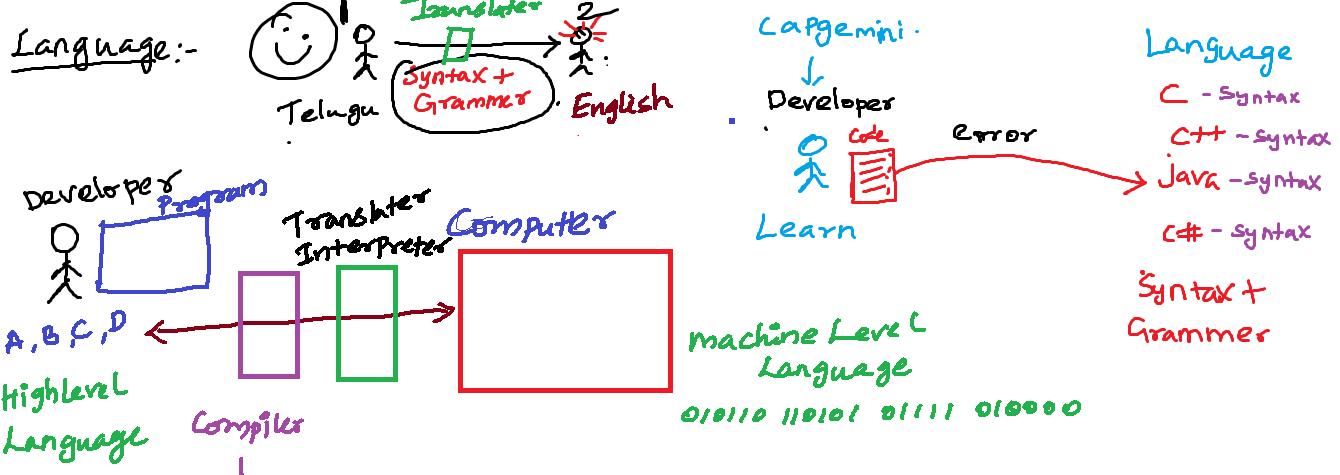
Basics of Language:-



Eclipse IDE:-

Workspace:

It is a folder contains one or more related java projects.

Project:-

It is a folder. It contains one more files related to that project

One or more java files

JRE System Library:-

Has all predefined java classes

Src :-

All user defined .java file must be place inside the src folder

Build -> javac filename.java

Run -> java filename

JDK

JRE

JVM

Sample.class 🡪 JRE 🡪 Machine code(output)

Sample.class 🡪 JDK(JRE), JRE 🡪 Machine code(output)

Sample.java🡪 JDK 🡪 Sample.class🡪 JRE🡪 Machine code(output)

.apk(.classes)

|

Phone (JRE)

|

Machine (Output)

Identifiers:-

All words in java. It is called as identifier

Reserved Keyword or User Defined Keyword

package name, class name, method name, variable name

Package Name:

* All letters must be in lower case
* Should not contains any space

Variable Name:-

* A java variable can start contain (a-z A-Z 0-9). Special character \_ $
* Variable name should starts with a character(a-zA-Z \_ $).
* It should not start with a number
* Variable name should be meaning full name
* If the variable is constant variable then variable name must be capital letter
* If variable name is single word all letters must be smaller case
* If variable name contains more than one word then first word all letter smaller second word onwards each word first letter must be capital

Method name:-

* If method name is single word all letters must be smaller case followed by bracket()
* If method name contains more than one word then first word all letter smaller second word onwards each word first letter must be capital then followed by ()

Class names:-

* Class name each word starting letter must be in capital all the remaining letters in lower case

Interface Name:-

* interface name each word starting letter must be in capital all the remaining letters in lower case

class Syntax:-

package <packagename>;

import <packagename>;

accessmodifier class <classname>{

constructor

variable

methods

}

Class in java:-

A class contains related members

Members – Member variable, Member Functions

Class is template or blueprint of an object

Class is a container in which we write related variable and methods (Encapsulation)

Object Oriented concept:-

Class and Objects:-

Why Functions?

Execute some logic or task

1. Function without arg without return value

**public** **void** multiply() {//perform task or logic

**int** a=29;

**int** b=45;

**int** c=a\*b;

System.***out***.println(c);

}

1. Function with arg without return value

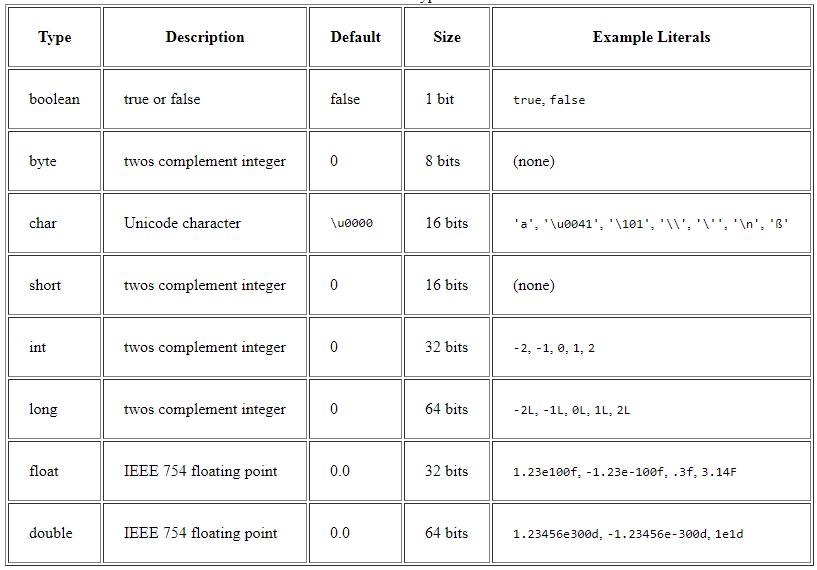
**public void add(int a,int b) { //perform task or logic**

**int c=a+b;// logic**

**System.out.println(c);**

**}**

1. Function without arg with return value
2. Function with arg with return value



Variable in java:-

Variable is an identifier for the memory location

Variable declaration

Datatype variablename;

Variable initialization

Variablename=value;

String name= “rajesh”

String rollno;

String year;

String sem;

int phone;

String email;

float cgpa;

rajesh, 3555SF, IV, VIII, 445454, [rajesh.kit@gmail](mailto:rajesh.kit@gmail), Chennai, 6.4

Types of variables:-

1. Instance variable
   1. Declared inside the class and outside of any method in the class without static keyword
2. Local variable
   1. A variable declared inside the method or block
3. Static variable
   1. Declared inside the class and outside any method with static keyword
4. Reference variable

Types of methods:-

1. Instance method OR non static method
2. Static method OR non instance method

Packages:-

Collections of related java classes, interfaces and enums.

Avoid the naming conflict

Variable declare:-

---------------------------

Accessmodifier Nonaccessmodifier datatype variable;

Class declaration:-

---------------------------

Accessmodifier Nonaccessmodifier class classname{

}

Function:-

Accessmodifier nonaccessmodifier returntype functionName(arg){

-----------------;

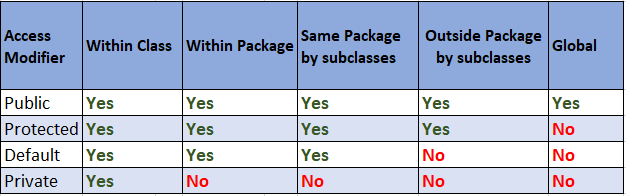
-----------------;

}

Access modifiers:-

private, default, protected, public

* Can be added before variable, method, class , interface



Non-Access Modifiers:-

Non access modifiers changes the default behaviour

Final, static, abstract, synchronized, volatile, native, transient

int mark=100;

Variable initialization:-

* Local variable -> while declaring a local variable we must initialize the value
* Instance or static variable
  + During variable declaration you can initialize

Flow Control

Conditional control statement

If, if else, if elseif, switch

If:-

if(condition){

statements;

}

Un-Conditional Control Statement

Break, label

{

}

Block – grouping set of statements

switch(condition or choice){

case label:

statements;

case label:

statements;

case label:

statements;

default

statements;

}

Java Operators:-

Assignment Operator

Int a+10;

Relational Operator -> comparing between values. It always returns a Boolean true or false

10>20 -> false

10>=20 -> false

10<20 -> true

10<=20 -> true

<,<=,>,>=,==,!= or <>

Arithmetic Operators

+,-,/,\*,%

Conditional Operator:-

(condition) ? value1 : value2; if(condition){

Statements

Else{

statements }

logical operators:

&&, ||, !

Control Statements:-

You can control statement or flow of statement execution

Conditional Statements:- if, if..else, if..elseif..else, switch

Un Conditional Statement:- break, continue, label

Looping statement:-

One or more java statement can be executed finite no of times. You have to loops

For, while , do..while, advanced for loop

For loop:-

for (initialization;condition;increment/decrement){

statements;

}

Advanced for loop:-

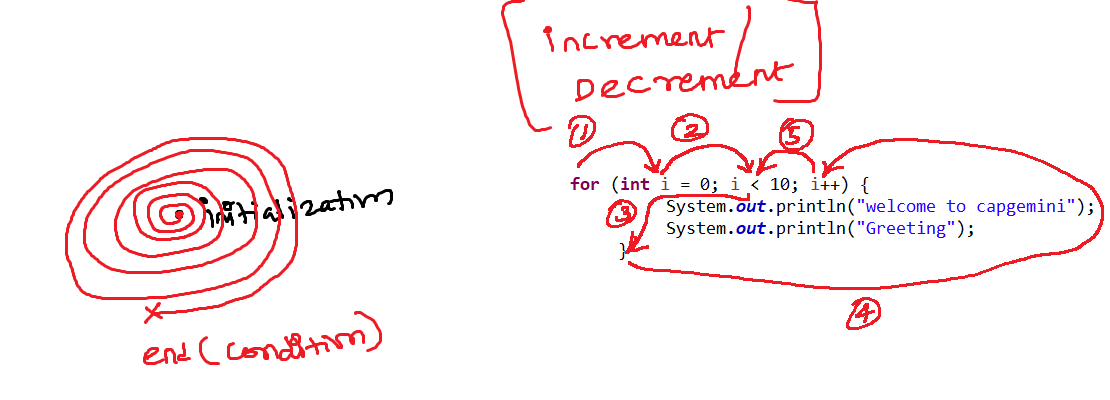
-----------------------------

for(datatype variablename:arrayname){

statements;

}

Instance of Operator:-



while loop:-

while(condition){

statements;

}

do while:-

initiliaztion

do{

statements;

increment/decrement;

}while(condition);

break and continue

Unlabelled break and Unlabelled continue

labeled break and labeled continue

Array:-

Group of similar data stored in a single variable is called as array

Array index starts with zero end with size-1

Eg:-

int[] mark=new int[] {46,34,67,23,78,89};// primitive array

String[] city=new String[] {"","","","",""};// reference type array - String predefined class

Employee[] employees=new Employee[] {e1,e2,e3,e4,e5,e6};//reference type array - Employee userdefined class

Object Programming:-

Classes:-

Encapsulation – data (variable) and the code (method) together kept inside the class it is called encapsulation

Putting all variable and the method inside the class

To make perfect encapsulation

All variable you need to keep it as private

Create a public getter and setter method through which other programs can

Access your data

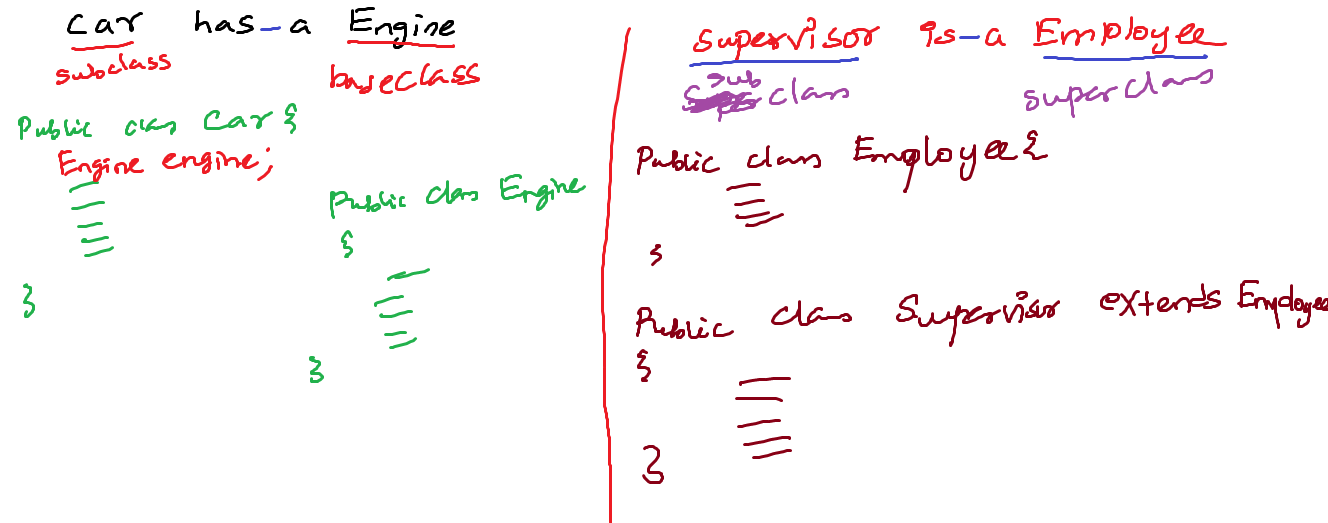
Inheritance can be achieved in two ways:-

Is-a inheritance

If you use **extends or implements** keyword to do a inheritance

Has-a inheritance

If you create an **reference** of one class into another class



Polymorphism:-

More than one form

A task can done more than one form

Send

Person A -----------------------------------------------------------🡪 Person B

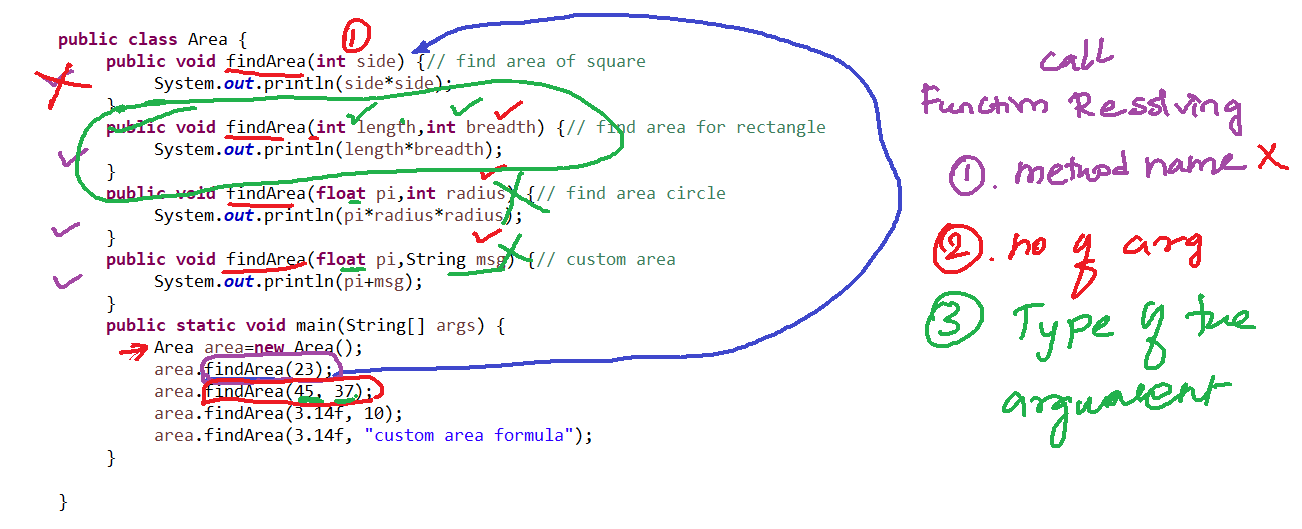
SendParcel (courier)

SendParcel (speedpost)

SendParcel (friends)

SendParcel (myself)

findArea:-



Function call resolving or method binding:-

For the function, call which method implementation or definition get executed decision-making is called function call resolving

If any conflict comes in the function call resolving

The function call resolving decision will be made based

1. Method name
2. No of arguments
3. Type of the arg

Function with same different argument in same or different class it is called as method overloading

Function with same same argument in different class it is called as method overriding

Abstraction:-

Hiding the implementation and showing the essential information is called abstraction

Car has very complex many mechanical parts, as you need to know how to drive the car

Ex:1

**public** **void** findArea(**float** pi,**int** radius) { System.***out***.println(pi\*radius\*radius);

}

This Implementation show all the details it is not abstraction

If method does have any implementation, it is called concrete method or implemented method or non-abstract method

Ex:2

**public** **void** findArea(**float** pi,**int** radius);

It is not the complete implementation and it is showing only the essential information (abstraction)

If method does not have any implementation, it has only the declaration. Abstract method or unimplemented method

Two ways you can implement abstraction:-

1. Abstract class - 100% abstraction
   1. Abstract class does not allow you to create object but you can inherit the abstract class

To other classes

1. Interface – 100% abstraction

If all the methods abstract method

For the abstract class and interface, you cannot create an object. But you can create reference variable

|  |  |
| --- | --- |
| Abstract Class | Interface |
| 1. Abstract class can have concrete or implemented methods | Interface can have abstract method or unimplemented methods |
| 1. An abstract class should have minimum one abstract methods may have zero or more concrete method | All the methods in the interface are abstract methods |
| 1. Inherit the abstract class to other child class **extends** keyword | Inherit the interface to child class **implements** the interface |
| 1. Abstract class methods need to explicitly add abstract keyword for abstract methods | All the method in the interface by default public abstract methods |
| 1. All the variable access modifier non access modifier default set by the developer | All the variable declared in the interface by default public static final variable |

A Class member:-

1. Member variable
2. Member function or method
3. Constructor
   1. Constructor is similar to your member function. It is special function that should not contain the return type and the name should be similar to your class name.
   2. Constructor will be called automatically. When we create an object for the class. Right after the object is got created then the constructor will be automatically

Constructor overloading

* A constructor with the same name and diff argument or parameter is called constructor overloading

No arg constructor

Arg constructor

Default no arg constructor

//jvm is going add one no arg constructor automatically by default during excution no arg default constructor

// jvm default no arg constructor it will all the instance and static variable and it initiliaze with the default values

Casting:-

Primitive Type Casting

* Primitive type
  + Converting one primitive to other primitive type it us called as primitive type casting
    1. Implicit type Casting or widening

byte – 1 byte – 8bits

short – 2 bytes – 16 bits

int – 4 bytes – 32 bits

long – 8 bytes – 64 bits

float – 4 bytes – 32 bits

double – 8 bytes – 64 bits

char – 2 bytes – 16bits

boolean – 1 bit

Explicit type Casting or Narrowing

Object Type Casting

* Object Type
  + Converting One object type to another Object type it is call Object type casting
    - 1. Implicit Object Casting or widening

Taking child class object reference and assigning into a parent class object reference

* + - 1. Explicit Object Casting or Narrowing

Taking parent class object reference and assigning into a child class object reference

Wrapper classes:-

A Object or reference type of primitive value

Primitive types (Wrapper classes)

byte – 1 byte – 8bits ----🡪 Byte

short – 2 bytes – 16 bits ----🡪 Short

int – 4 bytes – 32 bits ---🡪 Integer

long – 8 bytes – 64 bits ----🡪 Long

float – 4 bytes – 32 bits ----🡪 Float

double – 8 bytes – 64 bits ---🡪 Double

char – 2 bytes – 16bits ----🡪 Character

boolean – 1 bit ---🡪 Boolean

**Converting primitive type to the reference type or object type, we can do it by wrapper classes**

**Collection in java -> upcoming topic**

**Collections in java always deals with the object or reference types not with the primitive type**

**As a programmer -🡪 as a primitive type**

**User 🡪 Input 🡪 primitive type**

**public** **static** **void** main(String[] args) {

**int** salary = 10;

// Way 1:// convert primitive int type to the ref type

Integer iSalary = **new** ~~Integer~~(salary);

System.***out***.println(iSalary);

//Way 2:// convert primitive int type to the ref type

Integer i1=Integer.*valueOf*(10); // static method then we used to call the method using classname

System.***out***.println(i1);

**int** cSalary=i1.intValue();// converting reference Integer object into primitive int type

**int** icSalary = iSalary.intValue();

Integer ii=3456;// Autoboxing - automatically converts primitive value to reference type

**int** cc = ii; // autounboxing - automatically converts reference type into primitive type

**float** f=34.34f;

Float f1=Float.*valueOf*(f);

**float** f3 = f1.floatValue();

**char** ch='4';

Character c = Character.*valueOf*(ch);

**char** c1 = c.charValue();

}

Primitive type to reference type

Reference type to primitive type

String to a wrapper class reference type

String to a primitive type

Syntax error:-

Java exception:-

During the java code execution (Runtime), an abnormal condition will happen it is called **Java Exception**

**If exception is not handled it terminate the program execution abruptly** or immediately, it will allow to not execute the program further statements.

If the programmer handles an exception can terminate the program gracefully

Exception in thread "main" java.lang.ArithmeticException: / by zero

at exceptionhandlingdemo.Demo.main(Demo.java:9)

**----------------------------**

**Exception in thread "main" java.lang.NumberFormatException: For input string: "sfd20"**

**at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)**

**at java.base/java.lang.Integer.parseInt(Integer.java:652)**

**at java.base/java.lang.Integer.parseInt(Integer.java:770)**

**at exceptionhandlingdemo.Demo.main(Demo.java:10)**

**--------------------------------------------------**

**Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 7 out of bounds for length 3 at exceptionhandlingdemo.Demo.main(Demo.java:12)**

**Exception Type - java.lang.ArithmeticException**

**Exception message - / by zero**

**Packagename.classname.methodname the line number where exception created**

**When an exception happens during the code, execution it will create java exception class object. That exception object has**

1. **Exception Type**
2. **Exception message**
3. **Packagename.classname.methodname the line number where exception created**

Java exception handle:-

1. Try catch

**Try block where you suspect the code can create an exception**

**try**{ //try block

}**catch**() {// catch block

}

**During the code execution finally block always gets executed whether the exception created or not created**

**finally**{ // finally block

}

1. Propagate the exception throws keyword

Upto jdk 1.6

After jdk 1.7

Java Exception:-

1. Checked exception

If the exception class extends Exception class its comes under the category of Checked Exception

1. Unchecked Exception

If the exception class extends RuntimeException class its comes under the category of Unchecked Exception

Own Exception or Custom Exception or User Defined Exception:-

============================================================

Bank – account no – “”, null – InvalidAccountNumberException

Money->negative–InvalidDenominationException  
 Aadhar No -> InvalidAadharNumberException

If my **exception class** extends **Exception** – **custom checked exception**

If my **exception class** extends **RuntimeException** - **custom unchecked exception**

String:-

Collection of characters are called as string

String name=”capgemini”;

char[] name=new char[9];

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| c | a | p | g | e | m | i | n | i |

Name

String is a predefined class

Strings are immutable (value will not changes)

Two ways to create a String

1. String literal (Immutable)
   1. String collegeName=”IIT”;
2. String class object(Immutable)
   1. String collegeName=new String(“IIT”);
3. StringBuilder(Muttable)
4. StringBuffer(Muttable)

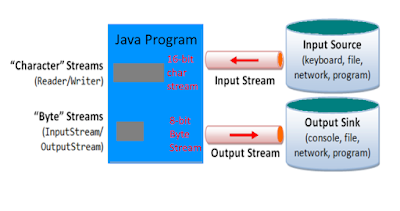
Java IO Stream:-

Input

(System.in) Keyboard, browser, File 🡪 Java Program

Output

Java Program 🡪 console, browser(System.out), File



Two types of streams in java

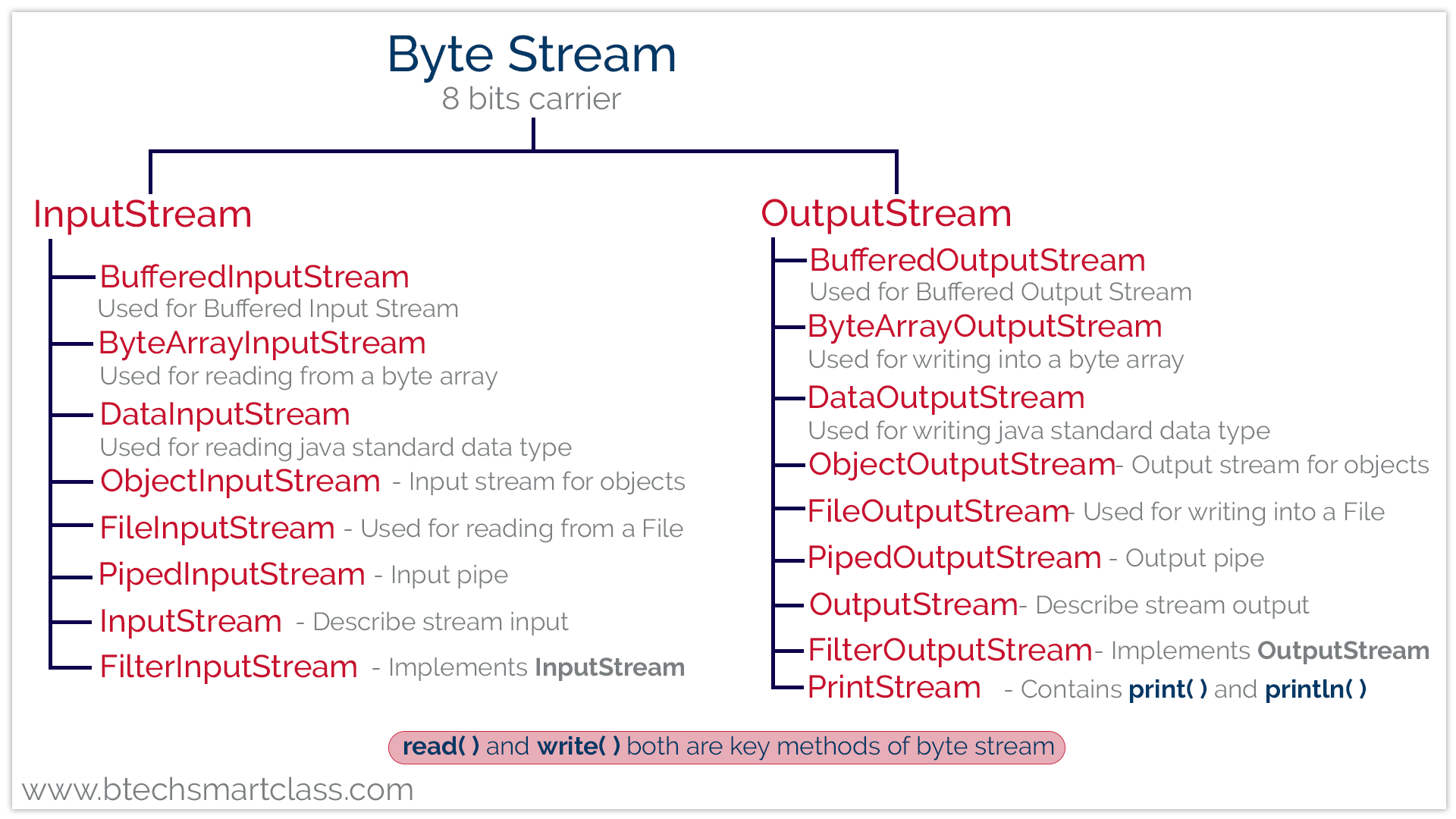
1. Byte Oriented stream (0-128)

Read or write the binary form of data

Image, video, audio

Read -> InputStream

Write -> OutputStream



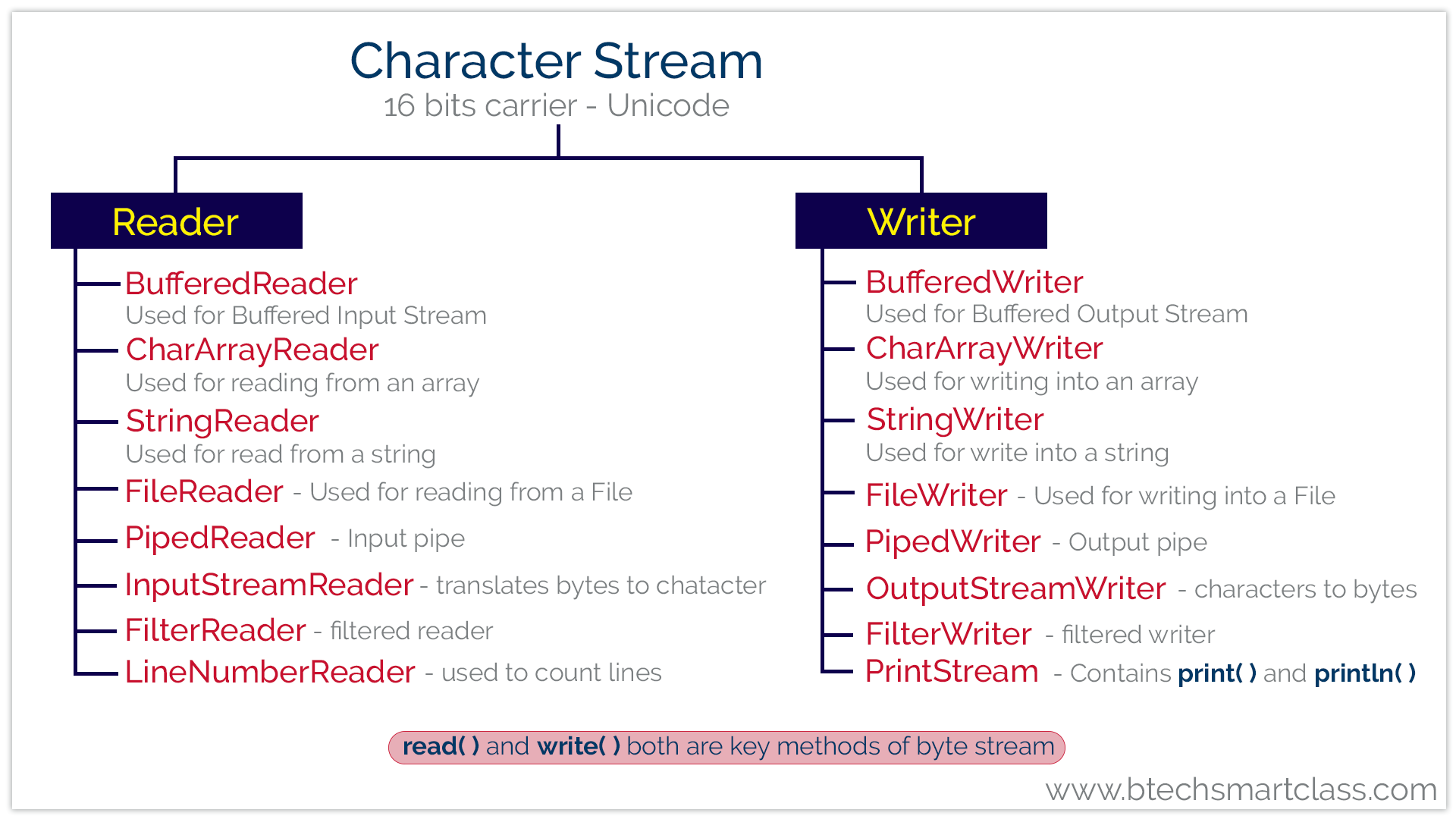
2.Character Oriented stream (0-256) – read / write

If you want to read text or character

Doc, xls, txt

Read -> Reader

Write -> Writer



Resources in java:-

If your java code or predefined class java code uses system hardware those classes are called as resources

Serialization:-

Array:-

Group or collection of similar types of primitive values or reference values (objects) are called as array

The group of values can be accessed by a single array variable name along with the index number

Array Index starts from 0 to size-1

Array -> primitive values, Reference values

Pro:-

* Array elements or item can be access by sequentially and randomly

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 56 | 34 | 89 | 45 | 9 | 80 | 44 |

Search 44

for(int i=0;i<array.length;i++){

if(array[i]==44){ //7 sec

}

}

Array[6]==44 – 1sec

Cons:-

* Once array size is fixed then you cannot increase or decrease further
  + int[] marks=new int[5];// static size
* Insertion and deletion of element involves movement of physical data back and forth.

Collections:-

Collection is a collection of or group of similar type of reference values (Object) that can be accessed by single variable name are called as collection.

Collections are dynamic in size. Size grow and shrink

Collection is a framework in java. Set of interfaces abstract classes and concrete classes

Each collection classes are backed by one data structure algorithm

All the collection are available java.util package

Collection stores a group object.

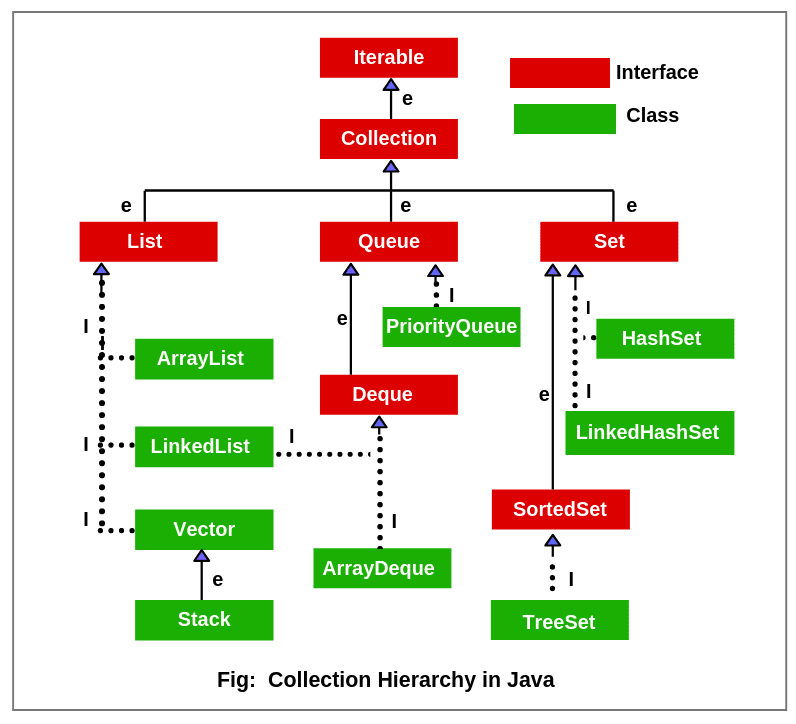
1. predefined object

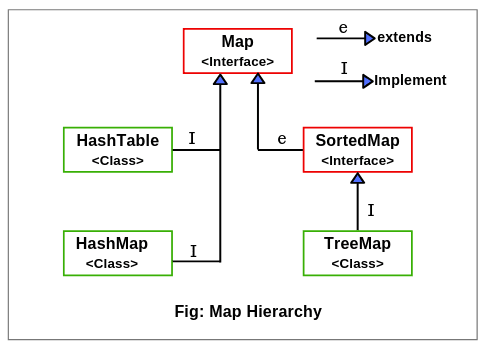
2. userdefined object

Datastructures -> array, linkedlist, stack, queue, hashing technique, tree and graph

Collection Class Hierarchy:-

===========================





How to you decide which collection class you need to use

1. Ordered or unordered

2. Duplicate

3. null

Collection – store group of objects

String s=new String();

s -> predefined string class object

Employee e=new Employee();

e -> user defined Employee class object

ArrayList list=new ArrayList();

-> array[16\*2] – index – random access

List.add(34);

LinkedList -> Doubly LinkedList datastructure

|  |  |
| --- | --- |
| ArrayList | Linkedlist |
| Array ds | Doubly linkedlist ds |
| Random access | Sequential access |
| Insertion deletion is very difficult | Insertion and Deletion is easy by rearranging the pointers |

Vector:-

HashSet :-

Unorederd

Not Accepting the duplicate (unique)

It will accept one null value

LinkedHashSet:-

Ordered

Not accepting the duplicate (unique)

It will accept one null value

TreeSet:-

Sorted Order (unorder)

Not accepting the duplicate (unique)

It will not accept even one null value

|  |  |
| --- | --- |
| Set Interface | Map Interface |
| HashTable,HashSet, LinkedHashSet, TreeSet | HashTable,HashMap,LinkedHashMap,TreeMap |
| HashSet – Unordered, no duplicate, one null value | HashMap – Unordered key, no duplicate key, one null key |
| LinkedHashSet – Ordered, no duplicate, one null value | LinkedHashMap – Ordered key, no duplicate key, one null key |
| TreeSet – Sorted in Ascending order, no duplicate, not even a single null accepted | TreeMap – Sorted in Ascending order key, no duplicate key, not even a single null key accepted |

List -> ArrayList, LinkedList, Vector -> Collections.sort()

Comparable –

Comparator –

HashSet,LinkedHashSet,TreeSet

Multitasking:-

More than one task executed in a computer is called as multitasking

Multitasking can achieved by 2 ways:-

1. Process based multitasking

Heavyweight

Context switch between the processes very costly

2. Thread based multitasking

Lightweight

Context switch is easy

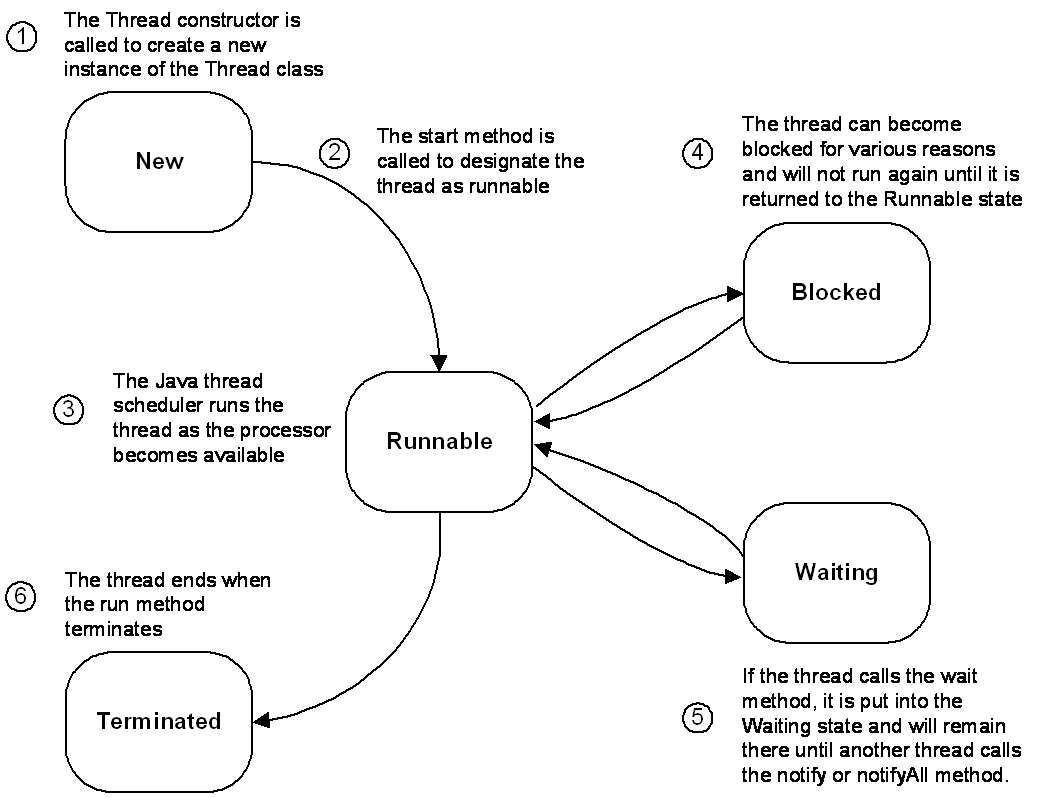
Thread -> executable in a cpu or processor

In java you can create a thread

Two ways you can create a thread

1. By extending a thread class

2. By Implementing a Runnable interface



Thread deadlock:-

All the threads are waiting for the same resource

Synchronized – can control the deadlock

Jdk 1.8:-

=======

Functional interface

Default method

Static method

Lamda expression

forEach

Stream API

LocalDate Loclatime

Functional Interface:-

* If an interface has exactly one abstract method that interface can call it as functional interface
* You can have one or more implemented method inside then functional interface
* From jdk 1.8 inside the interface you can write instance method with the default keyword and static method static keyword

1. User defined functional interface

2. Predefined functional interface

Runnable

Comparable

Comparator

Predicate

Function

Consumer

Lamda Expression rules:-

1. lamda expression does not take access modifier and nonaccess modifier

2. lamda expression does not take return type

3. lamda expression does not take method name

4. lamda expression after the arg need to place an arrow operator (->)

5. lamda expression implementation has only one statement can remove curly braces

5. lamda expression implementation has only one statement that too it is a return statement you can remove return keyword

Lamda expression:-

5. In the lamda expression has an arguments you can the data type of the argument

It provides a clear and concise way to represent one method

public void add(){

s.o.p(“welcome”);

}

Equivalent Lamda Expression:-

----------------------------------

()->s.o.p(“welcome”); // lamda expression

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public void add(int a,int b){

s.o.p(“welcome”);

}

Equivalent Lamda Expression:-

----------------------------------

(a,b)-> s.o.p(“welcome”);

public void add(){

return 56;

}

Equivalent Lamda Expression:-

----------------------------------

()-> 56;

public void add(int a,int b){

return a+b;

}

Equivalent Lamda Expression:-

----------------------------------

(a,b)->a+b;

Lamda expression can be written with the help of only **functional interface**