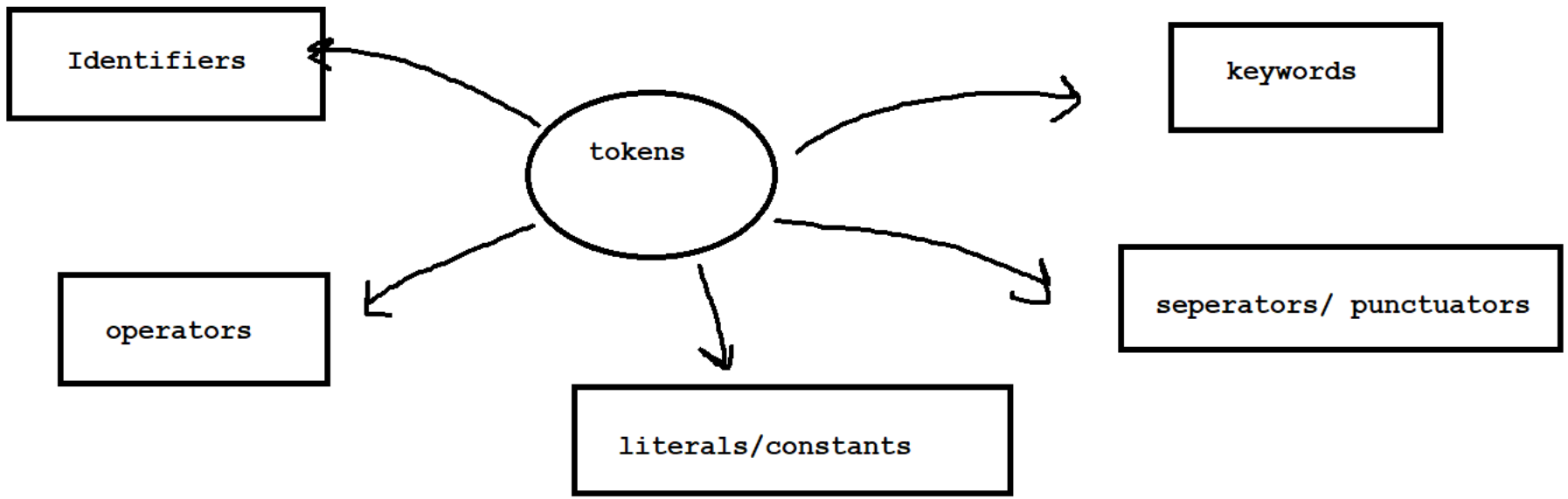
Token : every smallest individual uynit in java program known as tokens



Identifiers : user define things

rules:

1. Alphabets, numbers
2. there is no space allowed
3. first letter should be albhabet always Apple Apple123 123Apple
4. you can use numbers in between or at the end
5. no special symbols Apple@ (_ , $)
6. keywords are also not allowed (main, new, class, Main)

variables : variables are naming elements

10+20=30

a	b	c
10	20	30

syntax:

datatype variablename;

int a= 10;

int b;

Data tyes: describes what type of data you are going to use in your code

Data types

primitive

pre-defined datatypes

1.boolean(t/f)
2.byte
3.char (a to z / A to Z)
4.short (number)
5.int (numbers)
6.long (numbers)
7.float (point numbers)
8.double (point numbers)

non- primitive

combination of primitive

1.String
2. Array

	default value	size
Boolean	false	1 bit
char	'0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	2 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

types of variables

1. Local variable

-> decalre inside the class and surrounded by {} is called local v.

->constructor/block/method

```
package All_Programs;

public class local_variable {

    public void abc()// declaration of method
    {
        // local variables
        int a=10;
        double b= 3.2;
        System.out.println("a="+a);
        System.out.println("b="+b);

    }

    public static void main(String[] args) {

        local_variable obj= new local_variable(); // obj declaration

        obj.abc();//calling the method



    }

}
```

2. Instance Variables :

which declare inside a class but outside of the body of the method called instance variables.

```java
class ABC
{  // instance variables
    int a=10;
    int b;
    int c;
}


public class Instance_variables {

    public static void main(String[] args) {

        ABC obj= new ABC();

        obj.b=20;
        obj.c=30;

        System.out.println(obj.a);
        System.out.println(obj.b);
        System.out.println(obj.c);
```

3. Static Variables :

-> static variables also known as class variables
-> these variables are declared by using static keyword.

-> no object creation required.

```java
package All_Programs;

class Student
{
  static String name="sakshi";
  static double fees;
}


public class static_variables {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Student.fees=40000;

        System.out.println(Student.name);
        System.out.println(Student.fees);
    }

}
```

Operators :

1. Unary operator : (x++, x--, (pre -inc/ post-inc) ++a, a++)
2. Arithmetic op: (+,-,*,/,%)
3. Shift Operator: (shift the bits of num to left or right) (shift right(>>) shift left (<<))
4. Relational Operator: determine relationship (==, <,>,<=,>=,!=)
5. Bitwise Operator (bit by bit operation) (|, &, ^, ~) bitwise(or, and, xor, complement)
6. Logical op: (&&, ||, !) (and, or, not)
7. Ternary op: syntax: condition? value1 : value2
8. Assignment operator: (assign the values) (+=, -=, *=, /=, %=)

Conditional Statemnet

1. if statement
2. if- else
3. if - else -if
4. nested if

syntax: (true)

```
if(condition)
{
statement
}
```

nested-if

```
if(condition 1)
{
statement 1

   if(conndition 2)
   {
     statement 2
   }
}
```

if- else

syntax: (T/ F)

```
if(condition)
{
stament 1
}
else
{
statement2
}
```

if-else-if

syntax: more thn 2 condition

```
if(condition)
{
statement 1
}
else if(condition 2)
{
statement 2
}
else if(condition 3)
{
statement 3
}
else
{
statement 4
}
```

loops: repetation of any statemnet or condition multiple times.

```
sysout("hi");
```

Types :

1. for loop
2. while loop
3. Do while loop

1. initialization
2. condition
3. increment/ decrement

```
syntax : for

for(initialization;condition;inc/dsc)
{
statement
}
```

```
Syntax: while

initialization;
while(condition)
{
statement;
inc/dec
}
```

```
syntax: do while


initialization;

do
{
statement;
inc/ dec;
}
while(condition);
```

special programs :

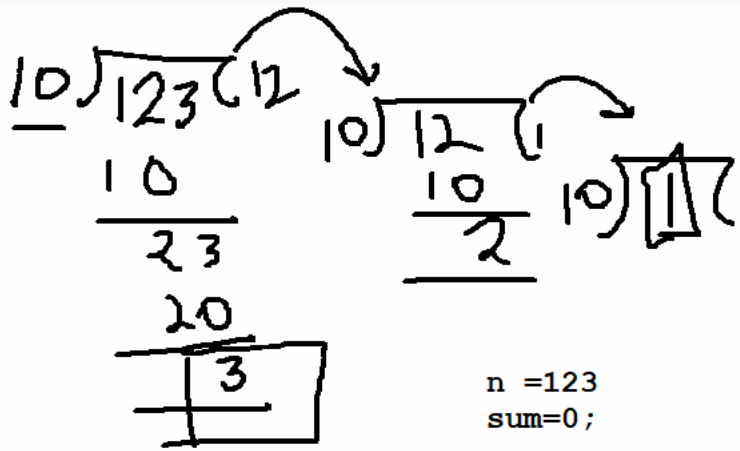
1. Sum of digits :

123

$\boxed{1} + \boxed{2} + \boxed{3} = 6$

153

1 + 5 + 3 = 9




```
n =123
sum=0;

r = n%10    =3
n=n/10 = 12
sum= sum+r =3
```

```
n=123

while(n>0)
{
r = n%10
n = n/10
sum = sum+r;
}
```

(12>0)

```
r = 12 %10 =2
n= n/10 =1
sum = 3+2 = 5
```

(0>0)

(1>0)

```
r = 1%10 = 1
n= 1/10 = 0._ =0
sum =5+1=6
```

sum of square of digits :

123

1 + 4 +9 =14

Armstrong numbers :

153

1 5 3 (cube)

1+ 125 +27

153

123

1 2 3
1 + 8+ 27=36

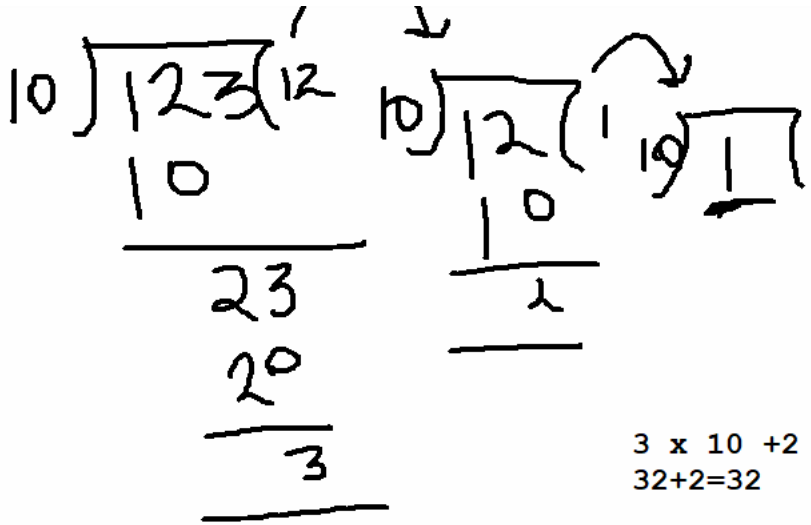
```
n= 153
sum=0
r
t=n

while(n>0)
{
r= n%10;
n= n/10;
sum= sum+r*r*r;
}

if(sum==t)
{
sopln("the no is armstrong")
}
else
{
sopln("no is not armstrong")
}
```

Reverse of number :

123

1 2 3

321




```
n=123
r, sum=0
while(n>0)
{
r=n%10;
n=n/10;
sum=(sum x 10)+r;

}

sopln(sum)
```

sum =0

sum=(sum x 10)+r

sum= (0 x 10)+3=3
sum= (3 x 10)+2= 30+2 =32
sum= (32 x 10)+1 = 320+1 321

3 x 10 +2
32+2=32

32x10 +1
320 +1= 321

palamdrome numbers :

n =131

1 3 1

131

```
n =212
t=n
sum=0, r
while(n>0)
{
r=n%10
n=n/10
sum=(sumX 10)+r;
}
sopln(sum);
if(sum==t)
{
sopln("no is palindrome");
}
else
{
sopln("no is not palindrome);
}
```

Array:

Array is a collection of elements and the elements are of similar type.
array is a container that holds data of one single type.

```
        0    1  2  3  4
a[5] = {10, 20,30,40,50}
```

types: one- dimentional array multi-dimentional array(2d array)

single- D array: it is an array in which each element accessed by using one index number.

syntax: datatype arrayname[];

```
datatype []arrayname;
```

```
ex: int a[] = {1,2,3,4,6}                int a[] = new int[10];
    int []a = {3,6,8,5}                  int []a= new int[7];
```

Two dimentional array: It is similar to one-D Array but It can have multiple rows and multiple columns.

```
1 2 3
4 7 9
3 4 6
```

syntax : `int a[][] = new int[3][3];` `int a[][] = {{1,2,3},{4,7,6}}`

cloning of Array:

1. cloning creates copies that are clones of the original elements or reference elements also.

2. cloning arrays are of two types: 1. shallow copy 2. deep copy

3. in a single D Array, a deep copy creates the clones of the original elements.

4. In multidimentional array, a shallow copy is created that means both arrays are pointing ro the same memory address.

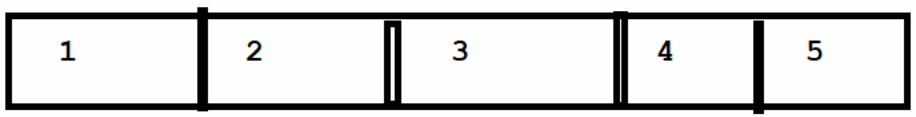
5. deep copy means a variable would have a copy of original array in a different memory location.

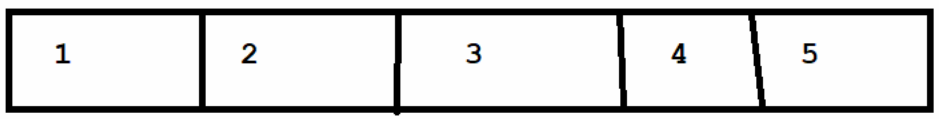
6. shalloew copy means both arrays are pointing to the same memory address.

7. in shallow copy if you modify one of these arrays, you will be modifying both arrays.

deep copy

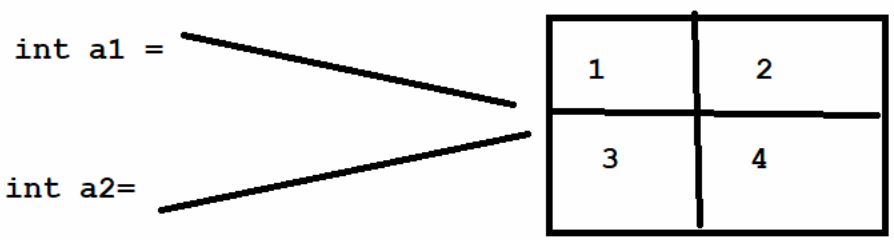
int a1=





int a2=

Shallow copy



String:

String is a sequence of characters enclosed with double quotes(" "). It is an immutable(not changable) object.

create a string:

1. String Literal String s="king";

2. Using new keyword String s= new String("king");

Exception handling :

1. An exception is an unwanted event that interupts the flow of your program..
2. Errors are generated while writing a programming code.
3. so these errors are displayed at compile time.
4. some of these errors not show up at compiletime but interrupts the normal flow of execution at run-time.
5. these errors are known as exception in your programs. and to handle those exceptions we use exception handling

1. It is an mechanism to handle runtime errors.

2. the main advantage of exception handling is to maintain the normal flow of the application.

3. All exception and errors types are sub classes of class Throwable.

4. suppose if an exception is not handled, it may lead to a system failure. that is why handling an exception is very important.

5. java provides specific keywords for exception handling like try, catch, throw, throws, finally.

try:

the try keyword is a block where we put exception code. the try block can not be used alone. A try block must be followed by catch blocks or finally block or both.

Syntax:

```
try
{
//statements which cause an exception
}
```

catch :

It must be used along with try block. it is used to handle the exception. multiple catch blocks are possible in java to handle multiple types of exception. the catch block catches the exception thrown by ztry block.

syntax:

```
 try
{
//statements which causes an exception
}
catch(exception(type) e(object))
{
//code for handling the error
}
```

finally :

finally block is used to execute the important code of the program. It will be executed irrespective of whether the exception is handled or not.

there should be only one finally block even if code is having multiple try-catch block.

syntax:

```
try
{
//statement which cause an exception
}
catch
{
//code for handling error
}
finally
{
//statement to be executed
}
```

final (value became constant/fix)	finally	finalize
1. keyword	1. block	1. method
2. use with: variable method class	2. use with either try or try-catch block	2. method is override for an object

throw:

the throw keyword used to throw a user define exception from a method or any block of code.

Syntax:

```
throw exception;
```

```
throw new ArithmeticException("divide by zero")
```

file handling

file:

the file class is the class that provides access to the file system to JVM.

file handling enables us to store the output of any particular program in a file and allows us to perform certain operations on it.

file handling means reading and writing data to a file.

```
import java.io.File;
import java.io.IOException;

public class FileExample
{
psvm(string args[]) throws IOException
{
File newfile = new File("nameoffile.txt");

if(newfile.createnewFile())
{
Sysout("file is created");
}
else
{
sysout("file is not created");
}
}
}
```

this program will create the file,nameoffile.txt under the current project directory, because we have given only file name as a parameter of the file class, which well be treated as a relative path by JVM.

the next line of code is checking the file presence of file with the same name in the system using the method createnewFile() that returns true or false .

based on the return of we are prining a text message on console. so in our case, the method returns true and create a file.

hence the cursor goes into the if section and prints the message.

Stream :

Stream is the sequence of data flowing from source to destination, Here source is called Input and destination is called Output. Input and Output streams support many data types such as character, string and object.

there are two types of streams: input and output and both implement byte and character type of streams to read and write the data.

BYTE byte streams are used to perform input and output of 8bit bytes. Classes such as FileInputStream and FileOutputStream support byte streams.

CHARACTER Character Streams are used to perform input and output of 16-bit Unicode charaters. Classes such as FileReader and FileWriter supports character streams.

input stream & file Reader :

input is trhe process to read the data from any source such as devioce, file, console, socket.

Input stream helps us to read the bytes coming from external sources.

Input stream is an abstract class and super class of all input classes.

FileReader is the class that helps to read the characters or text from file.

FileInputStream:

FileInputStream is a class that helps us to obtain input byte from file and build a connection between file stored in a storage and application

the syntax of FileInputStream :

```
FileInputStream(File fileObject)
        or

FileInputStream(String fileName)
```

output Stream:

Output stream writes data as an output into an array or any output device or file.

this is an abstract class and its sub-classes get implemented to generate the output.

Class that is part of outoput streams are FileOutputStream.

FileOutputStram

FileOutputStream is used to create a file in the file system and write data into that file.

if you try to write data into an existing file using FileOutoutStream and that named file doesn't exist in the file system then FileOutputStream will create a new fule with given namwe and add the data into it.

syntax:
OutputStream f= new FileOutputStream("path of the file or object of the file class")

To create a file using FileOutputStream, we need to pass an argument that will call the constructor and create the file. the argument can be an absolute or relative path of the file, or an object of File class.

InputStreamReader:

The InputStreamReader, a sub-class of the Reader class is a bridghe from byte streams to character streams.

this reads bytes and decodes those bytes into characters using specific charsets.

for ex:

InputStreamReader (InputStream in) is the declaration of InputStreamReader with default charset

InputStreamReader (InputStream in, String charset) is the declaration of InputStreamReader with given strings rtpe charset.

BufferReader:

The BufferReader class ia s subclass of the Reader class that reads text or buffering characters or character Input stream and provides efficient reading of characters, lines or arrays.

this can be used with FileReader or InputStreamReader or other reader classes to read the byte of inputs from file and return serialized or deserialized data.

Syntax:

```
BufferedReader bReader = new BufferedReader(readerObject);
```

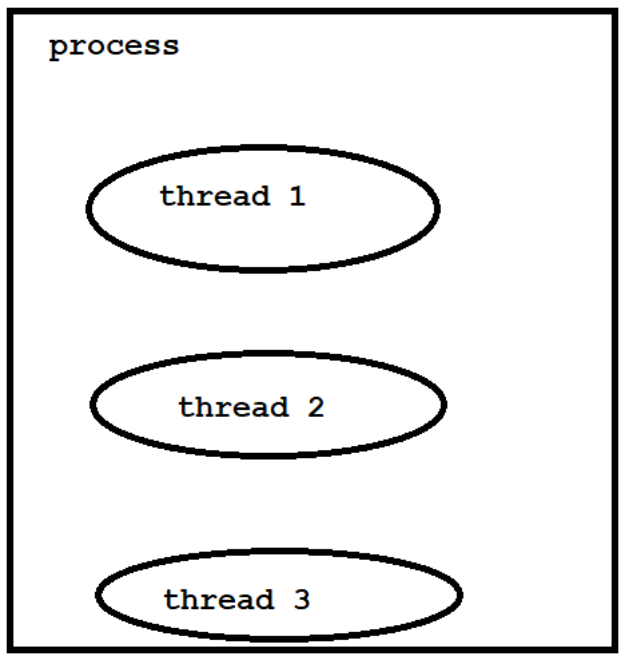
Multithreading:

thread:

thread is the smallest execution unit of process and a process may have many threads that are executing at the same time.

thread has its own execution path with in the process and shares the memory of the process with other threads, which are running in the same process.

thread doesn't allocate any memory, but it uses the memory allocated by its process; this helps faster and efficient communication between threads with in the same process.



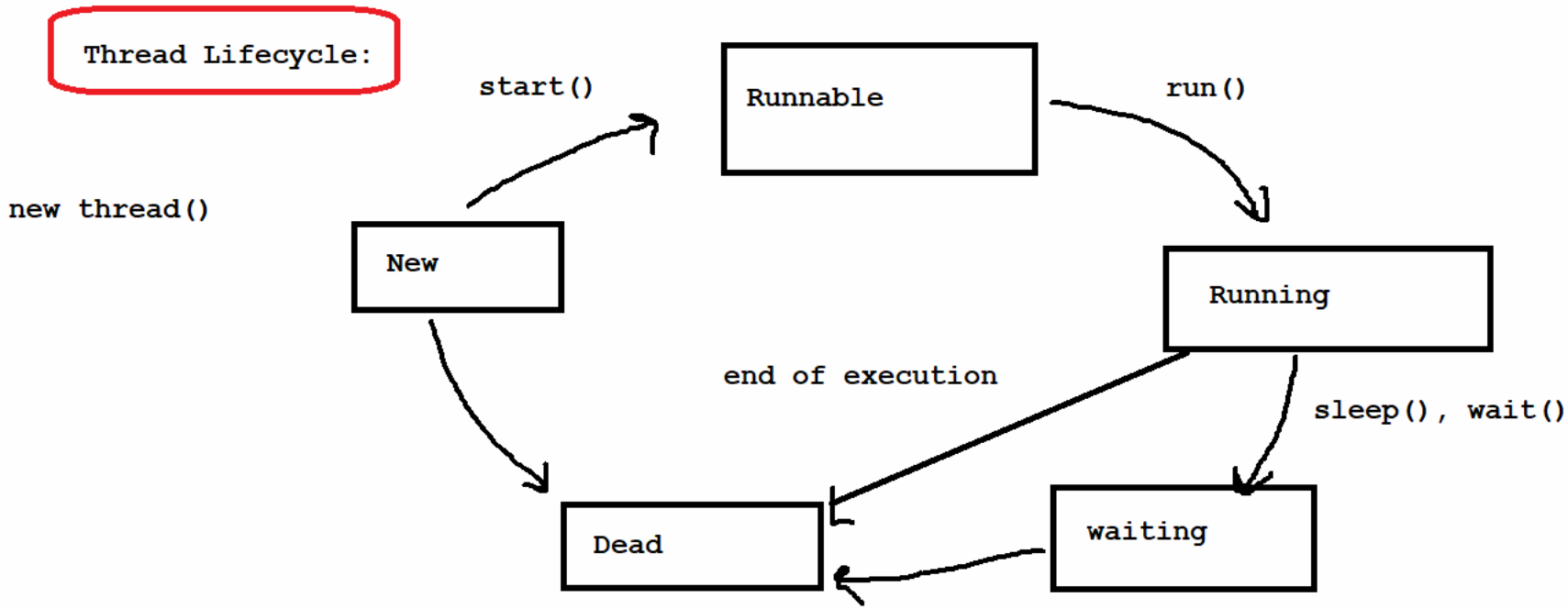
multithreading :

multithreading is a java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU

Each thread defines a seperate path of execution.

this means that a single program can perform two or more tasks simutaneously.

for ex: One thread is performing typing action on a file at the same time another thread is checking and resolving grammatical mistakes at the same time.


Thread Lifecycle:
start()
Runnable
run()
new thread()
New
Running
end of execution
sleep(), wait()
Dead
waiting

```
class Bookmyshow
{
int total_seats= 10;

public void bookseat(int seats)
{ synchronised (this){
if(total_seats>=seats)
{
Sysout("seats booked successfully");
total_seats=total_seats-seats;
}
else
{
Sysout("seats can not be booked");
Sysout("avaliable seats are"+total_seats);
}
}
}
```

```
public class thread_synchronisation extends Thread
{
static Bookmyshow b;  int seats;
public void run()
{
bookseat(seats);
}

psvm(String args[])
{
b = new Bookmyshow();

thread_synchronisation sumit = new thread_synchronisation();
sumit.seats=7;
sumit.start();

thread_synchronisation sonia = new thread_synchronisation();
sumit.seats=6;
sumit.start();
}
}
```

Java 8 features:

1. Lambda Expression

the new feature of java 8 is Lambra Expression. It is a function.

the function has no name, return type, or access modifier(private, public, protected)

Annonyms functions are also known as lambra expression.

it is avaliable in other programing languages like python. Ruby, c# etc.

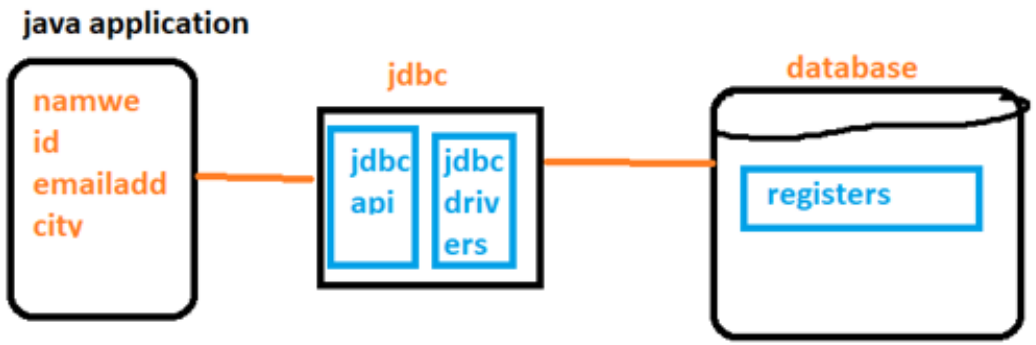
It is used to provide functional interface implementation.

A function that can be written independently of any class.

It allows you to iterate over a collection, filter it and retrieve data.

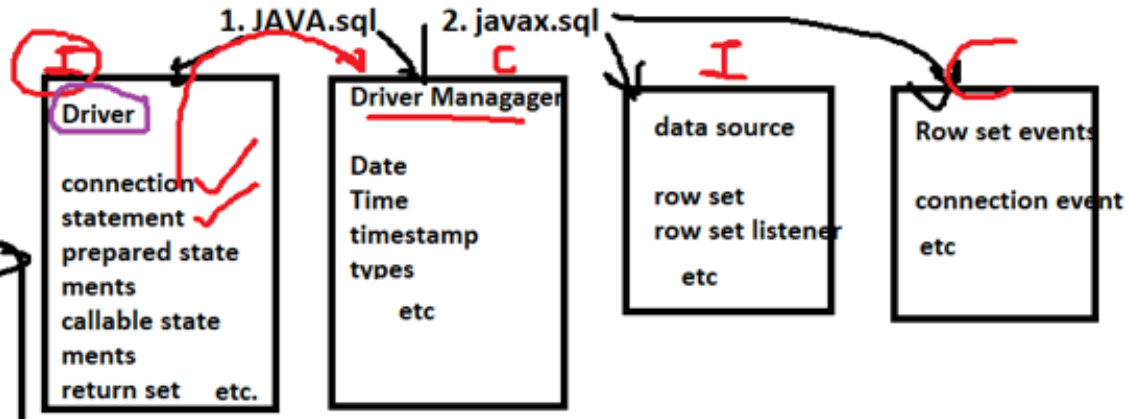
JDBC

1. Java Database Connectivity
2. JDBC is a technologgy which is used to connect java application with Database.



3. JDBC is an API which provides some classes a& interfaces by which we can connect Java Application with database.

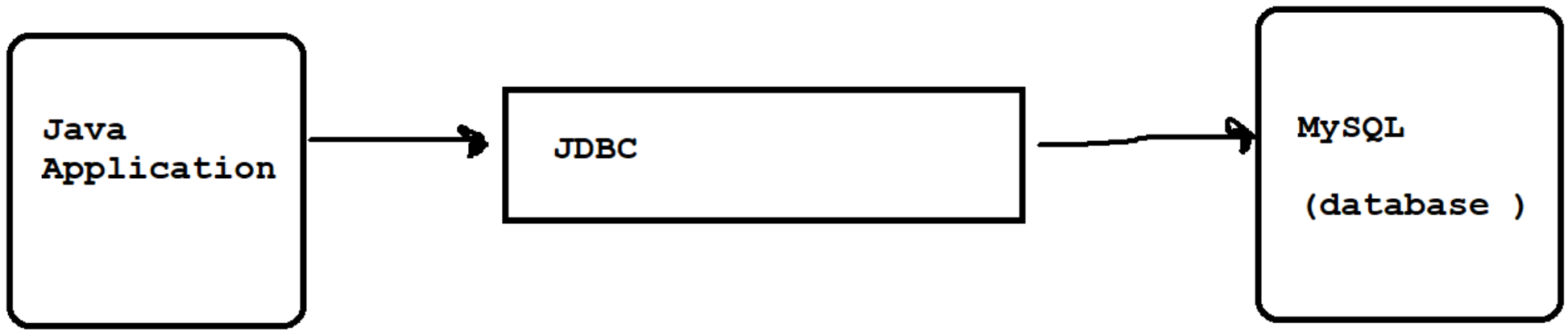
4. JDBC API has two packages:



5. JDBC is an abstraction which is provided by Sun Microsystems & implemented by Database vendors.

JDBC (Java Database connectivity)

It is a standard API provided by oracle for java applications to interact with different set of databases.



why JDBC

```
class Test
{
pubic static void main(String[] args)
{
String name= "john";
int age = 40;
double salary= 80000;

Sysout(name);
sysout(age);
sysout(salary);

}
```

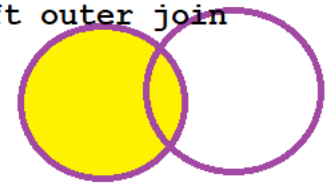
JDBC API

important classes and interfaces

java.sql.DriverManager
java.sql.Connection
java.sql.statement
java.sql.preparedStatement
java.sql.callableStatement
java.sql.ResultSet
java.sql.ResultSetMetaData
java.sql.DatabaseMetada
java.sqlException

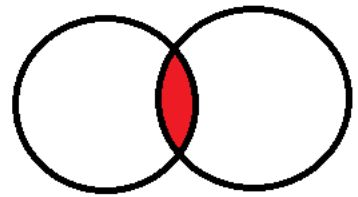
left outer join

select<fields> from
table A left join
table B
on A.kev = B.kev



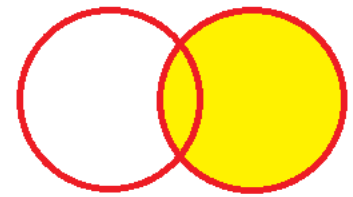
inner join

select <fields> from
tableA inner join
tableB
on A.key= B.key

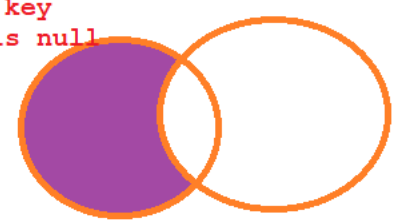


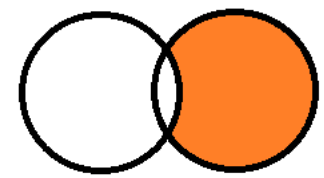
right outer join

select <fields> from
Table A rightjoin
Table B
on A.key =B.key



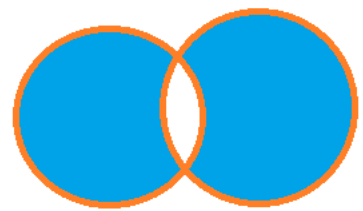
select<fields> from
table A left join
table B
on A.key = B.key
where B.key is null

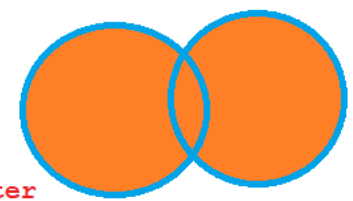




select<fields> from
table A join table B
on A.key =B.key
where a.key is null

full outer join





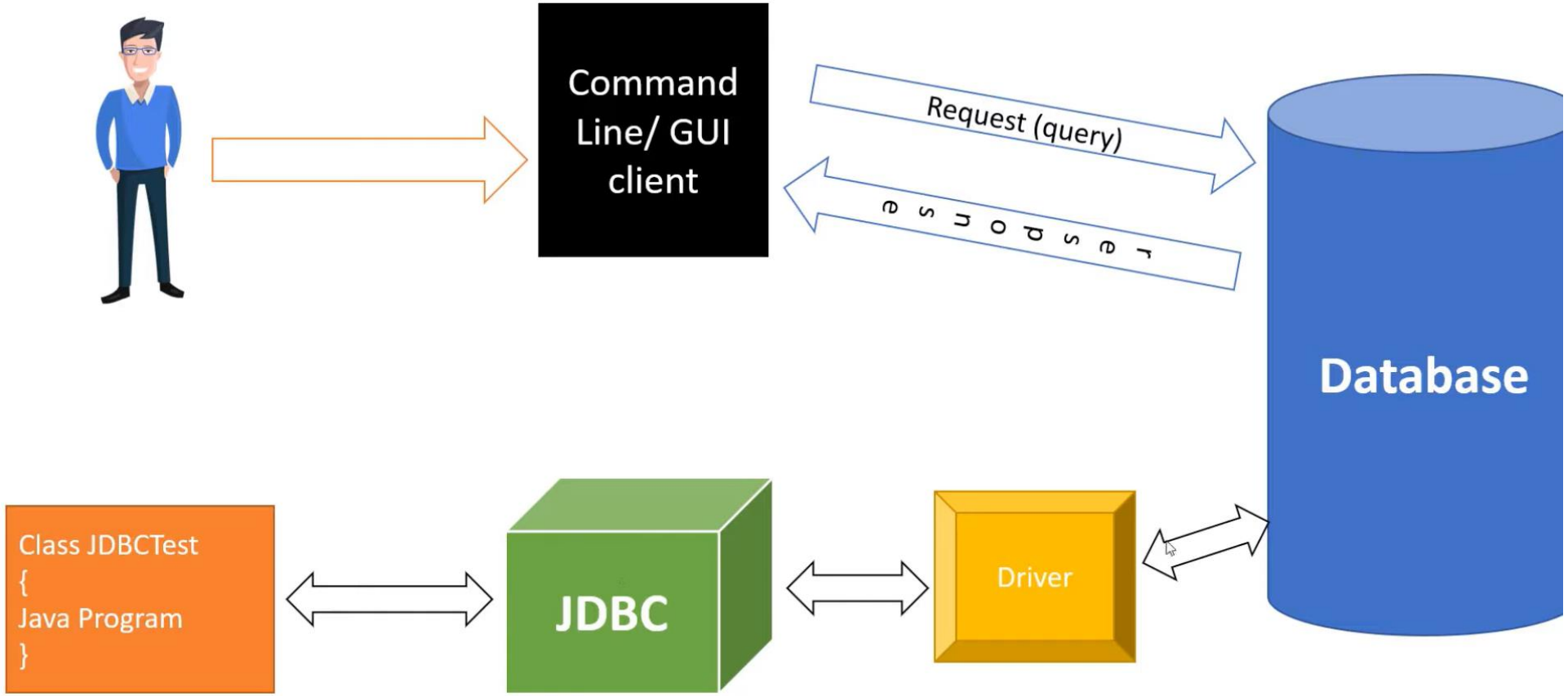
select <fields> from table A full outer
join table B on A.key= B.key

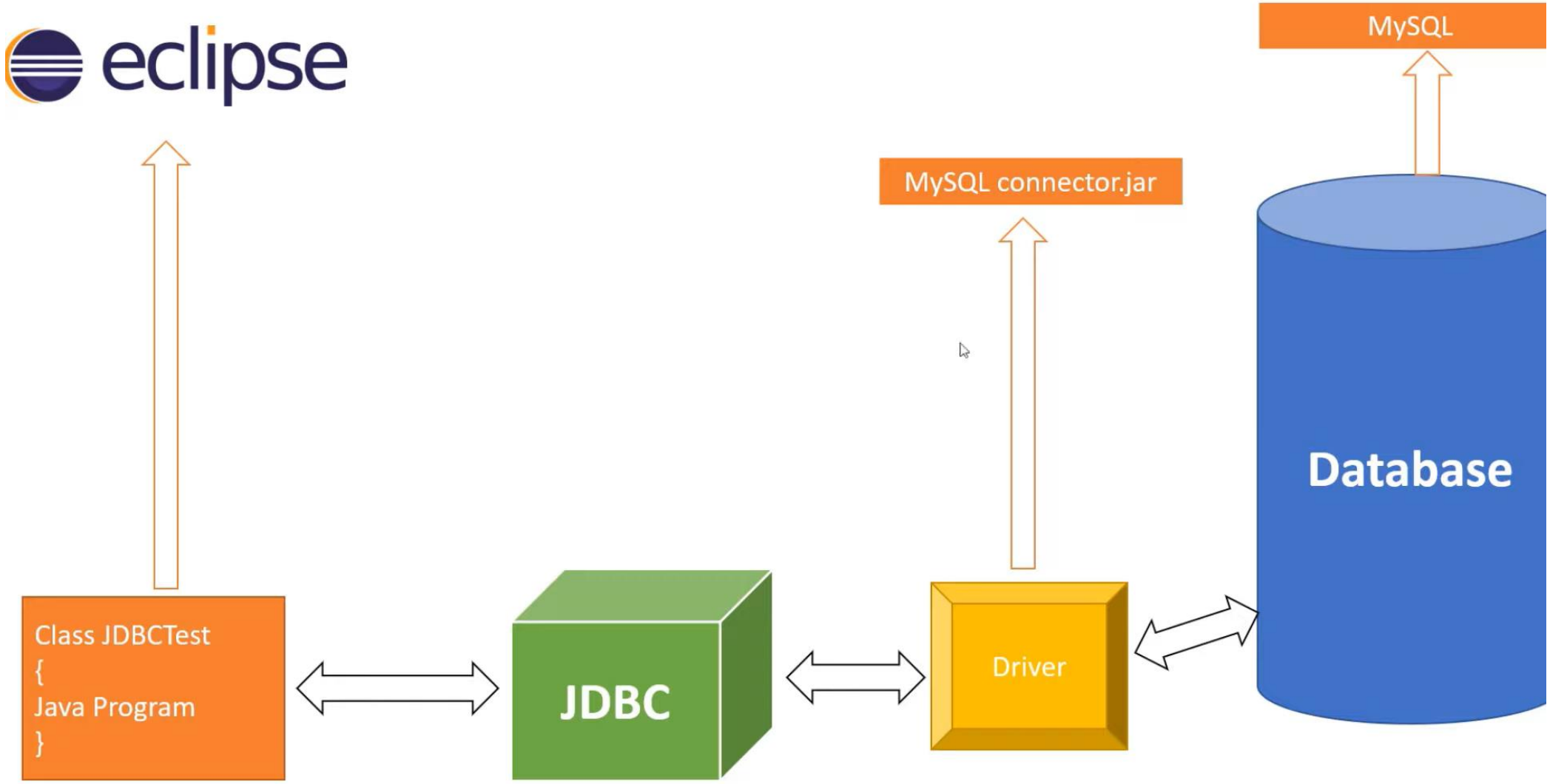
select <fields> from Table A outer join table B
on A.key =B.key where A.key is null or b.key is null

What is JDBC ?

- JDBC stands for Java Database Connectivity .
- JDBC is a Java API to connect and perform operations(insert , delete , update , select , etc) with the database.
- JDBC API uses JDBC drivers to connect with the database.

JDBC Working




eclipse
MySQL
MySQL connector.jar
Database
Class JDBCTest
{
Java Program
}
JDBC
Driver

How to create connection

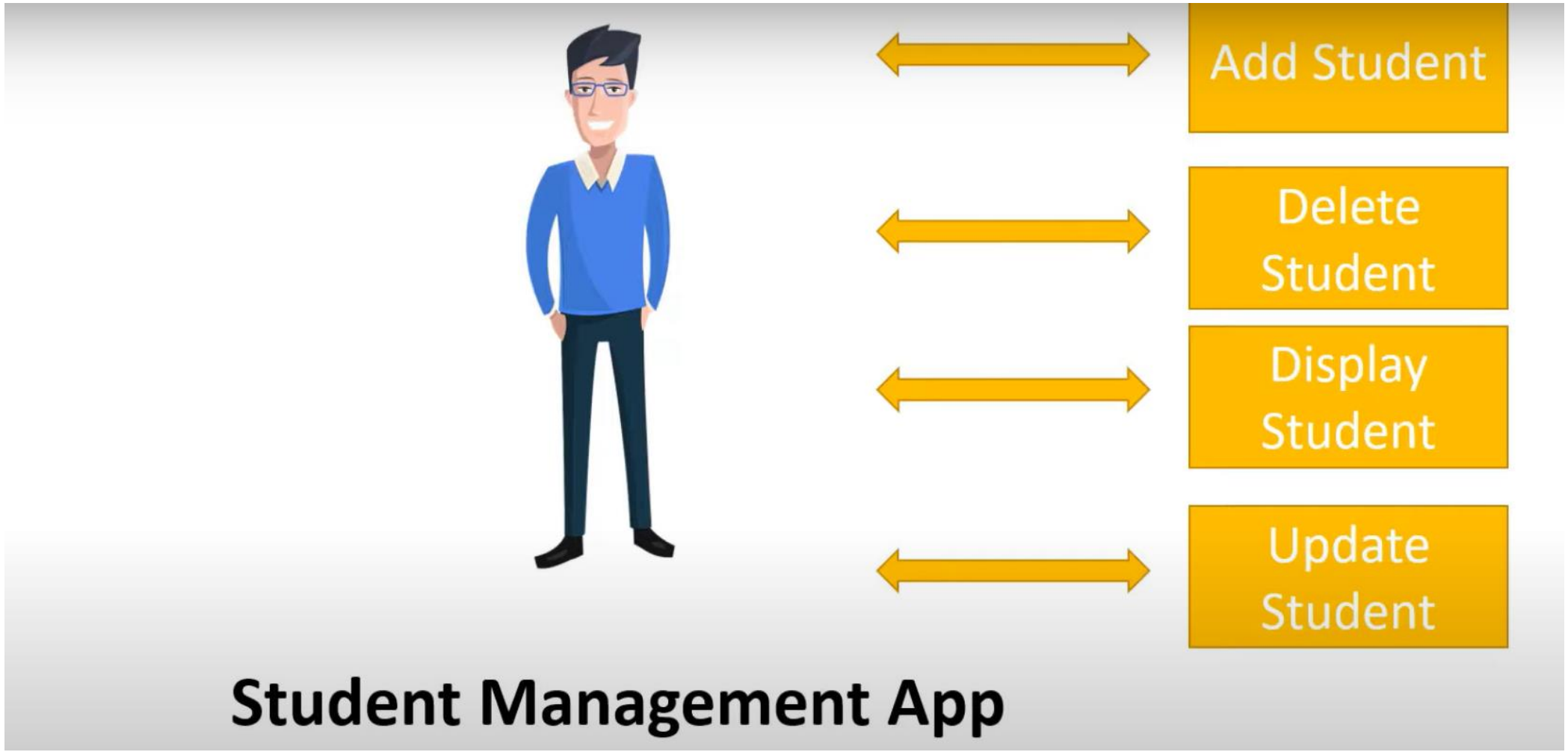
- To connect Java application with the MySQL database, we need to follow 5 following step
- *Load the driver class*.
- *Create Connection using DriverManager.*
- *Use connection to fire queries{ Statement for static queries and PreparedStatement for dynamic queries }*

select * from table

select * from table where col=?

How to create connection

- To connect Java application with the MySQL database, we need to follow 5 following step
- *Load the driver class.*
- *Create Connection using DriverManager.*
- *Use connection to fire queries{ Statement for static queries and PreparedStatement for dynamic queries }*
- *Process the result .* ResultSet
- *Close the connection.*


Add Student
Delete Student
Display Student
Update Student
Student Management App