Computer Basics

Computer:

Computer is an integrated set of algorithm and structures that are capable of storing, retriving and manipulating information according to the set of given instruction.

Code Representation

i) Pseudo P Code:

Pseudo code is a step by step written outline of your code that can gradually transcribe into the programming language.

→ It is a readable description of what a computer program should do.

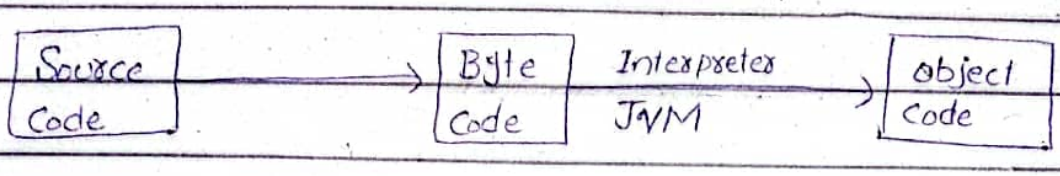
ii) Flow Chart:

Flow chart is graphical representation of what a compiler program should do.

iii) Source Code:

Source Code is the actual code in any programming language.

Why Java compiles code is hardware Friendly?



Sequential Programming

→ Instructions are executed one after another in a specific order.

→ Function Oriented Paradyme

→ C , C++ , Java

Function:

- A named group of instruction set.

1. Function Definition
2. Function Call
3. Function Prototype.

(i) Function : Definition:

Function definition has further two parts:

a) Function Header
- Return Data Type
- Function identifier
- List of Parameters

b) Function Body

return type → Function identifier → List of parameters.

```
int sum (int a , int b){
    return a + b;
}
```

Data:

The raw information is called data.

Information:

The meaningful data after processing is called Information.

Types of Data:

- Numeric
- Alphanumeric
- Boolean
- String
- Void

(i) Numeric data

- Whole
  - Byte (1 byte)
  - Short (2 bytes)
  - Int (4 bytes)
  - Long (8 bytes)
- Decimal
  - Float (4 bytes)
  - Double (8 bytes)

(ii) Boolean

- It returns two values 'True' or 'False'.

(iii) String:

- Anything in double quotes.

(iv) Void:

"null"

Constant in Java:

'final' keyword is used in java to declare a constant.

```java
final int PI = 3.14;
```

→ PI is now unchangeable.

→ "ROLL_NO". This naming convention is used for constant.

Types of Constant:

1) Literal Constant:

• وہ Constant جسے دیکھ کر compiler یہ بتا سکے کہ یہ کونسی زبان ہے۔ جیسے: 19 ، "a"

Me.

2) Memory Constant:

• Literal Constant کی value جس میں store کروائی جائے اسے memory constant کہتے ہیں۔

• اوپر والی example میں 'PI' ایک memory const. ہے۔ جبکہ 3.14 literal const. ہے۔

☆ Programming میں best approach یہ ہے کہ literal const. کو صرف memory const. کو define کروانے کے لیے استعمال کروایا جائے۔ اسے direct use نہ کیا جائے۔

Types of Parameters:

1) Formal Parameters:

• وہ Parameters جس میں memory const. کو use کیا جائے
اسے Formal param. کہتے ہیں۔
• یہ Func. definition میں use ہوتے ہیں۔

```
int sum (int a, int b){
    return a + b;
}
```

2) Actual Parameters:

• وہ Parameters جس میں literal const. کو use کیا جائے
اسے Actual Parameter کہتے ہیں۔
• یہ Func. Calling میں use ہوتے ہیں۔

```
sum (10, 5);
```

Types of Errors:

1) Syntax Error:

Every programming language has predefined grammar called syntax. So error in this syntax is called syntax error.

2) Logical Error:

• وہ error جو language کے اعتبار سے بالکل

غلطیاں جو کہ programmer نے غلط logic کو use کیا ہو جن
کی وجہ سے result غلط آئے۔ اسے logical error کہتے ہیں۔

3) Runtime Error:

← Compiler پہ error OK. ← runtime پر آئے اسے
runtime err. کہتے ہیں۔

→ Runtime errors are not caught by the Java compiler. They are caught by JVM.

- Insufficient runtime resources
- Input-output errors.

Type Casting:

(i) Explicit Casting:

- The conversion of larger data type to smaller data type is explicit casting.
- Programmer Forcefully changes its data type.

```
double d = 50.68;          // 8byte
int a = 10;                // 4byte
Implicit → int c = a * d;   }
Explicit → int c = (int) d; }  casted
```

Scanned with CamScanner

(iii) Implicit Casting:

- Casting performed by compiler automatically not forcefully is called implicit casting.

Function Prototyping:

C++ Code

```
# include <iostream>
void foo();
int sum(int x, int y);          > function Prototype

int main(){
    int result = sum(20, 30);
    foo();
    std::cout << result;
    return 0;
}

void foo(){
    sum(50, 40);
}
int sum(int x, int y){
    return x+y;
}
```

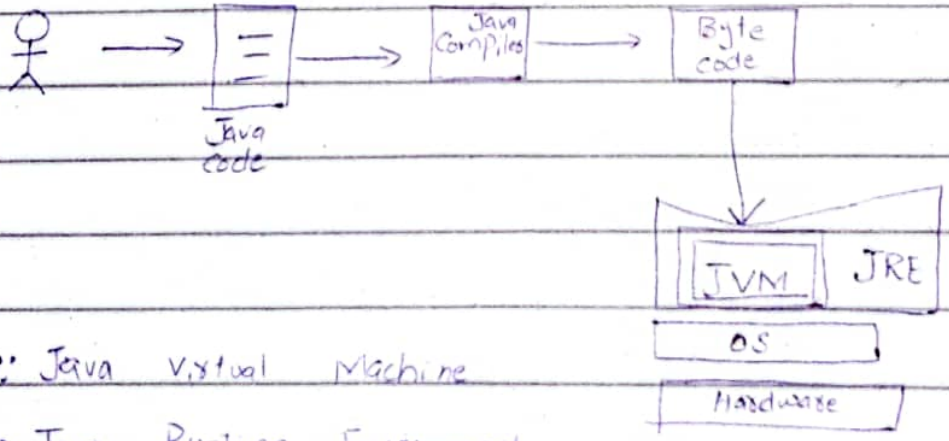
• A function prototype is a declation of the function that informs the program about the number & kind of ~~operators~~ parameters, as well as the type of value, the function will return.

How Java Compiler works?

• Java is a platform independent language due to JVM. Platform independent means it can run on any machine OS having JVM.

• Java application is platform independent; JVM is not platform independent.

• JVM will accept byte code.



∵ Java Virtual Machine

∵ Java Runtime Environment.

- The execution of java code will start from the main() method.

- So when you enter Java code → It is compiled to Byte code by Java Compiler → This byte code will go to JVM → JVM will start execution from main() method.

- When you create & compile a Student.Java file which is Java code, a new file Student.class will be created which is a Byte code.

- JVM runs in a Java Runtime Environment.

Arguments:

In arguments we only specify the datatype and sequence of data types in the parenthesis of a function prototype. We don't need to specify the identifier.

Parameters:

In parameters we need to specify the dataType, sequence & identifier.

JavaScript ← dynamic ایک ہے جس سے ← large
اس کے code میں بہت زیادہ bugs آتے ہیں۔
JS ← Node JS نے JS کے نام سے ایک library بنائی جو
server side کے لیے کام کرتی ہے۔ پہلے صرف browser کے لیے ہوتی تھی۔
Web Assembly ← Java کا Byte Code ہے۔
MSIL ← C# کا Intermediate Code ہے۔

Object Oriented Programming (OOP)

Entity: Anything which has value is entity.

- Any person, place, object or concept for which an organization wants to record or process data is "entity."
- An org. want to keep record of any "product" like "students" in IUB. (student)

Attribute:

- Characterestic of an entity for which an organization wants to record or process data is "Attribute" (roll no)

Operation/ Behavior:

- Function which an entity perform is "behavior" or "operation" (admission).

- ہم ریکارڈ وہ لیتے ہے کسی آبجیکٹ کو وہ جو کی نکل پاتے ہیں۔

Class:

- Class derived from classification.
  (1) Classifying programming information into organized & systematic manner.

ایک entity کے متعلق جتنے بھی behavioral attribute کو ایک الگ کلاس میں رکھ لیتے ہیں۔

- In OOP entity will be "class", attributes are data members, function is behavior.

```
class student {  ———— Entity (class)
    int rollno;  ——\
    float fee ;  ——/  Attributes (Data members.)
    void input(){ ——\
    }                 > Behavior .(functions)
    void output{ ——/
    }
}
```

(2) Class is a written description of an object. OR template of an object.

• جیسے درزی (کپڑوں کا) ماپ لکھتا ہے تو وہ "ماپ" class ہے اور سوٹ "object" ہوگا۔

• Physical existance of a class is "object".

(3) Class is an abstract data type (ADT)

(ADT)
↳ Declaration of data.
↳ Declaration of operation.
↳ Encapsulation of data & operation.

C is not ~~a~~ ADT. C++ is (ADT).

• So we can declare data & operation in the same class. in Java & C++ but not in C. That's how we classify the programming inform.

~~of~~ an

• اگر کمپائلر کو پہلے سے اس ویریبل کی data type Primitive ہے تو وہ
non-primitive ہے۔ اس کے برعکس data type ہے۔

• Class Library Developes & App Developer

• "Class library developes" is the one who make classes.

• "App developer" is the one who use the classes and develop apps.

• Data Encapsulation & Data Abstraction.

• Data hiding & Data abstraction is called "Data Encapsulation."
where "Data Abstraction" is the condition on data.

• بغیر hiding کے Abstraction ہو سکتی ہے۔ اور بغیر Abstraction کے hiding نہیں ہو سکتی۔

• In Data Encapsulation we don't need to get knowledge about the backend.

• And we don't need to check everytime.

Member Access Modifies.

(1) Member Access Modifies.

(2) Member Access Specifies

1. Private:

Private access modifies can only be accessed within the class & cannot be accessed outside the class.

Important Terms.

- Collection of classes is called "library or Package".
- Collection of Packages is "Framework".
- Collection of simulator + packages + Framework or other than Packages + Framework is called SDK (Software developmend kit).
- SDK of Java is JDK (Java. D.K)
- Repository is the place where to store project files.

Package Details:

- It is not recommended to use default package. Package name should be unique overall the world.
- "plc.org.cas.firstdemo" is the method to find unique package name.

2. Default:

- Default access modifier member is accessible within the same package but not outside the package.

3. Protected:

- Protected access modifier member is accessible outside the package for child class only and within the package for every class.

4. Public:

- Public access modifier is accessible everywhere.

Data Encapsulation.

- As rollno cannot be less than 0. So to check the rollno & store in the variable is data abstraction.
- To hide its value is data hiding.
- Combination of both data hiding & data abstraction is Data Encapsulation.

getter/setter

- Private کے members کو ہم getter/setter functions کے ذریعے access کرتے ہیں اور اس میں value کو hide کرتے ہیں۔ اس طرح ہم data کو encapsulate کریں گے اور Abstraction بھی apply کریں گے

```
private rollno;
public void setRollNO (int n){
    if (n >0){
        rollno = n
    }
    else{
        System.out.println("Invalid fee");
    }
}
```

Main

```
Student.setRollNo
(-10);
```

```
private sting address;
public void setaddress(string add){
    address = add;
}
```

Indirect Abstraction:

- اگرچہ یہاں پر ہم string پر کوئی condition نہیں لگا رہے لیکن سارے variables کو ایک طریقے سے access کرنے کے لیے ہم نے setter اور getter کو use کیا۔

- Encapsulate کیا جا سکتا ہے Function member اور Data member کو ایک ہی class کو۔

- ہم same scope پے same identifiers نہیں کر سکتے۔

- اگر ایک scope کے اندر کے same identifiers کے different scope ہوں تو اسے shadowing کہتے ہیں۔

"this" keyword

```
private int teacherId;
private
```

```
public int getTeacherID(int teacherId){
    this.teacherId = teacherId
}
```

private int id;

Setter (mutators) :-

```
public void setId(int id){
    this.id = id;
}
```

Getter (accessors) :-

```
public int getId(){
    return id;
}
```

Constructor

- Constructors are the special member function of the class

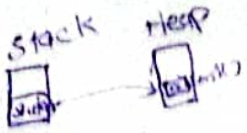
1) Constructor identifier always be a class identifier.
2) Constructors don't have explicit return type. & we cannot explicitly return a value from Constructor.
3) Constructors can't be called like normal member function. They are always be called with "new" keyword.
4) Constructor returns implicitly / automatically.
5) When a constructor gets called its object is created in mem

Constructor → Student student = new Student( );

Normal func. → student.input( );

objectNam

6) Constructor returns implicitly the address of object to stack.



Types of Data Types:-

- Value Type
- Reference Type

- "stack" میموری dedicated اپنی اپنی App کو۔
- "Heap" ایریا common Apps کا۔

Value Type:-

8byte

All primitive data types except ~~heap~~ string stores in Stack. It consumes limited space. (int a;)

Reference Type:-

String & all classes are ~~reference~~ stores in ~~type~~ Heap. These types can consume very large spaces.

e.g Student student = new Student( );
↓ in Stack ↓ In Heap

e.g: String, class, Arrays, interface are reference types.

Examples of Value Type:
byte, short, int, long, float, double, char, boolean.

- Reference of reference type will be store in stack.

Use of Constructor:

- Constructors are used to assign values to variable auto. when we create object.

- Android OS maintains user record is stack.
- Reference:-
    - Base address of object. (starting address)
    - Base address of member.
    - Meta data about member.

• اگر Constructor کا نام class سے same نہ ہوتا تو ہم constructor
کو call کرتے وقت کنفیوز ہوتے کہ "یہ کلاس" "class" کا "object" ہے۔

Types of Constructors.

There are two types of constructors:-

1) Implicit Constructor (جو کمپائلر کرتا ہے)
2) Explicit Constructor

Implicit Constructor:

- Constructor created by compiler which will be empty. It will be created only if its class is constructor-less. It will be default. Default constructor has no parameter.

Explicit Constructor:

A constructor which is created explicitly OR by programmer is explicit constructor. Its types are following:

- Parameterized Constructor.
- Default Constructor.

Poly morphism
many — Physical states

Types

1. Parameterized Constructor:

A constructor which has parameters.

```
public Student (int rollNo);
```

2. Default Constructor:

A constructor which has no parameter.

```
public Student ( );
```

★ many ← **Poly morphism.** → Physical States

Function ← Polymorphism کی دو طرح سے ہوتی ہے۔

- Function overloading.
- Function overriding.

Function Overloading: (Adhoc Polymorphism)

• ایک ہی نام کے multiple فنکشنز ہوں گے تو اسے ہم Function Overloading کہتے ہیں لیکن اس کے signature مختلف ہوں گے۔

Overloading Signature:

- No of parameter
- DataType of parameter
- Sequence of parameter.

★ • اگر functions کی return type مختلف ہو تو کیا وہ function overloading ہے؟

• Java کی زیادہ تر libraries کو C لینگویج میں لکھا گیا ہے۔

• Functional prog. میں ہم ایک نام کے ایک زائد فنکشن نہیں بنا سکتے۔

```
Public Student(int r){ rollNo = r}
```

Type of constructor:

3. Copy Constructor:

• اگر Parameter میں اسی class کا object آئے تو اسے copy constructor کہتے ہیں۔

```
int rollNo;
public Student (Student s){
    rollNo = s.rollNo;
}

Student st = new Student(3);
Student st1 = new Student(st);
```

Note:

• C++ میں ہمیں object بنا کر میموری فری کرنے کے لیے اسے delete کرنا ہوگا۔ نہیں تو وہ memory کنزیوم رہے گی۔

• Java میں JVM کا Garbage Collector اسے clean کرے گا۔ Garb. Colle اسے بعد میں clean کرے گا۔ (جیسے گھر میں صفائی والی صفائی کرتی ہے)

Garbage Collector اس وقت صفائی کرے گا جب New object کے لیے memory میں جگہ نہیں ہوگی۔

• Android کی libraries کو core Java میں code لیا گیا ہے

LLVM: (صرف یہ تین Non GC ہیں)

① C ② C++ ③ Rust

میموری کے اعتبار سے انتہائی safe ہے۔

Static & Non-Static Members.

Instance Member Non-Static Non-Shared	Non-Instance member Static Shared
↳ RollNo, Name, Email	↳ Rules, LCD, Gas Wifi
↳ Implicitly every member is non-static.	↳ We need to declare explicitly a memb. static. with keyword "static".
↳ یہ کلاس کے object والے Non-Static ممبرز ہوتے ہیں۔	↳ یہ کلاس کے class والے Static ممبرز ہوتے ہیں۔
↳ Student.getCount(); ~~class~~ Object Name	↳ Student.getName(); class Name

Naming Convention:

- Naming Convention is the best practices used by programmers to name variable.

→ PascalCase:

- GrossSalary
- Used for class names

class (static)
Student.setRollNo();

→ camelCase:

- rollNo
- used for variables

object (Non static)
student.setName();

• جب کوئی class لوڈ ہوتی ہے تو compiler اس کے static members کو memory میں initialize کرتا ہے (constructor کال کرتے ہیں)

• جب ہم object بناتے ہیں تو اس وقت compiler اس کے non-static members کو memory میں initialize کرتا ہے۔

Static Block

```
static{
    sout("static block is called");
}
```

• جیسے ہی compiler کلاس کو دیکھے تو وہ سب سے پہلے اس کے static members کو initilize کرتا ہے۔ اوپر والا code سب سے پہلے کال ہوگا۔

• جب ہم کسی بڑے code کو کم lines میں لکھتے ہیں تو اسے syntactical sugar کہتے ہیں۔

• ہم multiple static blocks کو call کر سکتے ہیں۔

"this" keyword:

• "this" ہمیشہ non-static members میں یوز ہوگا۔ یہ ایک reference var ہے۔ اس کے پاس اس object کا address ہوگا جس میں class کے لیے یہ یوز ہو رہا ہے

• "this" کا تعلق object کے ساتھ ہے اور static members کا تعلق class کے ساتھ ہے۔ اس لیے static میتھڈ میں "this" keyword یوز نہیں ہو سکتا۔

• static blocks میں non-static members کو access نہیں جا سکتا جبکہ non-static members میں static members کو call کیا جا سکتا ہے۔

Anatomy of main( ) function.

- ہماری Application کا point of entry (Main func.) ہے۔
- Main کا Function ہم کسی بھی (class) Top level میں بنا سکتے ہیں۔
جس کا class name اور File name ← same ہوں۔

- public بتا رہی ہے کہ یہ Function outside the class ← accessible ہے۔
- Static بتا رہا ہے کہ اس Function کا تعلق class کے ساتھ ہے object کے ساتھ نہیں ہے۔
- String[] args ← backend compatibility of کے لیے ہوئی ہے۔

```
public static void main(string[] args){ }
```

* • ہم صرف Top level class کو ہی public رکھ سکتے ہیں۔
* Non Top level class ضروری Default ہی ہوتی ہے۔
- Java اور C# میں تمام code کو class میں لکھا جاتا ہے۔ کوئی code ← class کے باہر نہیں ہوتا۔
- class کے صرف دو ہی access modifiers ہیں - public & default.
private & protected ← class کے members کے access modifiers ہیں۔

Package:

- package related classes کو group کرتا ہے۔ جیسا IUB کا IT اور transportation کا package الگ الگ ہے۔
- package ہم Command سے بھی بنا سکتے ہیں۔

- Top level class کو ہم public یا default رکھ سکتے ہیں۔
- Non Top level class صرف default ہو گی۔

Package Name:

- package name ہمیشہ unique (worldwide) ہوتے ہیں۔ package default ہم
- میں کوئی سزا نہیں بتائی گئی لیکن یہ best practice نہیں ہے۔
- package name کو unique رکھنے کا طریقہ یہ ہے کہ reverse domain name dot کے ساتھ package name رکھا جائے۔

```
package pk.org.cas.demo1;
```

- Make sure to make explicit packages & build classes in it.

Non Object Oriented Programming (non-OOP)

System . out . println ( );

- out → non-static func. of out.
- System → static member of System class
- out → print stream (class) → printStream
- println() is an overloading Function of diff sign.

- A function output shows in console/Command Line Interface/ CLI.

Scanner (Taking input from user):

```
import java.util.Scanner;
Scanner scanner = new Scanner(System.in);
                  Input from CLI
rollNo = scanner.nextInt( );
Sout ("The Roll No is:" + rollNo);
```

This rollNo is concatenated to string by using plus sign between str & int.

System. out . println ( );

- System is a prebuilt class.
- Static reference variable of a class of type printStream.
- println() is the function of the class for which reference variable is static

String Input:

- name = scanner.next();
  // It will stop taking input as "space" comes in a string.
- name = scanner.nextLine();
  // It will take complete string as input.

1. اگر float کو پہلے read کریں اور String کو بعد میں read کریں تو scanner کلاس String کو skip کر دیتا ہے۔ اس کا حل order change ہے۔

Anonymous approach.

① character = scanner.next().charAt(0);

↓ Anonymous technique ہم اس وقت use کرتے ہیں جب ہم نے function کو one time یوز کرنا ہے۔

② Sout(new Scanner(System.in).nextInt());

Book: The complete reference of Java

- "has" یا "is" جس function کے شروع میں آتا ہے وہ boolean کو return کرتا ہے۔ جیسے: hasNextInt() ⟸ چیک کرے گا کہ Input stream میں Int ہے یا نہیں۔

Operators

• Operator ایک symbol ہوتا ہے جو کسی compiler کو guide کرتا ہے operation کے متعلق۔

- Unary operator
- Binary operator
- Ternary operator

Unary Operator:

• Unary operator works on single operand.

-a , -12 , ++a

Binary Operator:

• Binary operator works on two operands.

a - b
operands

Ternary Operator:

• condition ? true statement : false statement

?:

It works on three operands.

• Ternary operator ~~ہے~~ is only one which is Conditional operator.

★ Side Effect of Operator:

• اگر operator کو use کرنے سے اس کے operand کی value change ہو جائے تو اس operator کا side effect ہے۔

a = 45;

++a;

a++

Precedence of operator:

- اگر ایک expression میں multiple operators آ جاتے ہیں تو compiler پہلے کس operator کو resolve کرتا ہے۔

* Expression:

- The combination of operator & operand which evaluate to a single value is called Expression.

1. Arithematic Operator:

- * / % & + −
  High-Precedence Low-Precedence.

Paranthetical operator:

(→ Paranthesis

{ → Braces

[→ Brackets

- Compiler has highest precedence of ( ).

Associativity of operator.

- جب ایک expression میں multiple operators ہوں اور ان کی precedence same ہو تو compiler پہلے کس side سے solve کرے گا۔
- Associativity of arithematic operator is left to right.
- Associativity of assignment operator is right to left.

2) Postfix Unary Operator:

i = 10 — high precedence

a = i++ + 5 — ① 15

a = 15 (//i = 11) — ② 15

2.i- Prefix Unary Operator:

i = 10 — high precedence.

a = ++i + 5 — ① 11

a = 16 ; (i = 11) — ② 16

3- Relational operators:

{ == , != , < , > , <= , >= }

- The result of relational operator will be boolean.
- Discovered by George Boll.

☆ . Condition:

ایسا Expression جو Boolean val. ← return کرے وہ Condition کہلاتا ہے۔

- All conditions are expressions but all expressions are not conditions.

Relational operator کے لیے دونوں operands کا same ← data type ہونا لازمی ہے۔

4- Logical operators:

- isFound → Boolean variable (Naming convent.)
- Logical operators ہمیشہ Boolean operands کے ساتھ use ہوتے ہیں۔

→ ! (Not) → unary

→ && (AND) → binary

→ || (OR) → binary.

- multiple conditions کو combine کرنے کے لیے (&& ، ||) یوز ہوتے ہیں۔
- Compiler کے لیے سب سے complex time conditions ہیں۔
- اس لیے condition کو جہاں ضرورت ہو وہیں یوز کریں۔

- Short-Circuiting:

- And op. کے ساتھ اگر پہلی condition ← False ہو جائے تو compiler دوسری condition ← check نہیں کرتا۔
- OR operator کے ساتھ اگر پہلی condition ← True ہو جائے تو compiler دوسری condition چیک نہیں کرتا۔

5- Assignment Operators:

=, +=, -=, %=, /=, --=
&=, |=, ~=, <<=, >>=, >>>= } 12

- All operators except = are compound assignment operator.

b = c % b

b %= b

☆ Statement:

- اگر expression کچھ return نہیں کرے گا تو وہ statement ہے۔

Student student = new Student(5); → operands

(5) → paranthytical operator

- جو constructor کچھ return نہیں کرتا اس لیے یہ statement ہے

System.out.println(10);

10 اور println - ہے call operator() - ہے identifier ایک println (func.)

کچھ بھی یہ کیونکہ ہے statement یہ لیے اس - ہیں operands
return نہیں کر رہا۔

· Number Systems:

(1) Binary (0–1) 1 = 1
(2) Octal (0–7) 1 1 1 = 7 (4 2 1)
(3) Decimal (0–9)
(4) Hexadecimal (0–9, a, b, c, d, e, f) (uses for memory address)

$(\_\ \_)_{10}$ (10^1 10^0) , $(\_\ \_)_2$ (2^1 2^0)

· 0 سے 9 تک values represent کرنے کے لیے unique digit ہے -

· 9 سے اوپر کو represent کرنے کے لیے multiple digits یوز ہوں گے۔

$\underline{7}\ \underline{8}$ (10^1 10^0)

· ہر digit اپنی value کے weight سے multiply ہوگا -

10^1 10^0
1 0

$R^{10} + R^{0} = 10$

Binary to Decimal:

	64 (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)
Binary:	1	0	1	1	0	0	1

Decimal: 89

· Binary سے decimal ← convert کرتے وقت اس کی value (2^i) representation
کو اس کے Bit (1) سے multiply کریں گے - اور آخر میں سب کو جمع کریں گے۔

Decimal to Binary:

Decimal : 89

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
Binary : 0	1	0	1	1	0	0	1

(89) number ہم میں conversion ← Binary سے Decimal ۔
اس سے قریب تر (64) کم digit representation کو minus کریں گے اور اس کی جگہ (1)
لکھیں گے ۔ پھر minus کرنے سے جو ans آئے گا اس کے ساتھ یہی process کو repeat کریں گے ۔

(2^6)	(2^4)	(2^3)	(2^0)
89 − 64 = 25	25 − 16 ≠ 9	9 − 8 = 1	1 − 1 = 0

Octal to Binary:

Octal : $(127)_8$

2^2 2^1 2^0	2^2 2^1 2^0	4 2 1
Binary : 0 0 1	0 1 0	1 1 1

Octal to Decimal:

Octal : $(127)_8$

Binary : (001 010 111)$_2$

Decimal : $(87)_{10}$

Binary to octal:

Binary : (1 4 2 1 4 2 1) (1100110)$_2$

Octal : $(146)_8$

Hexadecimal to Binary:

8 4 2 1
1 1 1 1 = 15

Hexadecimal : $(9b85fea)_{16}$

Binary : 5 f e 9 a
0101, 1111, 1110, 1010
8 b 8 0 0
1001, 1011, 1000

Binary to Hexadecimal:

Binary : $(1011\ 0110)_2$
8421 8421

Hexadecimal: b6

- Half byte is called nibble (4 bits).
- Range of 1 Byte is 255.

128 64 32 16 8 4 2 1
1 1 1 1 1 1 1 1 = 255

- 1byte is required to represent 1 bit.
- 1st Number System was BCD (1-nibble)
- ASCII (0-255) (8-Bits)
- Uni-Code to represent all languages
- UTF-16
- UTF-32 (Emojis)

32768 512 256 128 64 32 16 8 4 2 1 0

↳ This bit is used for sign representation.
↳ 1 For negative.
↳ 0 for positive.

Java میں character کو UTF-16 سٹور کرتا ہے - اس لیے یہ 2 Byte کا ہے۔

Binary Coded Decimal: (BCD)

- In this code each decimal digit is represented by a 4-bit binary number
- Decimal digit (0-9) → (# 143 is decimal number not a decimal digit.)
- 4 bit binary number (9 = 8 4 2 1 → 1 0 0 1).
- It cannot consume any digit exeed 9.
- (10-16) 4-bit states are unused. iso it is wasteful.
- Representing a decimal number requires extra bits.

ASCII Code:

- ASCII stands for American Standard Code for Information Interchange.
- I uses (8-bits) or 128 characters to represent numbers.
- 32 control code, 95 printable characters and a DEL character.

- Limitation of ASCII is that it cannot represent letters of languages other than English.

Unicode

- Unicode provides a unique number for every character, no matter what the platform, program or language is.
  - → UTF-8 (8-bit form) to avoid contradiction with ASCII.
  - → UTF-16 (Other language representation)
  - → UTF-32 (Emoji forms)

- Minus values are store in the form of 2's complement.
- Integer.toBinaryString(10);

6- Bitwise Operator

perform ← Bitwise تنفيذ operands على Bitwise operator .

يعني يتم -

1011 + 0100 = 1

- Exclusive OR (XOR) (^)

0	0	0
1	0	1
0	1	1
1	1	0

• & (Bitwise AND)

```
  0 0 0 0 1 0 1 0
& 0 0 0 0 1 0 0 0
Result 0 0 0 0 1 0 0 0
```

• | (Bitwise OR)

```
  0 0 0 0 1 0 1 0
  0 0 0 0 1 0 0 0
  0 0 0 0 1 0 1 0
```

• ^ (Bitwise XOR)

```
  0 0 0 0 1 0 1 0
  0 0 0 0 1 0 0 0
  0 0 0 0 0 0 1 0
```

Data manturization.

• جب کوئی چیز size میں کم اور efficiency میں زیادہ ہو جائے
تو اسے data manturization کہتے ہیں۔

• یہ technique صرف constant data پر اپلائی ہو سکتی ہے۔

Usage:

```
Ali:    0 0 0 0 1 1 0 1
Raza:   0 0 0 0 1 0 1 0
Unique(XOR): 0 0 0 0 0 1 1 1
Same(AND)    0 0 0 0 1 0 0 0
Mask Applied: 0 0 0 0 1 1 1 1
(OR)
```

CS = 1
IT = 2
AI = 4
SE = 8

a = 10 0 0 0 0 1 0 1 0

b = 8 0 0 0 0 1 0 0 0

a & b = 0 0 0 0 1 0 0 0 = 8

a | b = 0 0 0 0 1 0 1 0 = 10

```
public class BitwiseOperator{
    public static final int IT = 1;
    public static final int CS = 2;
    private int selectedCourses;
// Constructor
    public BitwiseOperator(int selectedCourses){
        this.selectedCourses = selectedCourses;
    }
// Getter & Setter
    public void setSelectedCourses(int selectedCourses){
        this.selectedCourses = selectedCourses;
    }
    public void getSelectedCourses(){
        return selectedCourses;
    }
// Functions
    public void ITSelected(){
        selectedCourses |= IT; }
    public void CSSelected(){
        selectedCourses |= CS; }
```

Bitwise Left Shift operator:

۱. اس operator میں ہم left ← side سے 'O' Bit ← insert کرتے ہیں اور باقی bits ← left کو move کر جاتی ہیں۔

```
int a = 10;
a << 1;
```

	128	64	32	16	8	4	2	1	New Bit
	0	0	0	0	1	0	1	0	← '0'
0 ←	0	0	0	1	0	1	0	0	Left Shift
↑ out								↑ In New Bit	

۱. اس operator کا side effect نہیں ہوتا۔

Bitwise Right Shift Operator-

۱. اس operator میں right side سے 'O' Bit ← Insert کرتے ہیں اور باقی Bits ← right کو move ہو جاتی ہیں۔

	128	64	32	16	8	4	2	1	
0 →	0	0	0	0	1	0	1	0	
	0	0	0	0	0	1	0	1	→ 0 ↑ out

Uses of Shift operator:

- steganography میں کسی image، video کے ساتھ کسی File کو attach کیا جاتا ہے۔ یہ Process shift operator کے ذریعے کیا جاتا ہے۔
- اس کے ذریعے pc میں virus بھیجا جا سکتا ہے۔
- یا کوئی important ← data ← transfer کیا جاتا ہے۔
- کسی بھی color کے 256 shades ہو سکتے ہیں۔

⇒ A R G B

↓ Opacity, Red, Green, Blue.

· Unsigned Right Shift operator: (>>>)

int a = 10

a >>> 1;

New	128	64	32	16	8	4	2	1
0	0	0	0	0	1	0	1	0
	0	0	0	0	0	1	0	1

· Complement operator (~)

- يقلب الـ complement الـ Bits بـ .

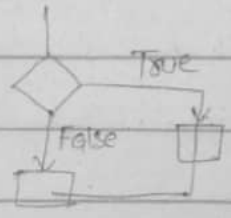
~a;

	1	0	1					
~	0	1	0					

Flow Control Statement./Decision Make

If statement:

```
If (condition){
}
```



→ Compound Statement operator:

```
{
}
```

· This operator determines the lexical scope of ar variable.

· It is used to combine multiple statements.

→ Lexical scope is where a variable can be accessed. (Scope of variable).

If (condition) {
    statement 1; → Check by if statement
    statement 2; → Not check by if statement
}   OR sequential statement.

- In if statement, compound condition operator is used to ~~check~~ execute/ignore multiple statements.
- If is a programming construct that allow to execue set of statement or ignore set of statement.
- "If" is should be used with variable condition Not with constant condition.
- ☆ It is better to usee getter & setter while getting or setting variable.

How to solve a problem?

→ The solution of a problem is within a problem.

→ Programming کر کے آتی ہے ڈیزائن کے نہیں آتی

→ problem کو پہلے سمجھو یعنی پہلے run ← dry کو Program

کو سمجھ کر اسے دماغ میں run ← dry کرو۔

String Comparision:

String ← Compiler کے compare کرنے سے == کو String ←

Objects.equal(a,b) سے compare کرو ref variables کے

compare کرو content کے String

210126 % 10 = 6
210120

~~Unsigned right shift operator.~~

· Nested If:

۔ اگر if کے true block میں if لگے تو اسے nested if کہتے ہیں۔

۔ اگر If کے else/false block میں If لگے تو اسے nested else if کہتے ہیں۔

۔ C/C++ میں main function کی return type ← int ہونی چاہیے اور اسے 0 return کرنا چاہیے۔ اگر 0 کے علاوہ کچھ اور return ہوتا ہے تو compiler اسے abnormal consider کرے گا۔

```
if (condition){
        if (condition){      → Nested If
        }
}
```

· Nested else-if:

۔ اگر if کے false block میں مزید if-statements لگے تو اسے nested else-if کہتے ہیں۔

```
if (condition){
        statement;
} else { sout ("Hello world");
        if (condition){
                statement1;
        }
}
```

۔ یہ ہم وہاں یوز کریں گے جہاں ہمیں else کے ساتھ if اور بھی statements سے write کرنے ہوں۔

OR

read ← char ← User

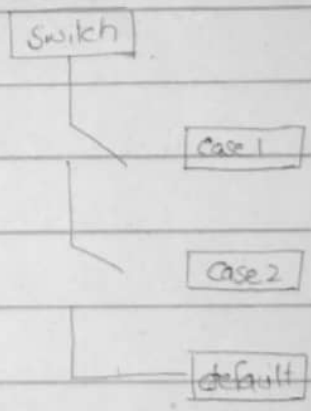
```
if ( condition) {
    statement1;
} else if (condition){
    statement2;
}
```

Switch Statement.

- switch کے conditions ایک نہیں ہوتیں۔
- switch ← runtime پر if کی نسبت زیادہ fast ہے۔
- اگر کوئی ایک case ← match ہو تو اس کے نیچے والے تمام cases ← execute ہوں گے۔

```
switch (expression) {
    case constant;
        statement;
    case constant;
        statement;
    default ;
        statement;
}
```

Case label



- اگر ہم چاہتے ہیں کہ match case کے بعد کوئی execute ہو تو ہم اس case میں "break" use کریں گے۔
- اگر کوئی default case نہ ہو تو اسے dangling case کہیں گے۔

★ جس operand کے ساتھ کوئی operator نہ ہو اس کے ساتھ + ہوتا ہے جسے primary expression کہتے ہیں۔

- Switch is a statement, so we cannot assign it to a variable.

Switch as an Expression:

- Java modified itself for "switch as an expression. in its latest version.
- switch کے expression میں ہم condition بھی لکھ سکتے ہیں لیکن condition بھی ایک expression ہے۔

```
return { switch (operator) {
        case '+' -> getFirstOperand + getSecondOperand;
                 ↑
                lamda
} ; -> semicolon
      is must here
```

OR

```
double result = switch (operator) {
        case '+' -> getFirstOperand + getSecondOperand;
        case '-' -> getFirstOperand - getSecondOperand;
        default -> 0.0;
}
return result;
```

اس case میں یہ return ہوگا

← switch as a statement میں ہم کسی variable کو store نہیں کر سکتے ہیں۔

← switch as an expression میں ہم variable کو store کروا سکتے ہیں۔

← اس میں سب cases کو ایک ہی data type return کرنی ہوتی ہے۔

← switch میں condition کو نہیں لگا سکتے۔

Conditional Operator (Ternary Operator)

condition ? true statement : false statement ;

```
int a = 10;
int b = 20;
int max = a >= b ? a : b;
```

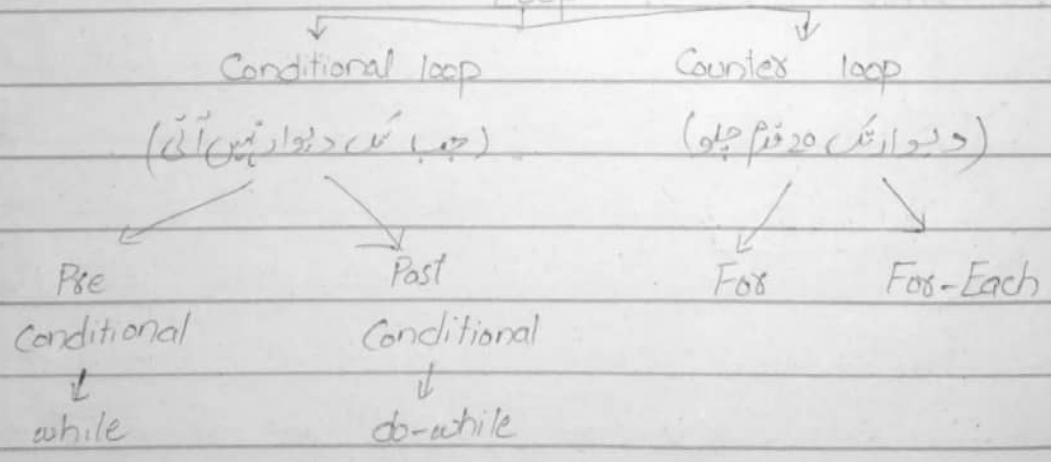
Type Inference:

```
var num = 12;
var myChar = new MyCharacter();
```

(MyCharacter: class name)

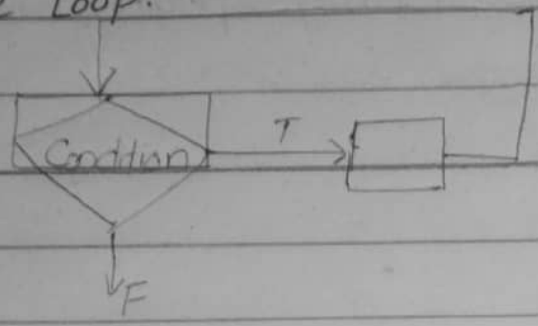
LOOPS.

- جہاں سے شروع ہو وہیں ایںڈ ہو۔
- اسے iterate بھی کہتے ہیں۔
- دو طرح کے ہوتے ہیں۔ => Basic loops



- by default loop ← Single statement کو deal کرتا ہے۔

1- While Loop:




```
while (condition){
        statement 1;
        statement 2;
}
```

```
while (condition):
        statement 1;
```

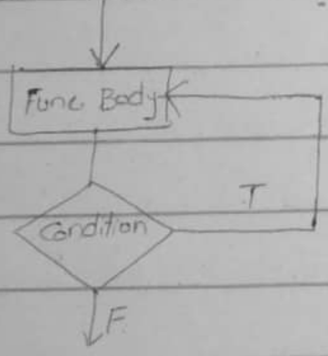
← اس کی automatic repeatation ہے اور conditional entrance ہے۔

← hasInt چیک کرتا ہے کہ یہ Int ہے یا نہیں تو یہ true یا false return کرے گا اور boolean ← return کرے گا۔

———— Assignment *——*

```
public static int readValidIntFromRange (String outMessage,
        int startRange ,
```

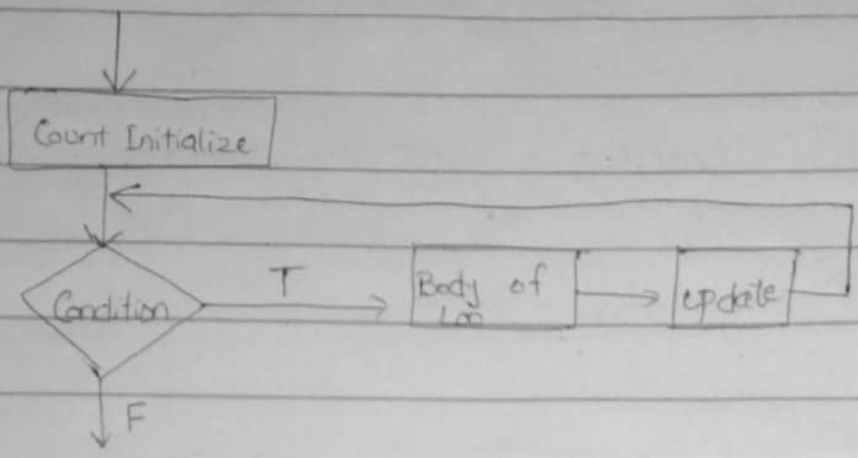
2 – Do – while Loop:



← do-while ایک بار لازمی run ہوگا۔

← اس کی repeatation conditional ہے اور entrance ← automatic ہے۔

3- For Loop



• for loop میں variable جو condition کو چیک کرتا ہے اسے Counter variable کہتے ہیں۔

Executed once

```
for (intialization ① ; condition ② ; update ④)
        Statement ③ ;
```

• for loop میں ہم ایک سے زیادہ intialize variable اور update کر سکتے ہیں۔

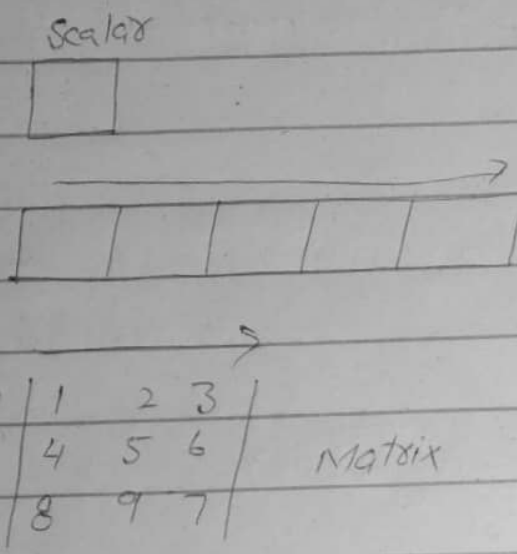
```
for (int i=1 , j=10 , k=20 ; i<=10 ; i++,
    J-- , k-=2)
        Statement;
```

Conditional variable

out ⇒

• اگر loop کے اندر ہمیں کسی variable کی ضرورت ہو تو
loop کے اندر ہی declare کرنا چاہیے۔

★ Nested Loops:- (Loop in Loop)



← Scalar کی چونکہ کوئی dimension نہیں تو اس کے لیے loop کی
ضرورت نہیں۔

← Vector چونکہ one-dimensional ہے تو یہ single loop سے حل ہو
جائے گا۔

← Matrix چونکہ two-dimensional ہے تو یہ دو nested loop سے حل
ہو جائے گا۔

← لہٰذا جتنی dimensions ہوں گی اتنی ہی nested loops لگیں گی۔

Image Processing:

(1) Raster Images

(a)

(b) Bitmap images.

(2) Vector Images

→ Bitmap images are two dimensional pixels.
جو کہ اس میں ہر pixel کو چیک کر color دیا جاتا ہے اس
لیے اسے Bitmap img کہتے ہیں۔

BMP : اس میں data loss سب سے کم ہوتا ہے۔ (size زیادہ ہوگا 20)

Joint Photographic Expert Group : JPEG (size نسبتاً 19)

PNG : latest ہے، اس میں transparent کے لیے
option alpha ہے۔

→ Vector images:
← ان images کو zoom کرنے کے لیے نہیں بنتی۔

BufferedImage sourceImage = null;
← جو image کو hdd سے load کر کے ram میں لائے گا۔

File file = null;

← Runtime کے جو error آتا ہے اسے exception کہتے ہیں۔
exception handling کے لیے ہم try-catch بلوز کرتے ہیں۔

Exception Handling.

← ایسا code جس کا ہمیں پتا ہو کہ وہ runtime پر error دے گا اسے try-block میں لکھنا چاہیے۔

```
try {
    ~~Source~~ File file = new File("Path");
    sourceImage = ImageIO.read(file);
}
③ catch (Exception ex){
    System.out.println(ex.getMessage());
}
① catch (IllegalArgumentException ex){

}
② catch (~~Ex~~ IO Exception ex){
}
④ finally { }
```

← ہم generalized catch block کو specified catch block کے آخر میں لکھیں کیونکہ compiler ← sequence wise ← exceptions کو read کرے گا۔

← finally block ہر صورت میں execute ہو گا۔ جیسے اگر server سے connection ← close کرنا ہو تو ہم finally block میں لکھیں گے۔

★ Checked & unchecked Exception.

★ The finally block may not execute if JVM exits while the try or catch code is being executed.

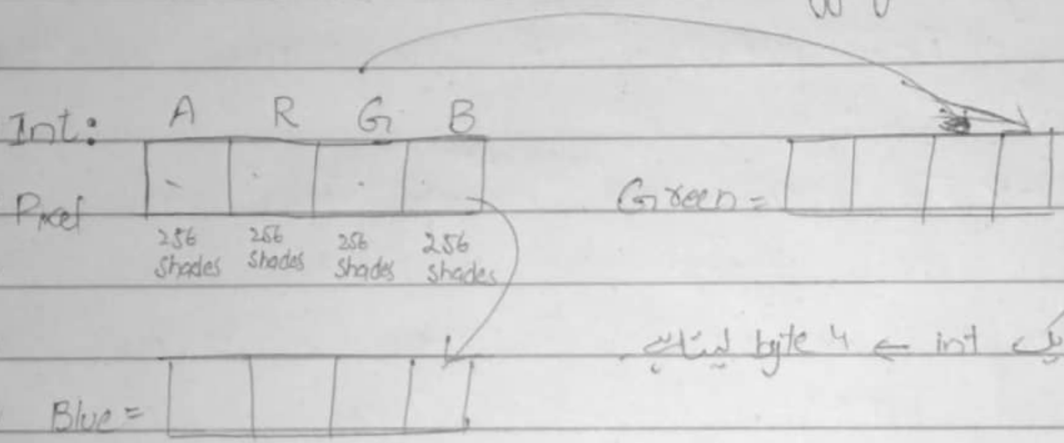
→ Image type is based on the basic colors

← جب image میں transparency لانی ہو تو ہم ARGB ← use کریں گے۔

Color Extraction From Pixel.

Syntactical string → URI → URL

→ assert is used for debugging.



← ایک int ← 4 byte لیتا ہے۔

← 0xff ← hexadecimal ہے۔ 0x ← Compiler کو بتاتا ہے کہ یہ ایک hexadecimal ہے۔

← hexadecimal ← 4bits لیتا ہے۔

← اگر ہمیں pixel میں سے alpha نکالنا ہے تو ہمیں اسے 64-bits right shift ہوگا۔ اسی طرح red کے لیے 32-bits اور Green کے لیے 16-bits اور blue کو right-shift-0-bit کرنا ہوگا

Stagnography:

Stagnography کے میں ہم least significant bit میں اپنا data چھپا سکتے ہیں۔

← Image پر border ← apply کرتا ہے۔

← frameImage + userImage

Collections

Array:

→ The collection/ (contain) of homogeneus data types.

→ The consecutive/ (contiguous) memory block of same data type.

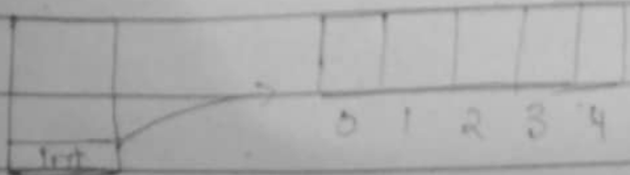
→ Arrays are reference type in java.

```
int list[ ] ;        // old style
int[ ] list1 ;       // new style

                              → Array size
int list[ ]  =  new  int[5];
int [ ]  list  =  new  int[5];
```

→ Array size is not modifiable.

→ If we don't know about how much data will come, we will use linked list.



Initializer

```
list[0] = 10;
list[1] = 12;
```

Intialize:

```
int[] list = new int[]{10, 12, 24, 26}; //old
int[] list1 = {10, 12, 24, 26, 28};  //New
```

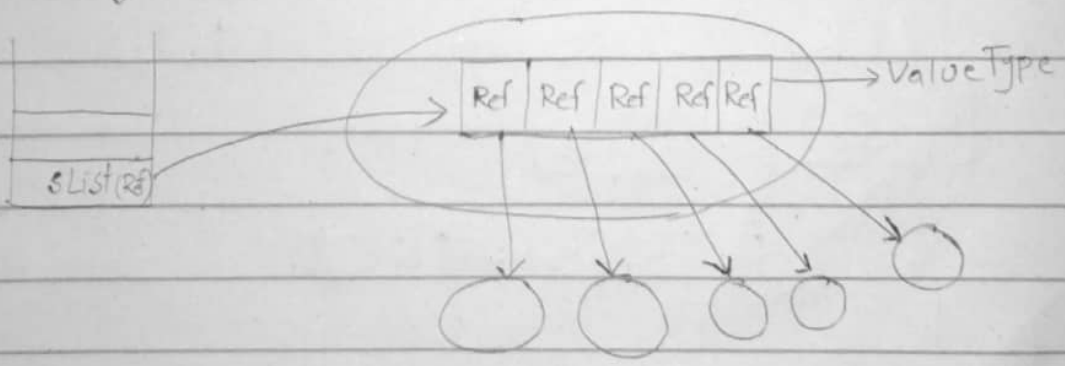
← user سے input میں array لے کر max بتانا ہے

← find کرنا ہے prime number

← count of even/odd بتانا ہے

```
Student[] sList = new Student[5];
sList[0] = new Student();
sList[1] = new Student();
```

Value Type Reference Type



← Array کے اندر صرف value type ہی آسکتا ہے۔

← اوپر student کے objects کی ref variable کی array بنی ہے۔

Multi-dimension Array

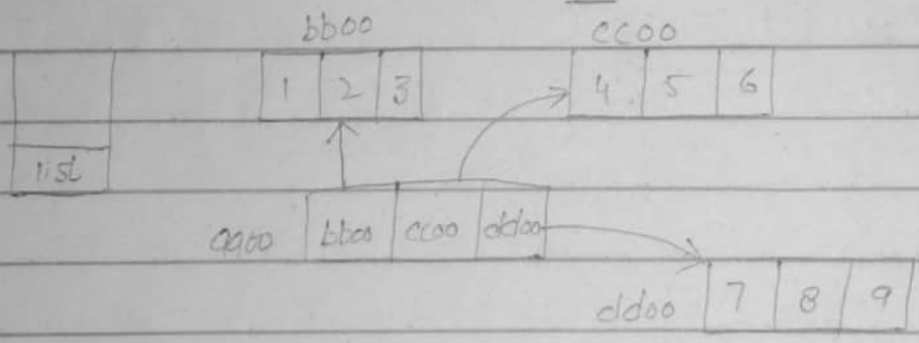
```
int [][] list = new int[3][3];
```

```
int [0][0] = 10
int [0][1] = 20
int [0][2] = 30
   :
int [2] [2] = 50
```

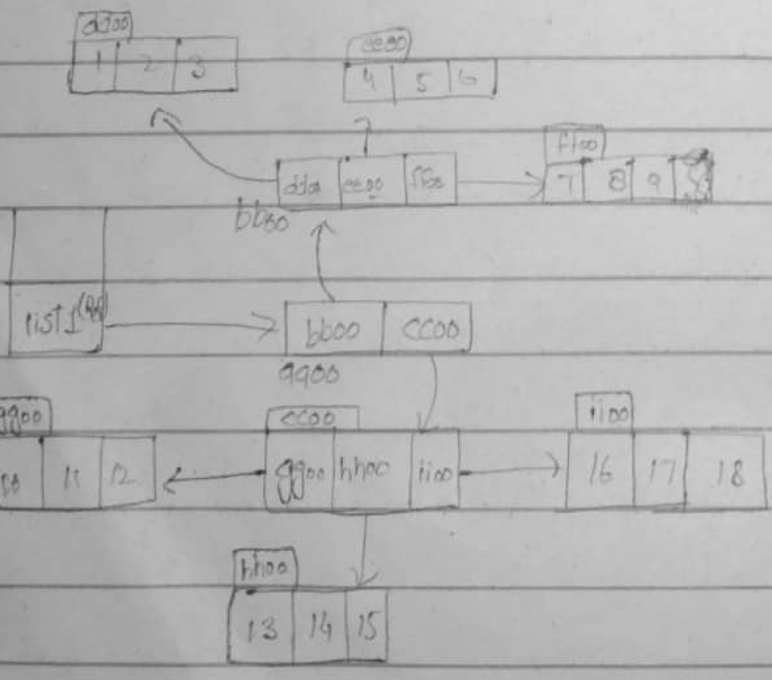
	0	1	2
0	10	20	30
1	3	4	5
2	6	7	50

- في last array بيبقى array of integer ←

new int [3][3]

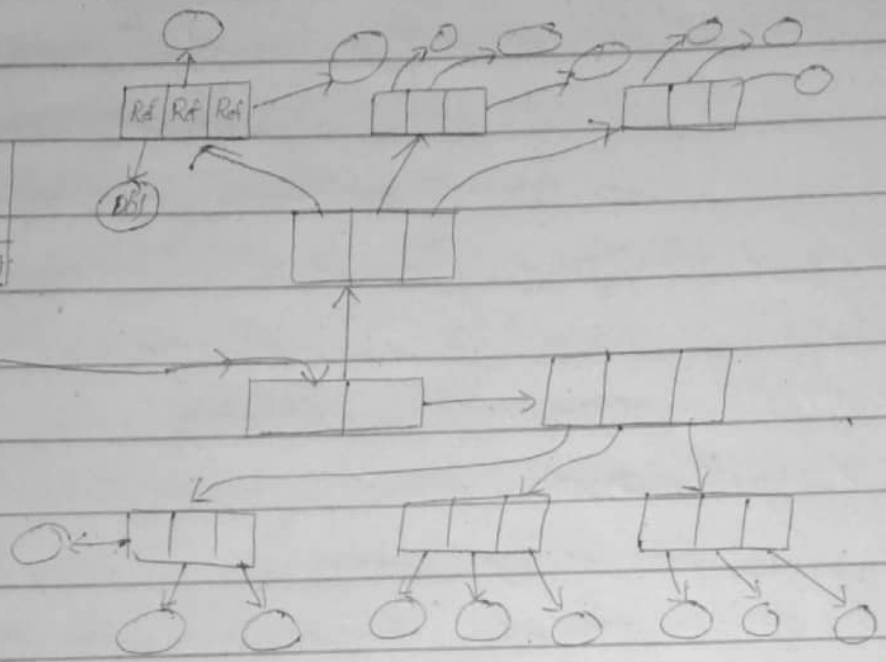


- int[][][] list = new int[2][3][3];



Reference Type in Multi-dimen. Array

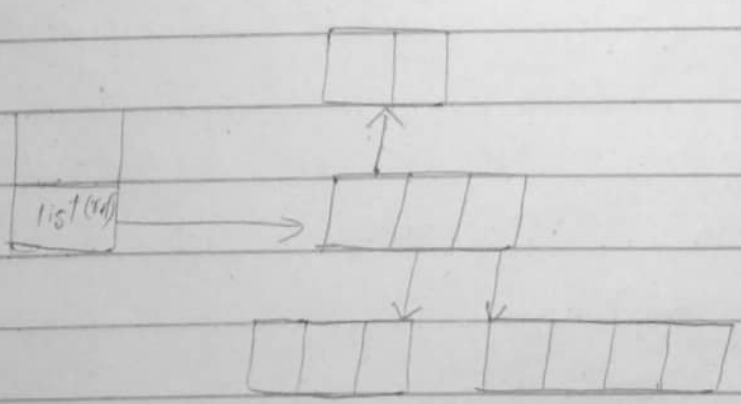
Student[][][] newList = new Student[2][3][3];



Jagged Array

- في variable ← last dimen من Jagged Array ←

int[][] list = new int[3][];



Array List

← ArrayList ایک third-party ← class ہے۔

← ArrayList کے ذریعے ہمیں add,update,delete کے builtin ← func. ملتے ہیں۔

← ArrayList ایک Generic/Template ← type ہے۔

← ArrayList کی "<>" میں ہم value type نہیں دے سکتے۔ صرف ref. type دے سکتے ہیں۔

* ArrayList's size is modifiable.

```
ArrayList <Integer> list;
```

← ArrayList کو ہم generic اس لیے کہتے ہیں کہ اس میں ہم جس مرضی dataType کا operation ← perform کریں اس سے ArrayList کے methods پر کوئی فرق نہیں پڑتا۔

methods(insert,update etc)

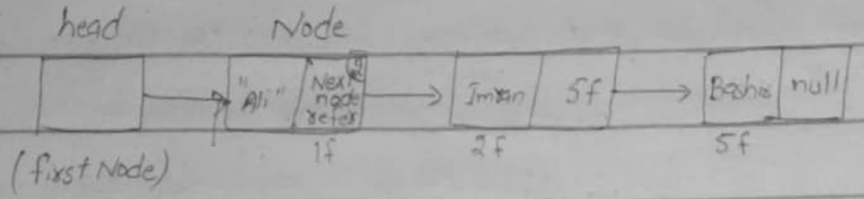
← ArrayList ← by-default ← 20 length کا array بنتا ہے۔ جیسے ہی 20 items سے increase کریں گے new array 40 length کا بنے گا اور پچھلے 20 members اس میں copy ہوں گے۔

```
ArrayList <String> list = new ArrayList <String>();

list.add All(list2)
list.contains("Omer");
list.indexOf("Omer");
list.remove("Omer");
list.removeAll(list2);
```

Linked List

← Linked List میں ہم ایک node بنائیں جو اگلے node کو refer کرتا ہے اور اس میں data ہو گا۔



← پہلی node ← head ہوگی جس سے اگلی nodes ← link ہوں گی۔

← آخری node کے ref میں null ہوگا۔

← Linked List میں traversing ← slow ہوتی ہے۔

← اگر ہمیں آنے والے data کا size پتا ہو تو ہم ArrayList use کریں گے اور اگر آنے والے data کا size نہ پتا ہو تو ہم LinkedList یوز کریں گے۔

Stack

← Stack میں data (List In First Out) ہو گا۔

← Stack کے تین functions ہیں۔

- Push
- Pop
- Peak

← Stack میں ہم کسی element کا index ← get method سے حاصل نہ ہو گا۔

Scanned with CamScanner

Hash Map (Key Value pair)

Convert میں form ← unreadable کو data میں Hash Map ←

کیا جاتا ہے۔ جو کہ re-convertable form ← میں ہوگا۔

Encription ← 2 type کی ہوتی ہے۔

Symmetrical Encription ←

A Symmetrical Encription ←

اس میں decription کے لیے ایک key کی مدد سے ہوتی ہے۔

Hashing ← ایک enc asymmetric ہے

Hashing ← میں ڈیٹا retrieval زیادہ fast ہے۔

```
Hashmap<Integer, String> map = new Hashmap<>();
map.put(1, "Ali");
```

Hashmap ← میں key unique ہوگی۔

Java ← key میں بنائے گی کہ data کیا store کرنا ہے۔

For Each Loop

→ Read only Loop

← صرف collection کو traverse کرنے کے لیے بنایا گیا ہے۔

```
LinkedList<String> list = new LinkedList<>();
for (String value : list){
    sout(value);
}
```

جس کی type collec ہے۔

اس سے ہم collection میں کچھ write نہیں کر سکتے۔

Advanced OOP

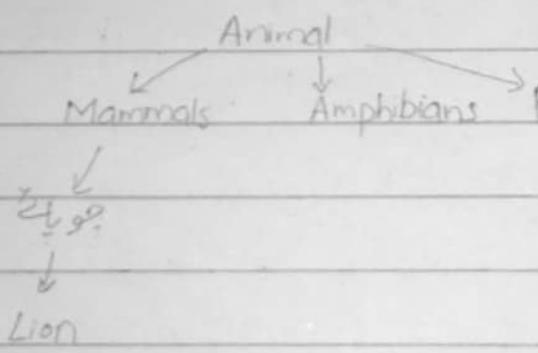
★ Inheritence:

- آپ کسی class کے تمام members دوسری class میں access کرنا چاہیں تو ہم inheritance use کریں گے۔
- Parent → child
- Super → Sub
- Base → Derive
- Inheritance کے لیے is-a relationship کا true ہونا چاہیے۔

```
Class Student extends MessStudent{
}
child          parent
```

- Coding سے پہلے Object Oriented Design کی جاتی ہے۔

Multilevel Inheritance ←



- parent کلاس child کے members access کرسکتا ہے ← child کلاس parent کے members access نہیں کرسکتا۔
- → Parent is generalized most class and last children is specialized most class.
- parent class اس کو extend کرے گی child class جس parent class ← implicitly object کی ہے۔

Constructor in Inheritence

← child class کے constructor میں parent کے constructor کو call کرنا mandatory ہے۔

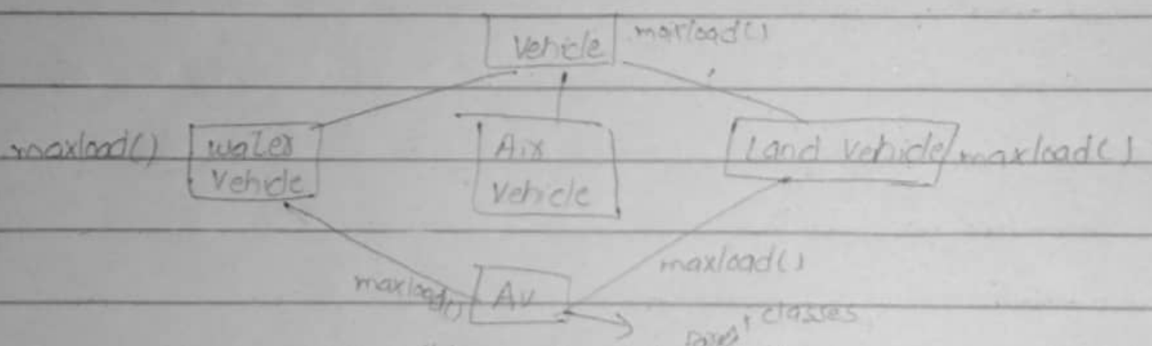
← اگر child class میں default con ہو تو وہ parent کے default cons کو implicitly call کرے گا۔

← اگر parent میں default con نہ ہو تو child class implicitly parent کے constructor کو call نہیں کر سکتا۔

☆ ← super keyword ← immediate parent کو call کرتا ہے۔

← Java میں multiple inheritance نہیں ہوتی۔

← Multiple inheritance میں diamond problem



اس کی immediate parent classes siblings ہیں۔

C++ میں یہ problem virtual inheritance سے solve کی جاتی ہے۔

← super سے grandparent کو access نہیں کر سکتے۔

Types of Inheritence.

- Explicit Inheritence
- Implicit Inheritence.

← جب کوئی class کسی class کو extend نہ کر رہی ہو تو وہ implicitly Object class کو extend کر رہی ہوتی ہے۔

← Java میں تمام classes کی parent class Object class ہے۔

Scanned with CamScanner

→ Explicit Inheritence use extends keyword.

Advantage

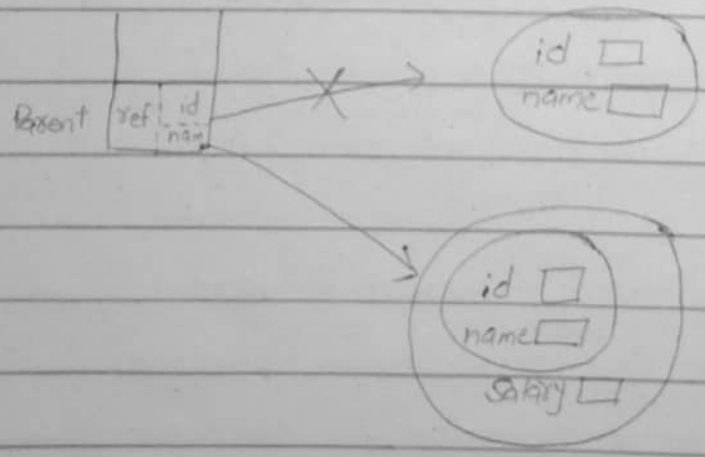
→ 2nd advantage of inheritence is "Generalization".

parent = new child();

Generalized reference variable.

Specialized object.

→ Reusability



Abstract Classes

← Abstract class کا object نہیں بنتا۔

← جس class کا object بن سکتا ہے اسے concrete class کہتے ہیں۔

← real world میں کوئی animal نہیں تو وہ abstract رکھی جائے گی۔

← جو چیز real world میں exist نہیں کرتی اسے abstract رکھا جائے گا۔

← Parent (abstract) کے members کو child میں use کرنے کے لیے parent کا constructor بنایا جاتا ہے۔

← Concrete child class میں abstract parent کے تمام abstract methods کا ہونا لازمی ہے۔

Function Overriding

→ Polymorphism Real.

← اگر parent کے کسی function کو same ~~differ~~ (کا + same name) child میں (function differs میں) Signature سے use کرنا ہو تو overriding ← use کرتے ہیں۔

→ Self calling of a function is called recursion.

```
@Override
public void setAge(int Age) {
    super.setAge(age);
}
```

← اگر super کا keyword نہ لگائے تو child کے setAge کی recursion ہو گی۔

← Overriding ہمیشہ parent اور child کے درمیان ہو گی۔

← کیا concrete class سے کسی abstract class سے extend ہو سکتی ہے۔ ؟ (Yes)

← کیا object class abstract یا concrete ہے۔ ؟ (Concrete)

← abstract class کے constructor اور function کی overloading ہو سکتی ہے۔ ؟ (Yes)

← abstract class میں members static data بن سکتے ہیں یا نہیں ؟ (Yes)

← object class کا constructor implicit یا explicit ہے۔ ؟ (implicit)

← Super کا keyword ← abstract child میں abstract parent کے constructor کو call کرنے کے لیے یوز ہو سکتا ہے یا نہیں ؟ (Yes)

Abstract Method

← abstract func کی body نہیں ہوتی - صرف signature ہوتا ہے -

public abstract void draw( );

← ایسے تمام methods جو کہ تمام concrete child class میں override ہونے must ہوتے ہیں اسے ہم parent میں abstract بنا دیتے ہیں۔

← اگر override کا keyword نہ لگا ئیں کیا overriding ہو گی ؟

Virtual Overriding	Non-Virtual Overriding.
public virtual void draw()	public overrides void draw()
{	{
}	}
ref variable کی type جس ← ہوگا اس سے کوئی لینا دینا نہیں ہوگا۔	ref variable کی type جس ← ہوگا اس class کی type draw کا کال ہوگا۔
decide پر runtime ← ہوگا کہ کونسا draw کال ہونا ہے۔ (draw کس obj کے ساتھ کال ہو رہا ہے)	decide پر compile time ← کیا جائے گا کہ کونسا draw کال ہوگا۔
Generalization ← اس میں ہوتی ہے۔ RTTI ←	Generalization ← اس میں نہیں ہوتی۔
→ Late Binding ←	→ Early Binding ←
→ Runtime Binding	→ Compile Time Binding.
→ Dynamic Binding	→ Static Binding.

Interface

← جس کے ساتھ ہم نے interact کرنا ہے Interface کہلاتا ہے۔

← ہر class کے public members ← primary interface ہیں اس class کے۔

← Interface ایک secondary interface ہے۔

← Interface کو بار بار implement کرنا پڑتا ہے برعکس inheritance کے۔ جیسے اولاد والد کی قابلیت کو بیچ کر بھی اس کی طرف لے سکتا ہے۔

← ایک interface کا ref variable ہر اس class کا reference رکھے گا جس میں اسے implement کیا گیا ہے۔

← Interface ~~صرف~~ Java میں نے سب سے پہلے introduce کروایا۔

← Interface کے identifier کے شروع میں "I" لگایا جاتا ہے۔

← Java 8 تک old interface یوز ہو سکتے تھے۔

← Interface کے اندر data members (variable) نہیں ہوتے۔ اس میں صرف abstract, default, static functions ہوتے ہیں اور final data members ہوتے ہیں۔

← class میں functions ← by-default ← Concrete ہیں۔

← interface میں functions ← by-default ← abstract ہوتے ہیں۔

← Interface ← partial implementation نہیں ہوتی۔ اس کو complete ← implementation ہوگی۔

```
public interface ICricket{
    void bowling();
    void batting();      > ~~Abstract Methods~~
}

public Student implements ICricket{
    //must have bowling() & batting();
}
```

← ایک class → multiple interfaces کو implement کر سکتی ہے۔

← تمام collections کی parent class → Iterable ہے۔

← interfaces → differ. type کی classes کو کسی ایک interface پر Generalize کرنے کے لیے یوز ہوتے ہیں۔

← Interface میں صرف default & static → member funcs کی body ہوتی ہے۔

★ All abstract, default & static methods in an interface are implicitly public, so we can omit public keyword.

? کیا default کا keyword صرف interface میں یوز ہو سکتا ہے؟

★ In interface all constants are implicitly public, static and final so we can omit them.

← Libraries دو طرح کی ہوتی ہیں۔

active. → جو کام پہلے سے library کو polish کر رہی ہے

passive. → library شدہ بنا کر چلا گیا۔

← abstract class → same type کی classes کو generalize کرنے کے لیے یوز کیا جاتا ہے۔

← abstract class میں constructor ہوتا ہے interface میں نہیں ہوتا۔

← abstract class → پر interface کو implement کر سکتا ہے

← اگر child کی abstract parent میں interface کا کوئی method functionality دیں تو child میں اس مخصوص method کو override کرنا لازم نہیں ہے۔ اس طرح interface کی partial implementation ہو سکتی ہے۔

Inheritance of Interfaces.

← ایک interface اپنے پاس کسی class کے obj کا ref variable رکھ لیتا ہے جس میں اسے یا اس کے child کو implement کیا ہے۔

← ایک interface ← multiple interfaces کو extend کر سکتا ہے۔

Abstraction

← Abstraction دو طرح سے ہو سکتی ہے۔

→ abstract class

→ Interface

← List کو implement کرتے ہیں۔ ← LinkedList اور ArrayList

CharSequence → is an interface that is implemented in String class

→ When making an architecture when we need concrete implementation we should implement abstraction and use this abstraction in concrete

Inner Class

← ایک class کے block کے اندر دوسری class بن سکتی ہے جسے inner class کہتے ہیں۔

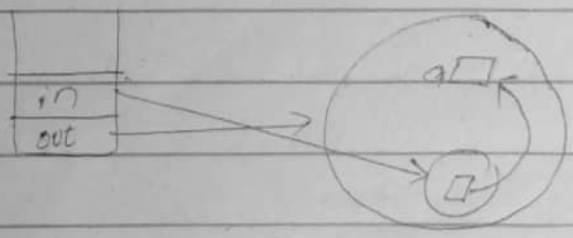
← inner member class کی data members کی طرح۔

← Linked List دو طرح کی ہوتی ہے۔

sequential linkedlist..

(tree, graph) non-sequential linkedlist..

← Inner class میں outer class کے private members کو access کر سکتے ہیں۔



← inner classes میں nested class ہوتی ہیں لیکن nested classes میں inner classes نہیں ہوتیں۔

← static inner classes کو nested classes کہہ سکتے ہیں۔

Inner Local Class

← اگر inner class کسی func. میں بنے تو اسے inner local class کہتے ہیں۔

← GUI programming میں Event based ہوتی ہے۔

← ایسا event جو کسی function کے ساتھ bound ہوتا ہے۔

→ For inner class to access a local variable that must be final. ←

← Event handling کے لیے inner local class بناتے ہیں نہ کہ
top level class بناتے ہیں۔

Inner Anonymous Class

← جس class کا کوئی نام نہیں ہوتا۔ اور نہ class keyword کا
interface
↓
OnClickListener listener = new OnClickListener{
@override
public void onClick( ){

}

}

→ Anonymous class is an expression.

- Only Final local members are accessible.

Inner Static Class

← top level class ہے
← اس کو ہم nested class کہتے ہیں کیونکہ دونوں classes
کا ایک same ہے۔
← Top level class ← implicitly ← static ہوتی ہے۔
← اگر 2 classes کا name ایک نہیں

Association.

- Inheritance
- Association

← ایک class کے اندر دوسری class کا object کو use کرنا association کہلاتا ہے۔

← اس کے لیے has-a کا relation ہونا چاہیے۔

Types

- Aggregation
- Composition.

← اگر association ہو رہی ہے اور جس میں ہو رہی ہے تو اگر وہ چیز لازمی ہے تو composition ہے۔ اس کے بر عکس دوسری ~~composition~~ association aggregation ہے۔

← اگر class میں دوسری associate class ہو رہی اور وہ ایک class سے associate ہو رہی ہے تو وہ composition کہلاتی ہے۔ اور اگر ایک سے زائد classes سے associate ہو رہی ہے تو وہ aggregation کہلاتی ہے