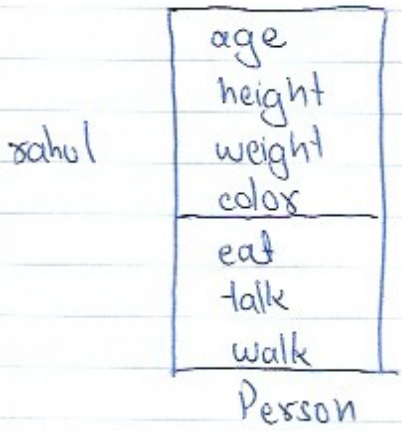
object
- exists
- properties
- action

class
- doesn't exists

class: Collection of objects
Collection of variables and methods



oops

class
variable
method
object

jvm: java virtual machine
It converts the byte code to machine code

Rules:

JDK -> JRE -> JVM

classname:
Scanner
ArrayList

methodname:
get()
getMessage()
printStackTrace()

variable name:

1) A to Z or a to z or _ or $
   0 to 9
   ex: int 123total; //wrong
       int total123; // correct

2) java is case sensitive

3) Reserved keywords can't be used.
   ex: for, while, break, if, int .. etc

4) Spaces are not allowed between the variable name.
   ex:
   int area of traingle; //wrong
   int area_of_traingle; // correct

- IDE (Integrated Development Environment)

1. Eclipse
2. My Eclipse
3. Netbeans
4. Sublime

How to compile:
syntax
javac filename.java

1) javac Test.java
2 javac first.java second.java
3 javac *.java

Note: once we compile .class file will be generated

how to run:
syntax:
java classname
ex:
java Person

Note: Only one class can be run.

public: Anybody can access.
private: Limited to that class

object → programmer
static → predefined

public: It can be accessed from anywhere
static: It doesn't depend on object, it can be accessed using classname.
main: start of program execution.

static

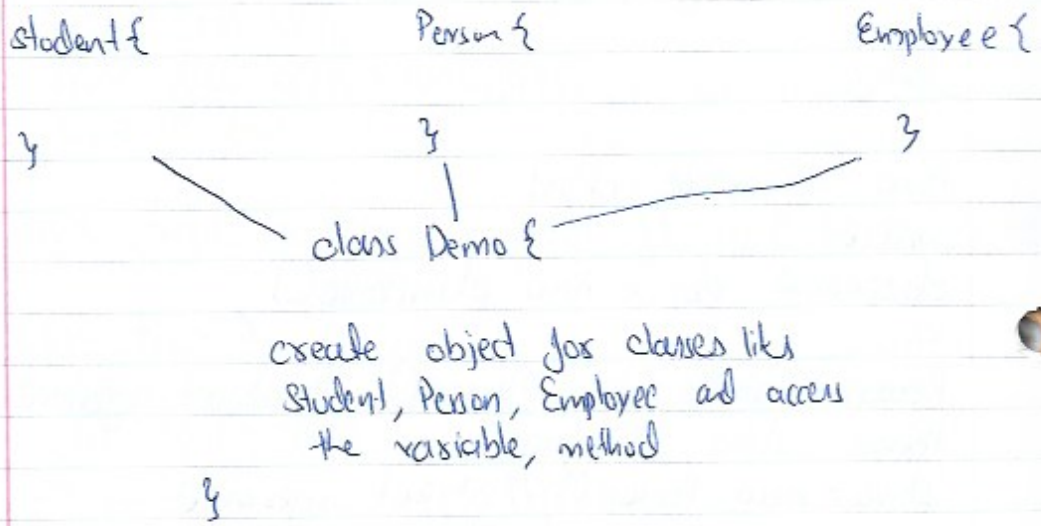
How to create object:
syntax:
classname obj = new classname();
ex:
Person rahul = new Person(); // object reference
Person shiva; // reference
shiva = new Person(); // object reference

note: In java the memory is allocated at runtime
Three different ways to import
→ import java.util.*
import java.util.TreeSet;
java.util.TreeSet ts = new java.util.TreeSet();

float b = 3.4f;

Output method in java:

System.out.println("welcome to java programming");



Types of variables:

1. primitive variable // int a = 10;
2. non primitive variable // Person rahul = new Person();

The primitive variables are classified

jvm:
→ method area (static)
→ heap area (object)
→ stack area (local)
→ pc
→ native area

java.io.*
java.lang.* (jvm import by default lang package)
java.util.*

static String clg;
if it is common to all the object

- Write 2 programs

```
class Student {
int id;
float per;
static String clg;
}
```

int → 0
String → null
boolean → false

```
class Employee {
 int empid;
 String designation;
 static String company_address;
}
class VariableDemo {
 public static void main (String args[ ] ) {
```

→ methods, overriding, abstract

→ operators

```
class  Car {
        int carid;
        String model;
        static String dealer_address;
}
class     VariableDemo {
     public static void main(String args[]) {
          Car bmw = new Car();
          bmw.carid = 10;
          bmw.model = "seriesNew";
          Car.dealer_address = "commerce, tx";
```

Day 2

java:
jvm → It converts the byte code to machine code

java consists of packages

packages consists of classes.
ArrayList → import java.util.*
classes consists of variables and methods

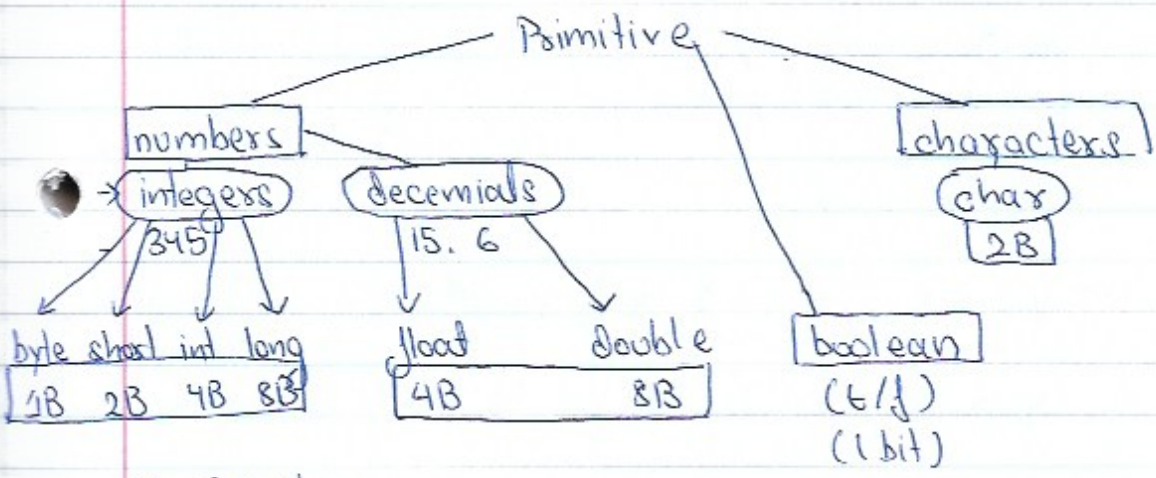
```
class Person {
int age;
void eat() {
}
}
```

if we want to access the variable (age)

- Data Types
  Classification of information

Data Types are of two types:

1) Primitive
2) Non Primitive



Non-Primitive

- Array
- String
  Class
-

Operators and Assignments:

1 increment / decrement:

pre-increment: ++y
post-increment: y++

pre-decrement = --y
post-decrement = y--

```
int x = 4, y;

y = ++x ; //x = 5 and y = 5

y = x++ ; //x = 5 and y = 4
```

```
int x=4, y;
y = --x ; x=3 and y = 3
y = x-- ; x=3 and y=4
```

2 Arithmetic Operators:
(+, -, *, /, %)

MAX(max_type, type of a, type of b)
byte + byte = int
byte + short = int
int + long = long
long + float = float
double + char = double
char + char = int

ASCII - A(65) a(97)

3 Relational Operators :
(<, >, >=, <=)

4 Equality Operators :
a = 10 (==, !=)
a = b

5 Bitwise Operators :
- &, |, ^(XOR)

& T & T → True
| F | F → F
^ T | F → T F | F → F T | T → F

Type casting:

- Converting one datatype to another datatype

1) Implicit
2) ~~Explicit~~

(1B) byte → (2B) short → (4B) int → (8B) long →
(4B) float → (8B) double
|
(2B) char

2 Explicit

(1B) byte ← (2B) short ← (4B) int ← (8B) long ← (4B) float
|
(2B) char
(8B) double

Assignment Operators:

There are 3 types of assignment operators.

1 simple assignment operators ex: int a = 10
2 chained assignment operators
3 compound assignment operators

```
int a, b, c, d;
a=b=c=d=20;
```

compound assignment:

```
+=
a+= 30
a=a+30
%=
&=
|=
^=
```

Evaluation order (*, /, +, -)

```
x=1+2*3+4*5/6;
x= 1+6+20/6;
x= 1+6+3;
x= 10
```

- Control Statements:

1. simple if
2. if else
3. nested if else
4. if else ladder

- Static
- Non-Static

Method: A set of instructions written to perform a task.

1. pre defined
2. users defined

Every method has two properties:

1. return type
2. parameters

```
return_type method(par1, ..parN) {

}
```

```
int sum() {
int a=10, b=20, c;
c=a+b;
return c;
}
```

void indicates we are not transfering the value.

Based on these propesties the methods are classified into four types.

ex: void menu()

1. Method without return type and without parameters
2. Method without return type and with parameters
3. Method with return type and without parameters
4. Method with return type and with parameters

- with return type i.e only one value can be transfered from sub block to main block.

- with parameters i.e many values are transfered from main to sub block

```
# class Restaurant { void menu (){ s.o.p ("1. Starters, 2. Main Course")
}
}
class Customer {
  public static void main (String arg[]){
        Restaurant kfc = new Restaurant ();
        kfc.menu();
  }
}
```

```
void takeOrder (String order) {
    System.out.println ("order given "+ order);
}
```

Main-> kfc.takeOrder ("Desert");

```
String serveFood() {
    return "Desert";
}
```

Main -> String plate = kfc.serveFood();
System.out.println(plate);

```
String payBill (int amount) {
    System.out.println("Amount given is " + amount);
    return "paid";
}
```

Main -> System.out.println("Bill " + kfc.payBill(432));

```
class University {
    void listOfColleges() {
      1. jntu . 2.00 3. Cbit
    }
    void selectCollege (int clg1.) {

    }
    int seatAlloted() {

    }
    String payFee(int fees) {
        return "confirmed";
    }
}
class student {
   public static void main(String args[]) {
```

Day 3

a % 2 == 0

Nested if else:

syntax

if we want to write multiple conditions.

```
if cond-1 :
  if cond-2 :
```

if else ladder:

```
if (cond-1)
  st-1;
else if (cond-2)
  st-2;
else if (cond-n)
  st-n;
else
  st;
```

int a, b, c; n requirement - 1

↓

n => 3 - 1 = 2

hi

loop:

1. while loop
2. do while
3. for loop
4. nested for loop

- initialization;
  while (condition) ← checks again if true execute again ← if true executes
  {
  st-1;
  st-n;
  }

amount = 10000
deposit = 2000
amount = amount + deposit

n = 5
1 = i
2
3
4
5

sum = 15

marks = 565
bonus = 5
marks = marks + bonus

n = 5
5 * 4 * 3 * 2 * 1

n = 5, x = 1 ;

```
while (n >= 1) {
    S.o.p(i);
    x = x * n;
    n--  ;
}
```

Assignment

n = 5 i = 1

5 * 1 = 5

5 * 2 = 10

....

5 * 10 = 50

10) 123 (12 → /
120
3 → %

```
int n = 123, x = 0, rem = 0;

while(n > 0) {
  rem = n % 10;
  x = x + rem;
  n = n/10;
}
```

Assignment n = 153
cube
1 + 125 + 27

sum = 153
if number is armstrong display armstrong

→ exit level:
initialization;

do {
st-1;
st-n;
}
while (condition)

→ for loop

++/--

for(initialise; condition; logic)

output in java:
= = = = = = = = = =

Every method is in a class.
class is in package.
java.io.*
java.lang.*
java.util.*
java.io.PrintStream
class PrintStream {
void println();
}

Object ⟶ method

java.lang.System
class System {
static PrintStream out;
}
System.out.println();

> javap java.io.PrintStream

> javap java.lang.System

Input in java:
java.io.InputStream
class InputStream {

}

java.lang.System ⟵
class System {
static InputStream in;
}

method area (static)
heap area (object)

- System.in : It contains the input object in String format

Scanner: It converts the String format object to required format.

```
java.util.Scanner
class Scanner {
   int nextInt();
   Long nextLong();
   float nextFloat();
   String next();
}
```

```
Scanner sc = new Scanner(System.in);
int a;
a = sc.nextInt();
float b = sc.nextFloat();
String name = sc.next();
```

import java.util.Scanner;

polymorphism :

poly → many
morphism → forms / states / properties

(1) method overloading
(2) method overriding

- method overloading:
method can be written same with different type of parameters, no of parameters or different order of parameter

method name is same with different order of parameters.

```
for (initialisation; condition-1; ++/--)
{
for (initialisation; cond-2; ++/--)
{
st-1;
}
}
```

2 × 3 = 4

```
for (i=1; i<=2; i++) {
for (j=1; j<=3; j++) {
  S.op("Hi");
}
}
```

1) hi
2) hi
3) hi
4) hi

i	j
1	1 2 3 4 5
2	1 2 3 4 5
3	1 2 3 4 5
4	1 2 3 4 5
5	1 2 3 4 5

```
i = 1     1
    2     1 2
    3     1 2 3
    4     1 2 3 4
    5     1 2 3 4 5
```

```
for(int i=1; i<=5; i++){
    for(int j=1; j<=5; i++){
        cop
    }
      System.out.println(j);
}
```

```
i    j
1    1
2    2 2
3    3 3 3
4    4 4 4 4
5    5 5 5 5 5
```

```
*
* *
* * *
* * * *
* * * *
```

```
i = 1     1
    2     2 3
    3     4 5 6
    4     7 8 9 10
```

j= 5 4 3 2 1 i = 1

5 4 3 2 2

5 4 3 3

5 4 4

5 5

```
for(i=1; i<=5; i++){
for(j=5; j>=i; j--){
```

j 1 2 3 4 5 i = 5

1 2 3 4 = 4

1 2 3 = 3

1 2 = 2

1 = 1

j<=i

i=

1 1

0 1 2

0 0 0 3

0 0 0 0 4

1 0 1 0 1 5

Assignment

```
1 2 3 4 5        i = 5
2 3 4 5            4
3 4 5              3
4 5                2
5                  1
```

i — 3 for loop

```
____ 1           1
___ 1 2          2
__ 1 2 3         3
_ 1 2 3 4        4
1 2 3 4 5        5
```

Array:
Collection of homogenous elements.
Collection of similar elements.

How to declare an array:

1 datatype ar [];

int a [] = new int [5];

2 datatype [] ar ;
int a;
a = new int [5];

3 datatype []ar ;

int x[];
int [] x;
int []x;

1) single dimensional array : a[]
2) two dimensional array : a[][]

java 8
for each loop:

for (datatype variable : array / collection) {

}

methods types
→ return
→ parameters

- Constructor:

classname obj = new constructor();

It takes part in object creation and it initialise the variables.

properties of a constructor:

1) Constructor name is same as classname.
2) Constructor has only one property ie parameters.
3) Constructor can't have any return type ie not even void
4) Constructor gets executed when object is created ie. once per object creation.

1 zero argument constructor (without parameters)
2 parameterized

Constructor overloading

Day 5

SQL: (structured query language)

oracle -> database
mysql
mongodb
sql servers

- create

- alter
  - modify
  - add
  - rename
  - drop

rename

drop

> connect system/manager

Rules of SQL:

1) Dublicate table names are not allowed
2) Dublicate colums in a table are not allowed
3) Every command or quesy ends with a semicolon.
4) The character values can be given in single quotes itself.
5) SQL is case insenitive i.e. what ever case case we write it will take upper case.

Datatypes: classification of information.

numberss(size):
It can store without decimal values.
ex: id number (10)
The size limit is 38. the size indicates the number of digits.

number(precision, scale):
It can store the decimal values
ex:

76234 .234

percentage number(5, 3)

76.234

- characters:
  char(size):
  It can store only characters.
  ex: name char(30)
  The size is limited to 2000.

- varchar(size)/varchar2(size):
  It can store alpha numberic values also
  ex: rollno varchar(20)
  The size is limited to 4000

Date: To store the system date
Time: time
blob: binary
clob: To store the information in character objet (like files, images)

• create table <table_name> (<column_name 1>
<data_type1>, ..., <column_NameN><data_typeN>);

ex

> create table jnit (id number(10), name varchar(20),
salary numbe(110));

alter :

modify :

syntax :

alter table <table_name> modify (<column_name 1>
<new_data_type1>, ..., <column_nameN><new_data_
typeN>);

ex

> alter table jnit modify id varchar(10);

* add

> alter table jnit add doj date;

> alter table jnit add (address varchar2(20), email
varchar 2(30));

rename :

alter table <table_name> rename <old_column> to <new_column> ;

alter table jnit rename column doj to joining

drop :

alter table <table_name> drop column <col_name>;

alter table jnit drop column joining ;

> desc jnit ;

> alter table jnit drop (address, email) ;

rename jnit to jnitinc ;

→ username
→ password

child/derived/sub parent/base/super

Sum extends Square

a set of instruction written

- method: to perform a set of tasks

input()
sqr()

```
int a;
a * a;
```

Square

accept()
add()

```
int b;
a + b;
```

Sum extends Square

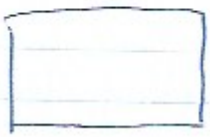
Single level inheritance

Tiger access animals and mammals

Electronic Vehicle

Mobile Car

Samsung Audi



Mammals

Animals

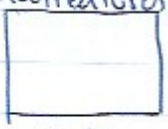
Tigers

Car

start()

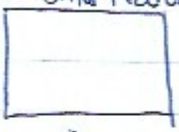
↙ superclass

audiFeatures()



Audi

bmwFeatures()



Bmw

Day 6

DML: (Data Manipulation Language)

> Insert:
All columns Single row:

> Insert into jnit values (1023, 'bob')

- All Columns Multiple rows

> insert into jnit values (&id, '&name', &salary);

> /

- Particular Colums Single row:

> insert into jnit(id, name) values (1029, 'abc').

- Particular Columns Multiple rows:

> insert into jnit (id, name) values (&id, &name');

update:

Entire column update:

> update <table_name> set <col_name1> = <value1>, ... ;

> update jnit set doj = sysdate;

> update jnit set doj = '25-aug-2020';

- Particular row single column update

> update jnit set salary = 20000 where id = 1026;

- Particular row multiple column update:

> update jnit set salary = 32000, name = 'ramesh'
where id = 1026;

- Multiple rows single column update:

> update jnit set salary = 25000 where id in (1029, 1030);

> update jnit set salary = salary + 5000;

Delete:

> delete from jnit;

Particular record delete

> delete from jnit where id = 1038;

Multiple record delete

> delete from jnit where id in (1029, 1026);

Select

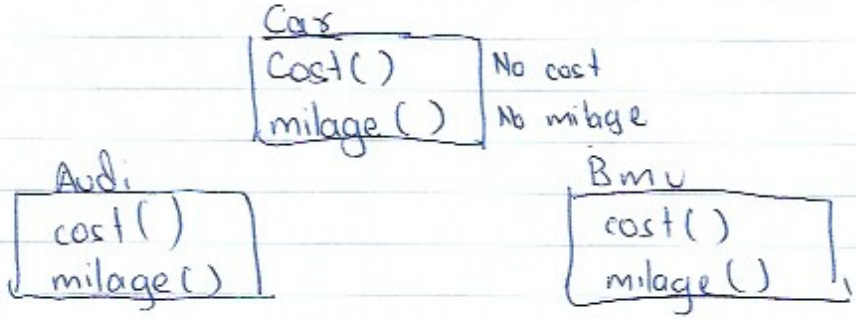
> select * from jnit;

> select * from jnit where id = 1025;

> select id, salary from jnit;

> select

- Method overriding:
method name can be same with same type of parameters or same order of parameters or same no of parameter.



Rules of overriding:
1. Separate classes are required for overriding
2. Inheritance relationship is mandatory
3. Seperate objects are required for overriding
4.

class casting:
converting one class object to another class

upcasting
downcasting

upcasting:

A super class reference storing the sub class object.

```
Car x = new Audi();
x = new Bmw();
```

Day 7

select * from tab

> select * max(masks) from student;

> select * min(masks) from student;

> select * sum (masks) from student;

> select * avg (masks) from student;

> select * count(masks) from student;

> select max(masks), section from student group by section;

group functions :

max( )
min( )
sum( )
avg ( )
count( )

* along with grasp functions we can't have where clause

→

→ having works only with group by

* constrains :

1) unique: it doesn't allow the dublicate but it allows null values
2) not null: it doesn't allow null values but it allows dublicate values.
3) check : based on condition it will allow the value.
4) primary key: It is the combination of unique and not null.
   It doesn't allow no null values and dublicates

* TCL:
  rollback
  commit

temporary [primary memory] window
secondary memory

until we write commit or we close the

> rollback;

> commit;

abstract class:

Collection of abstract method or concrete methods.

abstract method: the method which doesn't have any implementation or body.

concrete methods: The method which has implementation or body.

For abstract class we can't create the object.

```
abstract class Animal {
    abstract void lifespan();
    void breath() {
    }
```

Assignment

Day 8

Hisite constructor call using super

> select * from student where marks = (select max(marks) from student);

inner joins

Day 9

method name must be same with different type of parameters or order of parameters or number of parameters.

method overriding
method name must be same with same type of parameters or order of parameters or number of parameters

method: A set of instructions that perform an action.

method has two properities:
1 return type
2 parameters

instance variables the memory is occupied in heap area

which variables must be declared as static?
The variable whose value is common to all the objects.
static occupies method area.
static depends on classname.

jvm. it convert byte to machine code

classcasting: converting one class object into another class object

upcasting: sub class object is converted into super class object i.e upcasting.

- abstract class:
  collection of abstract methods and concrete methods.

- abstract methods: The method which doesn't have any implementation.

- concrete method: The method which has implementation or body.

- interface :- It is a pure abstract class.
- A abstract class which contains all the abstrad methods instead of using abstract class we can use interface.
- Interface all the methods are by default abstract and public.
- A class must implements an interface, a class cannot extend an interface
- Interface supports multiple interitance
- If a class implements an interface then all the abstract methods must be overrided.

```
class A {
void sum(){
}
}
```

```
class B {
void sum(){
}
}
```

```
class C extends A, B {
public static void main(String args[]){
C x = new C();
x.sum();
}
}
```

7 If we are overriding the interface abstract methods then all the overrided methods must be declared as public.

- Electronics (Assignments)

- Constructor
- Super
- this

8 we can't

Inner classes. A class written within a class is known as inner class.

1 Non static inner class
2 static inner class

Day 10

- Exception Handling:

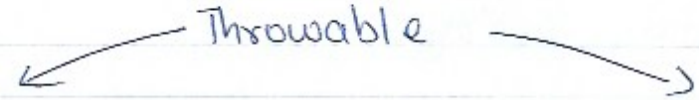
In a normal flow of execution, if an abnormal situation occurs then jvm will terminate the program by throwing the exception class object.

- Exception Handling:

```
try {
    risky code;
}
catch(ExceptionClassname reference) {
msg related to exception
}
```

we have two predefined classes.

1. Class
2. Object: Object class is the super class for all the classes.



Exception

RuntimeException
ArithmeticException
NullpointerException
ArrayIndexOutOfBoundException
ClassCastException

IOException

EOFException

Error

- Syntax error:
  if we miss any (;) or ({) in the code it will give syntax error

- logical error:
  a=10, b=5
  if(a>b)
  S.o.p("b is max");

- Runtime Exception:

- Exception is classified into two types

1. Un-checked Exception : The exception which are unknown to the compilers are known as Un-checked Exceptions.
2. checked Exception: The exception which are known to the compilers then those are checked exceptions.

> javap java.lang.RuntimeException

Exception handling means to skip the instruction with the exception and execute the remaining part of the program correctly.
or
To stop the inappropriate

1)
```
try {
   riskycode;
}
catch (ExceptionClass ref) {
   msg
}
```

2)
```
try {
   riskcode;
   try {
      riskcode;
   }
   catch( ExceptionClass ref) {
      msg
   }
```

finally:

- finally is a block which is used to deallocate the memory or close the objects.

ex:

jdbc:

```
Connection con;
con.close();
```

- finally is a block which gets executed irrespective of the exception
- throws is the keyword used to generate
- To handle the checked exceptions we can use throws and try, catch.
- To handle the unchecked exceptions

Day 11

- Interface support multiple inheritance
- Exception handling in bit questions
- india bix java

Collection

Wrapper class:

Is java pure oops?
because of the primitive data types java is not pure oops it is fully oops.

pure oops:
java script
python

For every data type we have pre defined class
int → Integer

```
int a = 10;
Integer i = new Integer(a)
```

Integer x = 5; //Autoboxing

Generic → unknown type

Collection

A group of individual objects

note: with the help of generics and wrapper classes.

Set

It is also a group of individual objects

→ It doesn't allow duplicate value.

→ It allows null values

→ It allows heterogenous elements

Hashset: It follows random order

LinkedHashSet: It follows insertion order

TreeSet: It follows ascending order
It doesn't allow hetrogenous elements

first(); Assignment
last();
lower(E);
floor(E);
ceiling(E);
higher(E);
pollFirst();
pollLast();

List:
It is also a group of individual objects
→ It allows duplicate values
→ It allows null values
→ It allows hetrogenous elements

ArrayList:
It follows insertion order
select * from employee;

LinkedList:
select * from employee where id=1023;