CORE JAVA

JDK-15

```
public static void main(String[] args)
```

Note - while making a class name the file, first letter of the filname should be capital

eg - Hello, Basics

Comment - //

- Java is a case sensitive language.

To print - System.out.println("I am Tanu");

```
Public Class Demo
{
   Public static void main (String args[])
   {
      System.out.println("welcome to TechM : Demo");
       Dev d = new Dev();
       Clerk C= new Clerk();
       Test t= Test();
   }
   Class Dev
{
      Dev()
        { System.out.println( "welcome : Dev ");
        }
}
Class clerk
{      clerk()
       {  System.out.println("welcome: clerk");
```

Scanner – Used to take i/p values.

```
import java.util.Scanner;   ( just like #include <stdio.h> )
class Emp
{
   Emp()
   { Scanner sc = new Scanner (System.in);
     System.out.println ("Enter A value:");
      int a = sc.nextInt();
     System.out.println ("Enter B value:");
      int b = sc.nextInt();
      int c = a+b;
      System.out.println ("Result is :"+c);
   }
}
Public class Demo 2
  {  public static void main(string arg[])
     {
        Emp e = new Emp();
     }
}
```

```
import java.util.Scanner;
class Emp
{   String name; int age; int sal; String desig;
Emp()
  {
    Scanner sc = new Scanner(System.in);
    System.out.print("Name : ");
    name = sc.next();
    System.out.print("Age : ");
    age = sc.nextInt();
    System.out.print("Designation : ");
    desig = sc.next();
  }
  void display()
  {
    System.out.println("----------");
    System.out.println("Name : "+name);
    System.out.println("Age : "+age);
    System.out.println("Salary : "+sal);
    System.out.println("Designation : "+desig);
  }
Public class Demo2
{
  public static void main(String args[])
  {
    Emp e = new Emp();
    e.display();
  }
}
```

objects — Do all the work-

Class- Classes contain instructions for how object can be created, as well as how the objects carry out certain behavior

constructer - Constructor ~~method~~ is created with the same name as class name

for taking different data in while calling-

right click → source → Generate Constructor using fields. → ok

~~this is the variable in~~

this - this keyword is a variable in java & it points to current object back in (C) class

It represents particular instance

- Objects Exist During Application Runtime

stack - (LIFO)

Heap — Say Car myCar;

myCar = new Car();

It is not the actual object. It is the address for in memory where that object is located

Garbage collection - It is a process that runs in a the heap. It looks for those variables which has nothing no object & eliminates to clear the space in the memory

Say my car = new Car();

my-car = new Car(); ✓ this will work

So this is garbage now

Reference variable - myCar = new Car();

Instance variable - used to maintain state for that particular variable object of the these two instances.

class Inheritance -

Interface – does not have body hence called abstract methods

Interface is a contract (It promises to implement that process).

- A class can only extent 1 class. But can have multiple interfaces.

* Abstract classes – abstract class can only be extended

so cannot instantiate an instance of an abstract class. It is only meant to be parent of another class.

Say – public abstract void move() var

(add unimplimented method).

→

Polymorphism – makes program dynamic

```
moveAnimal(fish1);
moveAnimal(sparrow1);
}
public ~~class~~ static void moveAnimals
                (Animal animal) {
    animal.move();
}
```

Substring — (2) CDEFG

0	1	2	3	4	5	6
A	B	C	D	E	F	G

string str =

str.substring (0,2) → "AB"

str.substring (3,7) → "DEFG"

To compare we usually use == ✗ but here
java use .equals

say if(a.equals("hello")) → for simply apply → !;

& if you want about alphal case letter say assigned "there" written "THERe"
to resolve & for this apply

say if (b.equalsIgnoreCase ("THRE"))

Also — to find particular string.

```
string a = "hello";
string b = "there";
system.out.println(a.charAt(3));
}
```

→ l.

b=str.lastIndexOf("j")

to find index — say

```
string str = "Hello there yogi";
Int a = str.indexOf ("there");
System.out.println (a);
```

O/P → 6

while loop

for not conditn
while (!(count<=10))

```
int count = 0;
while (count <= 100) {
    System.out.println ("Aditi" + count);
    count = count + 1;
```

break;

runs only 1 time & stop.

→ Aditi : 0

for loop

String name = "ABJJKIM";

```
for (int i = 0; i <= 100; i = i + 2)
{
    System.out.println (i);
}
```

i += 2

prints all even numbers

Nested for loop & debugger

```
for (int idx = 0; idx < 100; idx++) {
    for (int j = 0; j < 10; j += 2) {

        System.out.println ("the value of idx was"
            + idx + "------" + j);
    }
}
```

compiling Java program using command line

say- filename - Multiplier

for compiling code -

javac Multiplier.java

for running - java multiplier.java

creating and Deploying executable Programs using JARs

JAR (Java ARchive)

cd - change directory
pwd - present working directory

say to runtime program → java -jar Mario.jar

→ jar -tvf Mario.jar → to see all files | file

T → table of contents
v → verbose
f → file

→ jar -xvf Mario.jar (x-extracts)

To extract content of the files

To create —

$ jar -cvfm myprogram.jar manifest.mf *.class

all the class in current directory means taking all attributes of .class file

Here c — create
m — manifest

so the new file will be created called my program & under it manifest will be added.

Can also do $ jar -cvf a.class b.class c.class

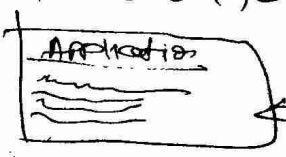
Scanner — used to take input from user

for it specific libraries are their see below

```
import java.util.scanner;
public class Application{
  public class void main(string[] args){
    for(int i=0 ; i<=3 ; i++){
      scanner input = new Scanner(system.in);
      system.out.println("Enter some Text:");
      string enteredtext = input.nextline();
      system.out.println(enteredtext);
    }
  }
}
```

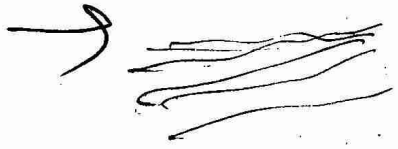
scanner can also be used to read data from files

you will get this, hover your mouse on to File



```
import java.io.files
public class Application {
    public static void main(String[] args) throws ~~~ {
    }
    File file = new File ("myfile.txt");
    Scanner input = new Scanner(file);
    while (input.hasNextLine()) {
        string line = input.nextline();
        System.out.println (line);
    }
    input.close();
```

hover your mouse → Add throws declaration



Note - we don't invoke methods for null obj.

- public class FileNam extends Exception {

FileNam: child class; Exception: parent class

Collection → Array list <type>

Java fls - also has Generics

⇒ Basically the linked list is faster for manipulating and the array list is faster for retriving data to retrieve values from a link list

In array to add & remove

Variable.remove(1); ((1,2,3...))

Csey number.remove(2);

Variable.add("Hello");

Csey words.add("there");

for removing first ~~number~~ cea console

Variable.removFirst();

any number

Not - in windows cd called a change directory.

Set – A set is basically very similar to a list, except that it prevents duplicates.

MAP – is basically like dictonary. (u will see some words & defination).

in MAP interface remember key difference of adding is variable name. put ("key", "value");

cuz normaly add we do .add but here

Tree map – gives the I displays the order is which I like typed. in correct order

TreeMap | HasmapLink — cannot be ~~used for~~ have duplicates. ~~cop~~

Note – u cannot use primitives in generics

finally – is a block which will execute always.

finilaize – method wich will remove the unwanted code

FileReader — reads the file but reads character by character

BufferReader — will read line by line

Generics - it is a advanced java syntax

- every collection class has Iterator

Threading — At any given point in time, only single line of code was executing.

multithreading — two or more lines of code can run at the same time

Note — its never legal to start a treand more than once

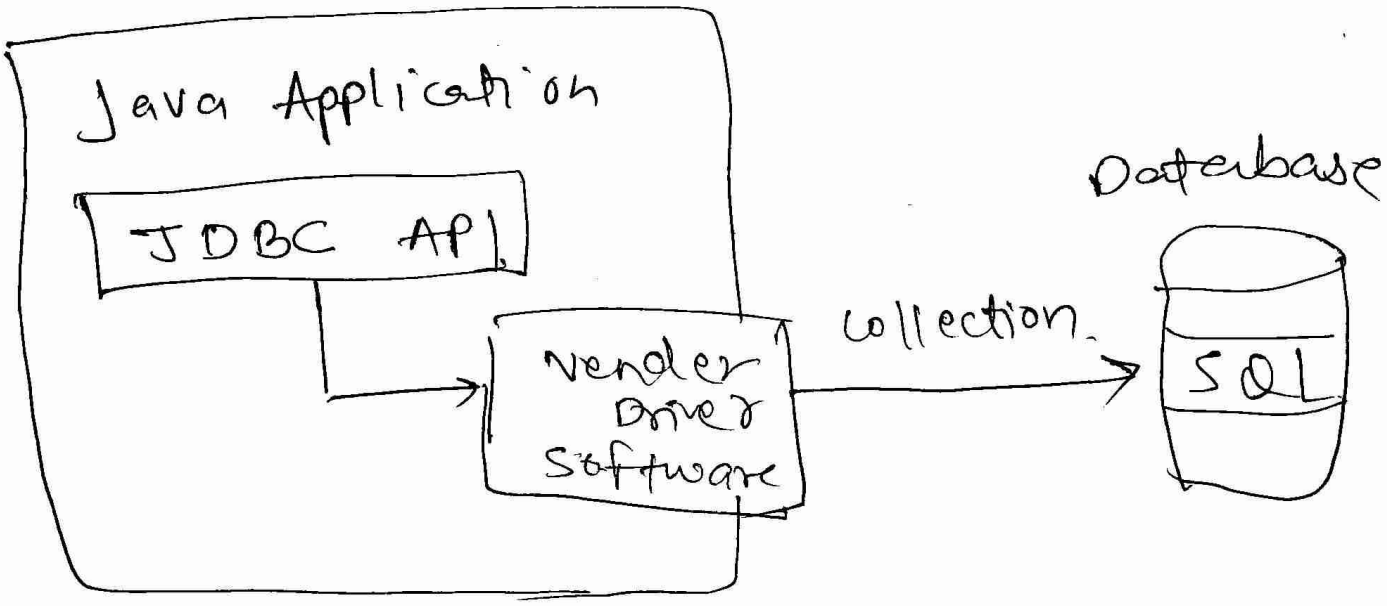
Pool — ExecutorService newFixedThread (2) 2 threads

- execute
- shutdown(), shutdownNow()
- isTerminated()). awaitTermination(,);

Vendors in MySQL used

postgres, microsoft, IBM, Oracle

JDBC



Jshell (commands)

~: $ java -version → To know version

~: $ jshell → welcome

/help → To get help

System.out.println("Hello") → to print
(u don't need ;)

> string var = "hello"
var ==> "hello"

> var
var ==> hello"

> /list - it will show all the commands run upto!

> void printTenTimes (string var) {
...> for (int i=0; i<10; i++) {
...> system
...> System.out.println(var);
...> }

Note ↳ under the block loop (;) is needed only

> /drop var (say /drop printTOTimes)
dropped variable var.

/edit — to edit method.

say /edit printTOTimes

↑ history - shows all thing u typed uptill

~: $ jshell -c /users/intiazahamed/Desktop/superman.jar

path correct

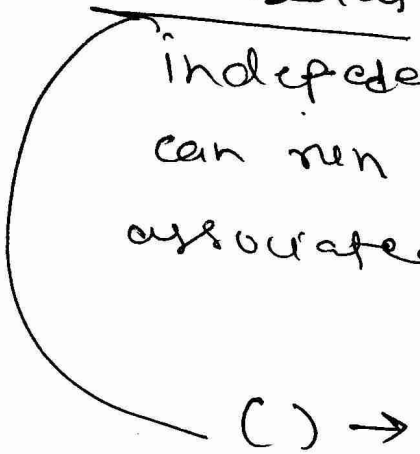
→ to include jar files to our jshell

> import com.printer.utils.*

/imports — u will able to see all imports

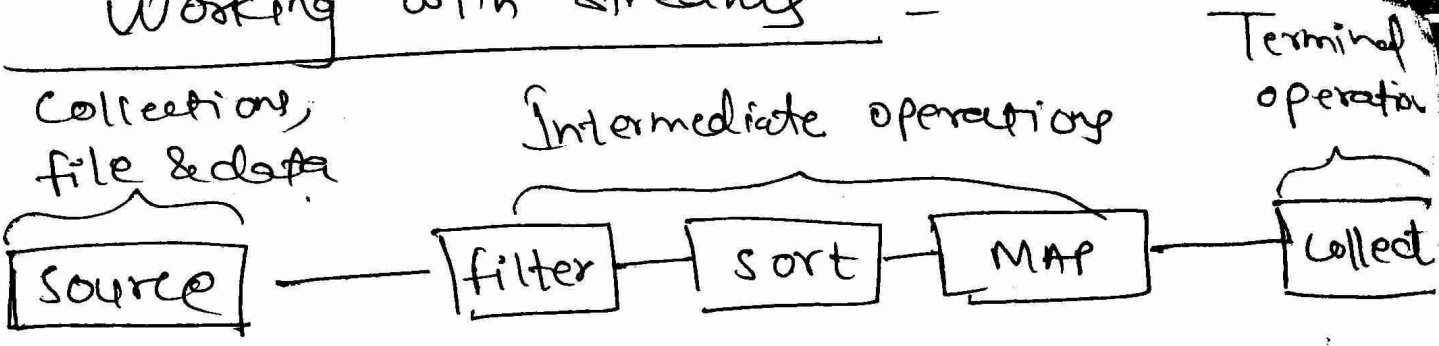
functional interfaces of Lamdas

→ Lambdas allow us to create independent blocks of code that can can run on their own, without being associated to any class. (in java 8)



Note → Lamdas are blocks of code used to implement a method defined by a functional interface

Working with streams -



Intermediate operations - Return a stream, so we can chain multiple intermediate operations together.

- 404 can have zero or more operations
- order matters : filter, sort or map.↑
- for very database, we can use ParallelStream → to enable ~~to perform~~ multiple threads to perform operations.

→ Eg - intermediate operation — filter() / distinct / map() / skip — sorted()

Terminal operations - can return void, or a non-stream result such as list or a single value

- One terminal operation only
- forEach() applies the same funct^n to each element
- collect() saves the elements into a collection
- reduce the stream to single summary element
  - count()
  - max()
  - min()
  - sum()
  - summaryStatistics()

IntStream
- forEach
- range()
- Stream.of
- sorted()
- findfirst()
- ifPresent

- startsWith
- map()
- average()
- stream

Arrays.asList

- toLowerCase()
- length()

- close()
- collect()
- contains()