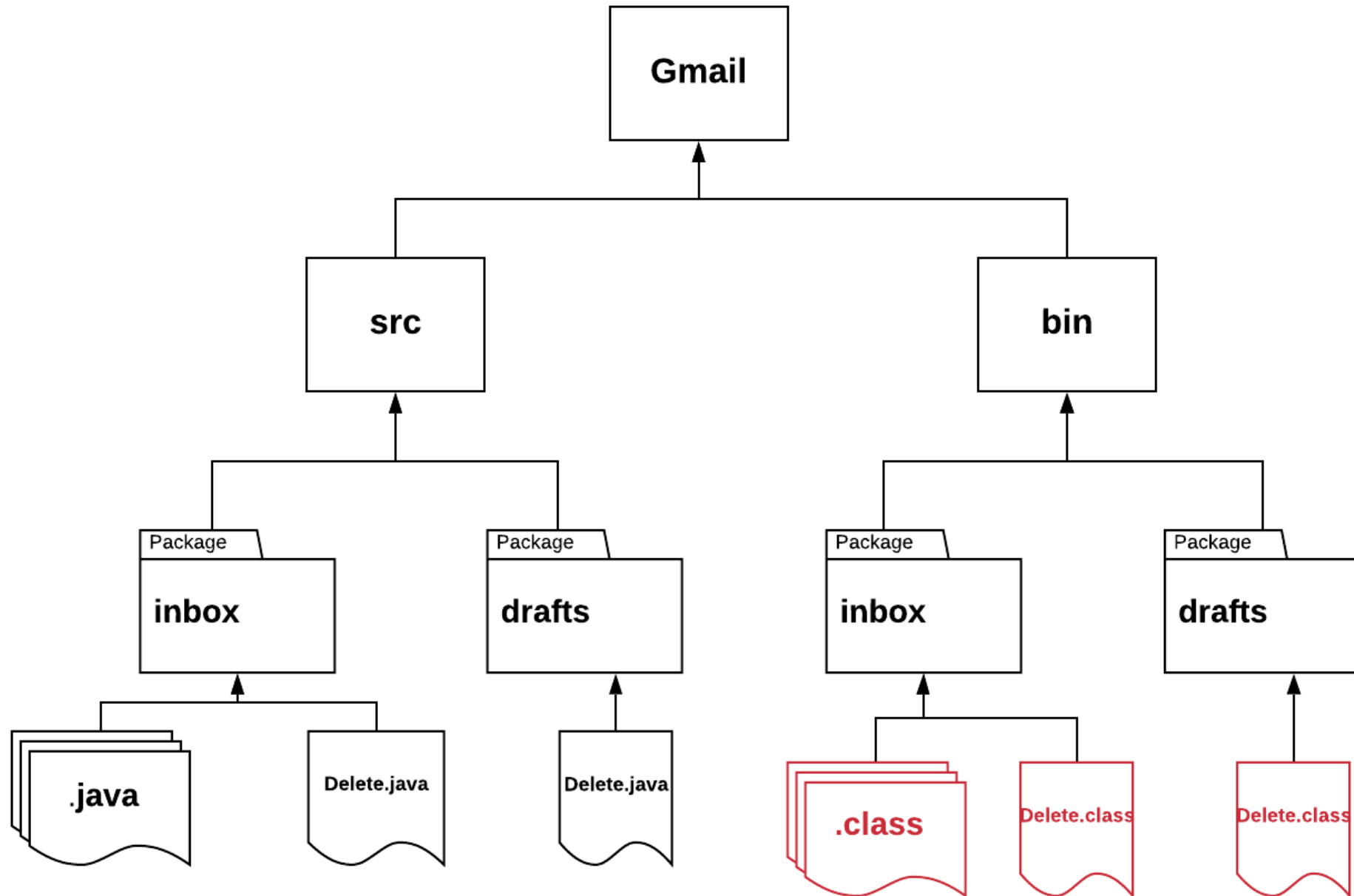
BASAVANAGUDI

Packages

- Managing Source code is difficult
- Creates Naming collisions
- Security for classes and its members is not guaranteed



Packages



Packages

- A java package is group of classes and interfaces which are related to one single module in the given project.

Package naming convention

- package name is always written in all-lowercase.
- package names are always written in reverse order of domain

convention 1

domain.appname.modulename
com.gmail.inbox

convention 2

domain.companyname.appname.modulename
com.google.gmail.inbox

Packages

- Data members and function members of a class present in different package can be accessed by two ways
 1. Using fully qualified class names
 2. Using import statement
- A class name which is written with its package name is called as fully qualified class name.

Access Specifier

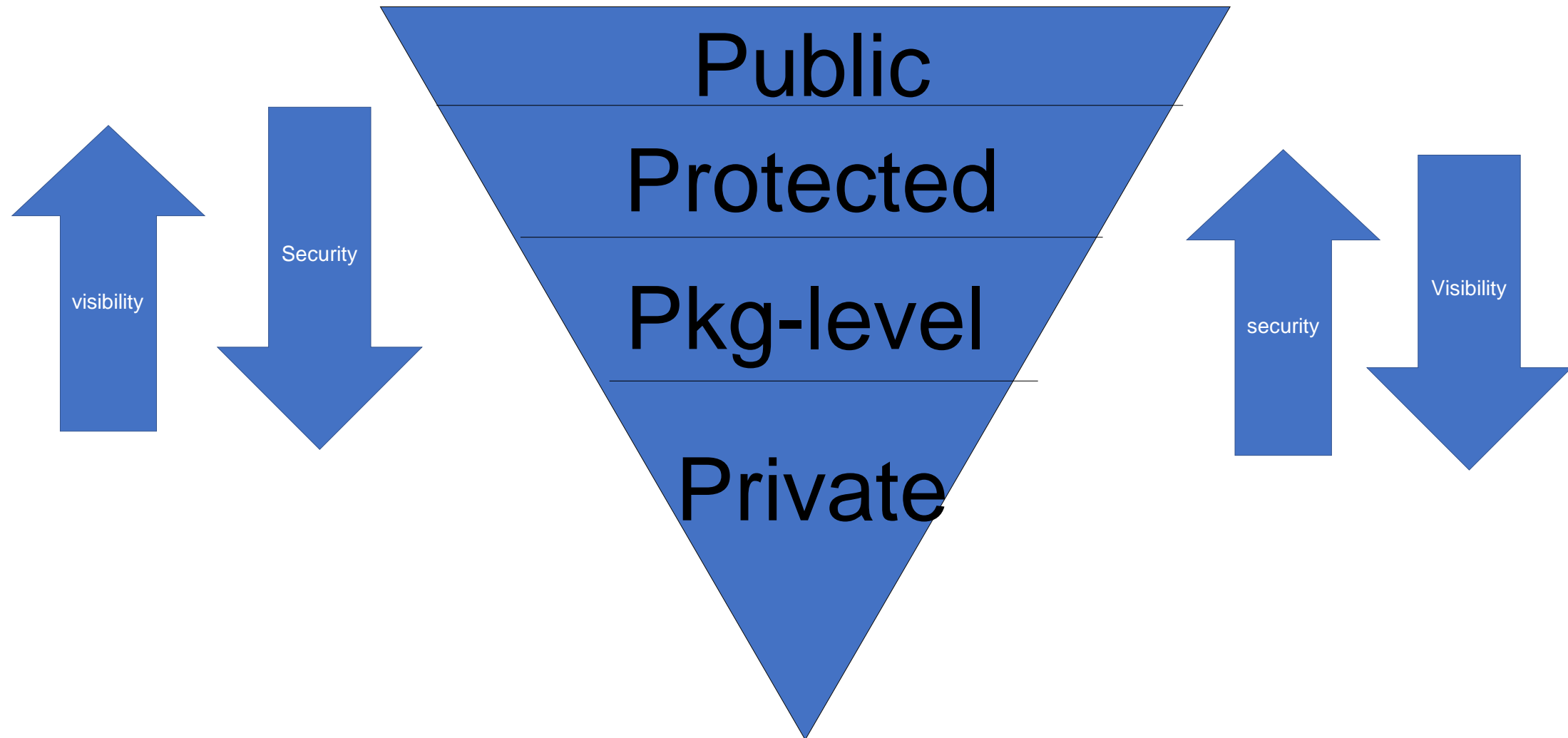


Training & Development Center

BASAVANAGUDI

Access Specifier

- Access specifiers are used to provide security for the classes and its members by controlling the visibility.



Access Specifier

public :

- if you declare any entity as public, then it can be accessed by the classes present in same or different package.
- public entities will have highest visibility and lowest security.

protected :

- If you declare any entity as protected, then it can be accessed by the classes present in same package.
- protected entity can be accessed by other class present in different package through inheritance and creating the object of **SUBCLASS ONLY**.

pkg-level(default) :

- if you declare any entity without using any access specifier keyword then it is considered as pkg-level member(default member).
- if you declare any entity as pkg-level(default), then it can be STRICTLY accessed ONLY by the classes present in SAME package.

private :

- if you declare any entity as private, then it can be accessed ONLY by the class within which they are declared.
- private entities will have highest security and lowest visibility.

Encapsulation



Training & Development Center

BASAVANAGUDI

Encapsulation

- Declaring data members as private and grant the access through getter and setter methods is called encapsulation
- getter method returns the current value of the data member
- return type of getter method depends on the data type of the data member who's value will be returned
- setter method updates the value of the data member.
- return type of setter method will be usually void because it doesn't return any value.
- Encapsulation is used to provide security for data members against the invalid values.

SECTION 3



Training & Development Center

BASAVANAGUDI

Java Libraries



Training & Development Center

BASAVANAGUDI

- * `java.lang`

- * It contains all the classes which are essential to write simple
- * java program or to develop complicated java applications.
- * To use classes of `java.lang` package we need not write any import statement of fully qualified class names.

Object class

- * Object is super-most class in java
- * Each and every class directly or indirectly inherits from object class

docs : <https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html>

* methods of Object class

-hashCode()

-toString()

-equals()

-wait()

-wait(long)

-wait(long,int) } --> Threads

-notify()

-notifyAll()

-clone() --> Object cloning

-getClass()

-finalize() --> Garbage collection

hashCode()

- * Returns a hash code value for the object.
- * hash code value of an object is an unique integer value which is generated by the jvm based on the address of the given object.

toString()

- * Returns string representation of the Object
- * String representation contains
 - *Fully qualified class name
 - *@ character
 - *hexadecimal value of hashCode

equals(anotherobject)

- * it compares the hashcode value of given 2 objects and returns true if they are same else it returns false.

Strings Class



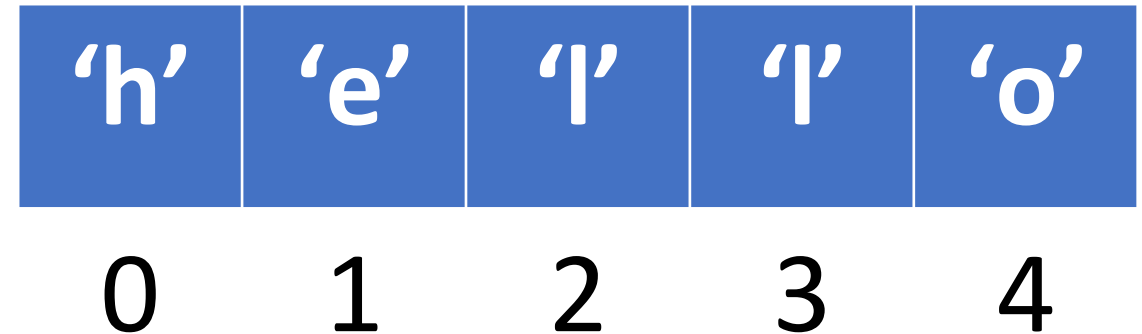
Training & Development Center

BASAVANAGUD

Strings

- String value is a group of characters which is written in the double quotes.
- We can access the characters or perform any operations on the string only by using the methods of String.
- Internally String value is stored as a character Array by the JVM.

String str = "hello"



Methods of String

- `length()` : It returns the count of number of characters present in the given String.
- `charAt(index)` : It returns the character at the given index from the given String.
- `indexOf(char)` : It returns the index of the given character from the given String. It returns -ve value if the character do not exist.
- `indexOf(char, startIndex)` : It returns the index of the given character from the given String by starting the search from the given startIndex.
- `equals(string)` : It compares characters of given two strings and returns true if they are exactly same else it returns false.

Methods of String

- `equalsIgnoreCase(string)` : It compares characters of given two strings by ignoring the case and returns true if they are exactly same else it returns false.
- `substring(start, no.of chras)` : It creates a String from the given start index with given no. of characters and returns the String.

- **NOTE**

- `length` and `length()`
- `length` -> variable used in array
- `length()` -> method of String

String Class

String value is group of chars written within double quotes

- * In java String is a class which is used to store group of chars the String object.
- * String class is immediate sub-class of Object class
- * String class implements Serializable, Comparable<String>, CharSequence interfaces
- * String class is final and it cannot be inherited.
- * hashCode() of Object class is overridden in String class which returns an integer value that is generated based on unicode values of every character present in given String

String Class

- * toString() of Object class is overridden in String class which returns the string value present in given string object.
- * equals() of Object class is overridden in String class which compares characters present in given 2 strings and returns true if they are same else returns false
- * String objects are created in special memory area called string pool in heap areaa
- * The String pool consist of two parts
 - 1.Constant pool
 - 2.Non-constant pool

String Class

- * String objects can be created in 2 diff ways
 1. By using new operator
 2. Without using new operator
 - * If you create a String object without using new operator then the object will be created in Constant pool.
 - * If you create a String object by using new operator then the object will be created in non-constant pool.
- Note : == operator compares address of given two Objects.
- * Within constant pool duplicates are not allowed.
 - * Within non-constant pool duplicates are allowed.

String Class

- * If you create any string object in following format then the resultant string object will be created in non-constant pool.

"" + Svar

Svar + ""

Svar + Svar

String class is immutable. Explain **** or Why String objects are immutable

If you try to re-intialize a string object then, a new object will be created with the new value & the reference variable starts pointing to new object leaving old object de-referenced.

String Class

Advantage : If 2 ref. variables are pointing same string object then changes done on the object through one reference variable will not effect another reference variable.

Disadvantage : If you reintialize same reference variable for n number of times then it will result in creation of multiple de-referenced objects which will occupy the memory space and may lead to OutofMemoryError.

To overcome the disadvantage of immutable property java developers created two new classes

1. StringBuffer
2. StringBuilder

StringBuffer and StringBuilder Classes



Training & Development Center

BASAVANAGUDI

StringBuffer Class

java.lang.Stringbuffer

- * write a function to return the first half of the given string
- * StringBuffer is mutable sequence of characters.
- * A string buffer is like a String but can be modified without creating any de-referenced object
- * StringBuffer is final and cannot be inherited
- * StringBuffer is immediate sub-class of Object class
- * Any modification to StringBuffer can be done only by using the methods of StringBuffer class

StringBuffer Class

- * + (concatenation) operator is not supported by StringBuffer
- * StringBuffer objects can be created only by using new operator.
- * toString() of Object class is overridden in StringBuffer class which returns the string value in given string object.
- * StringBuffer is a thread-safe class and safe for use by multiple threads

StringBuilder Class

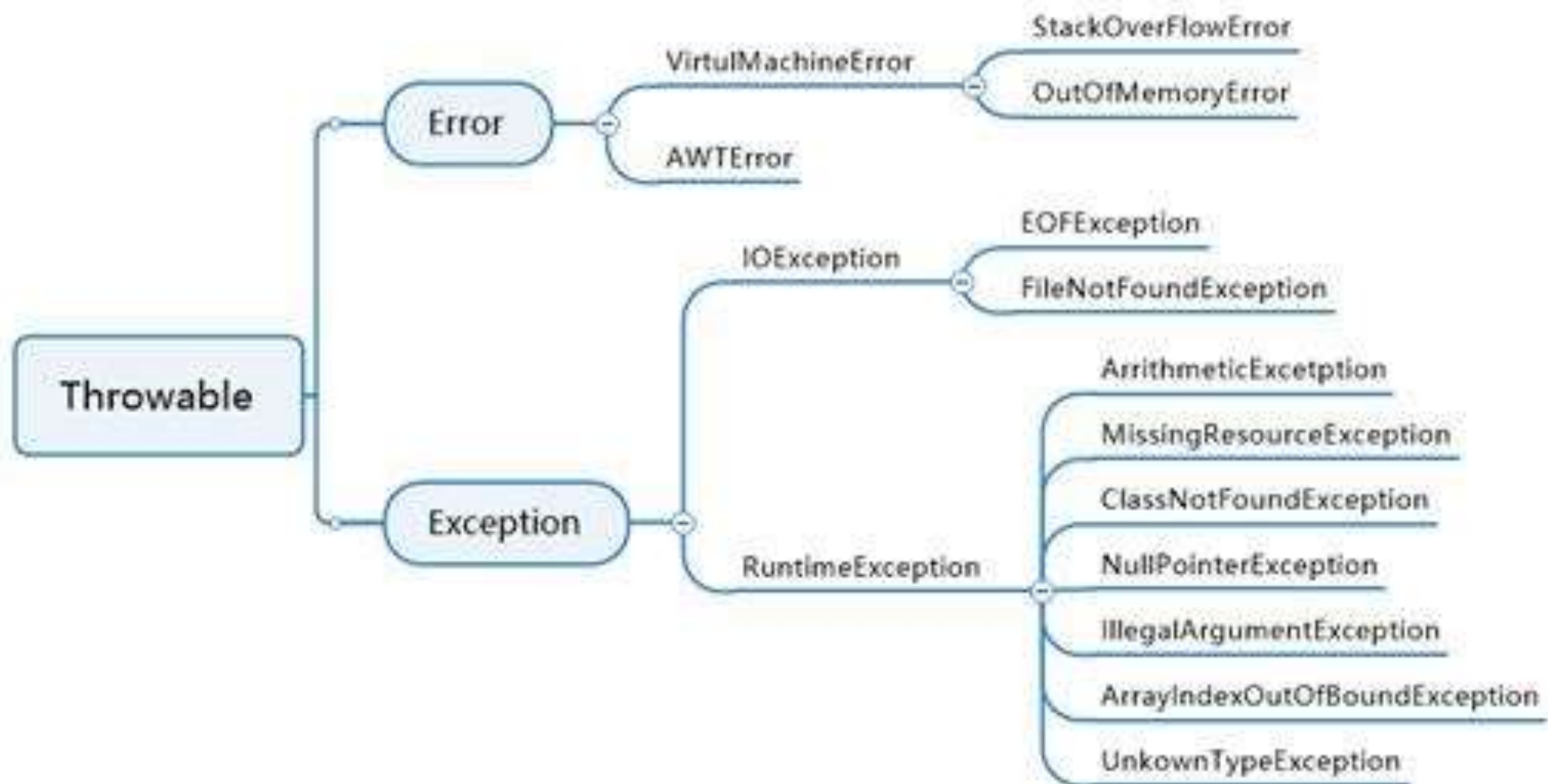
* StringBuilder are not safe for use by multiple threads.

Exception Handling



Training & Development Center

BASAVANAGUD



Exception Handling

- * Exception is an unexpected event which occurs at Runtime due to unexpected operation performed by a single line of code
- * Exception handling is writing code which will recover the program from the exception and continue the execution.
- * Error is an unexpected event which occurs at Runtime due to hardware or software failure of the system.
- * When ever there is an exception JVM will create an Object of corresponding exception class.
- * JVM will pass the exception object to the method which created the exception by using throw keyword.

Exception Handling

- * If the method is not able to handle the exception object, then JVM will terminate the method's execution.
- * If no methods are handling the exception object then JVM will call Default Exception Handler which will handle the exception Object.
- * Default exception handler will handle the exception and prints
 - Name of the exception
 - Reason for the exception
 - Complete stack Trace

Exception Handling

try-catch block

```
try
{
    risky lines of code
}
catch(ExceptionClass ref)
{
    alternate code
}
```

try with multiple catch blocks

```
Exception e = new ArithmeticException();
Exception e2 = new ArrayIndexOutOfBoundsException();
```

finally block

- * finally block is executed irrespective of occurrence of exceptions.
- * finally block should be written after the catch block.

Exception are of 2 types

1. Checked exception
2. Unchecked exception

- * Exceptions which are not checked by the compiler at the compile time are called as unchecked exceptions.
- * Leaving RuntimeException and its Subclasses all other exceptions are checked exceptions.
- * RuntimeException and all of its Subclasses are Unchecked exceptions.

* **Exception propagation :**

Passing the exception object from called method to calling method is known as Exception propagation.

* Unchecked exceptions will be implicitly propagated by the JVM.

throws keyword

* throws keyword is used to propagate checked exceptions explicitly by the programmer.

throw keyword

* throw keyword is used to throw the exception explicitly by the programmer according to the application requirements.