18/12/2023

a1, a2, a3	m1(), m2()
b1, b2, b3	m3(), m4()

```
Package P1;
Public class ExampleE1
{
    Private int a;
    Private int b=20;
    Public int geta()
    {
        return a;
    }
    Public int getb()
    {
        return b;
    }
}
```

```
Package P1;
Public class ExampleE2
{
    Public static void main
                (String[] args)
    {
        ExampleE1 ref = new
                ExampleE1();
        int c = ref.geta();
        int c1 = ref.getb();
    }
}
```

* ~~Acb~~ we can achive encapsulation with the help of methods known as getters & setters.

1) getters are method or group of methods which are design to return the value to the caller method.

Note:- getter method should always be Public & ~~sho~~ it must have return type

22/12/2023

* Inheritance :-

- Inheritance is a mechanism to inherit Properties of Parent class & child class.
- We can achive inheritance betn two classes by using keyword extends.

Ex.

```
Class Parent
{
  int a=10;
 Public void add()
  {
    S.o.p.(a);
  }
  Parent()
  {
    S.o.p.("Hii");
  }
}
```

```
class child extends Paren
{
  int b=10;
 Public void subtract(
  {
    S.o.p.(b);
  }
 child()
 {  super();
   S.o.p("Hii"from child
 }
Public Static void main(
{
  child ref= new child
 int bl=ref.a;
   ref.add();
 intref.S.o.p.(ref.b);
   ref.subtract();
}
}
```

* Types of Inheritance :-

1) Single (Level) Inheritance :-

This is a type of inheritance where we inherit one class inherits the Property of another class (only 1). By using keyword extends.
object class is Considered as super most Parent class in java.

Ex.
```
class A Extends object
 {
   int a=10;
   P. V. multiply ()
    {
     }
  P. S. V. main (String [] args)
   {
     A ref = new A();
     ref. multiply();
   }
```

with the help of ref we can access child (class A) Properties as well as Parent (object) properties.

2) Multi Level Inheritance :-

It is a type of inheritance sub-class inherits Property of super class (more than one class)..

Ex.
```
class Vechicle extends object
 {

 }
class Car extends Vechicle.
{
 }
Class TATA extends Car
 { }
```

③ Hierarchical Inheritance :-

It is also type of inheritance which allows to inherit one class into many sub-class.

Ex.

```
Class Gfg
{
  int subjectno = 20;
  P. void study()
  {
    S.o.p.("from study");
  }
  P. void Solveproblem()
  {
    S.o.p. ("from solveproblem");
  }
}
Class Ranjeet extends Gfg
{
  String subject = "DSA";
  P void studylang()
  {
    S.o.p. ("from studylang");
  }
  P. S. void main (String [] args)
  {
    new Ranjeet =ref= new Ranjeet();
    ref. studylang();
    ref. study.
    ref. Solve problem
  }
}
```

④ Multiple Inheritance :-

It is a type of relⁿ betⁿ classes where one class (child class) inherit property of more than one class (parent class) at a same time. by using keyword extends.

Class A	class B	Class C extends A, B
{	{	{
		super();
		}
}	}	

* Compile time Polymorphism :-

- If a member (variable, method) has one name many forms here Compiler decides which form has to be executed at Compile time is known as Compile time polymorphism

25/12/2023

M T W T F S S
Page No:
Date: 25/12/23
YOUVA

* Polymorphism :-

Polymorphism is a mechanism through which we can perform multiple task with same entity.

Types of Polymorphism

1. Compile time polymorphism
   1. method overloading
   2. Constructor overloading
   3. Constructor chaining
   4. Method Chaining
2. Run time Polymorphism
   1. Method overriding
   2. method shadowing
   3. Method chaining.

1)* Method Overloading :-

It is a process of keeping/creating multiple methods whos name is same throughout the class. but they differ in formal arguments.

By :-

1. Number or length of length of formal argument.
2. Diffrent datatype of formal argument.
3. Sequence of ~~ff~~ the formal argument.

(Note :- ~~Then~~ It will only work when datatype of formal arguments are of diff. types.)

② Constructor Overloading :-

It is a Process of creating multiple Constructor inside same class having same name as that of class name. but they differ in formal arguments of each Constructor.

they should differ in :-

① Length of formal argument

② datatype of formal argument & sequence of f.A.

Ex.

```
class Santa
{
  Private int a;
  Santa(int a)
  {
    this.a= a;
  }
  Santa(double a)
  {
    this.a+= a;
  }
  Santa(short b)
  {
    S.o.p.(b);
  }
  P.S.V.main(String[] args)
  {
    Santa bhai = new Santa(10);
    Santa bhai1 = new Santa(10.5);
  }
}
```

26/12/2023

M T W T F S S
Page No.:
Date:
YOUVA

③ Method Chaining:-

~~Metho~~ It is a Concept of calling multiple methods with same refrence. in same line/ Single line.

```
Class Method_Chaining
{
  Public int M1()
   {
     S.o.P(" From M1");
     return 10;
   }
  Public method_chaining M2()
   {
     return this;
   }
  Public method_chaining M3()
   {
     return this;
   }
  Public method_chaining M4(int a, int b)
   {
       S.o.P (b*a);
       return this;
   }
  Public Static void main (String[] args)
   {
     method_chaining ref = new method_chaining();
      ref. M2(). M4(10,20). M3(). M1();
   }
}
```

* Advantages of Compile-time Polymorphism :-

① It is easy to fix bug /error in Compile time Poly.
② It is increase code readability and maintability
③ It helps in writting more efficient Code.
④ Using this Concept we can achive static binding which that happen at Compile time

2) Run Time Polymorphism :-

It is a mechanism through which we can execute a method of child class ~~through~~ by Calling ~~by~~ through Parent refrence
That we have
① method overiding :-

It Can be achive betn two classes by using
① Is a relationship (extends)
② upCasting

```
Class Car
{
  int Price = 150;
  Public ~~Str~~ void drive():
  {
     S.o.p("drive");
  }
  Public void drift()
  {
     S.o.p("from drift");
  }
```

21/12/2023

M T W T F S S
Page No.:
Date:
YOUVA

* UpCasting :-

string the refrence / address of object of child class in Parent type variable.

Runtime 2 · obj = new Runtime1 (); → C.C.
P.C.

↳ We can only Call / Access properties of Parent class But in case of child and Parent have same method the Parent class method will be executed.

* DownCasting :- when we Convert Parent type to child type we do downcasting become with child type refrence we can ~~occur~~ access / Call Properties (methods / variables) & Parent Class as well as child class

runtime 1 obj2 = (runtime1) obj;

28/12/23

② Method Shadowing :-

It's a concept of opps through which we create same static method in two diff class having is a relationship where the method of parent class is called & also gets executed from parent class with the help of upcasted refrence.

Package Example

```
Public class MS
{
    static Public void talk()
    {
        S.o.p.(" from talk of MS");
    }
    Public Static void run()
    {
        S.o.p.(" from run of MS");
    }
    Public void eat()
    {
        S.o.p.("from eat of ms");
    }
    Public void call()
    {
        S.o.p(" from Call of MS");
    }
}
```

Package Example

```
Public class Ms1
{
    Public void call()
    {
        S.o.p("From Call of Ms1");
    }
    Public void dance()
    {
        S.o.p("from dance of Ms1");
    }
    Public static void run()
    {
        S.o.p("from run of Ms1");
    }
    Public static void main(String[] args)
    {
        MS ref = new Ms1();
        ref.call(); // Method Overiding
        ref.run(); // Method shadowing.
    }
}
```

29/12/2023

* Complete class :-

Complete class is a class having method which are Comple (Declaration + Body).
Body / method block is also known as implementation.

25/29/12/2023

* Abstract class :-

- Abstract class is also known as incomplete class Because it can/it allows to create incomplete method. (only method declaration).
- It is important to declare class as abstract with keyword (abstract)
- we can create abstract method inside a abstract class by using non-access specifier as abstract.
- Abstract method are incomplete method which doesn't not have implementation or body it contain only declaration

Ex

Abstract class

```
abstract class A
{
    P. abstract int add();
        Abstract method
}
```

Normal class

```
Class A2
{
    P. void add2()
    {
        S.O.P. ("Hi");
    }
}
```

* What members we can have inside abstract class

1. Static variable
2. Non-Static variable
3. Static Block
4. Non-static Block
5. Static method
6. Non-static method
7. Constructor
8. Abstract method.

* Variable can be final.

* In abstract class we cannot create object of that abstract class.
- Why :- we can't create abstract class as we know abstract class is incomplete blue print (incomplete features) So object can't be created for undetermined features. So we can't access any non-static members inside main method while using abstract class.
(Because we need object to call non-static members) we can execute abstract class with the help of main / static block.

* Abstraction :-
Abstraction is a process through which we can hide the implementation or body of a method & display only the feature to user.

Advantages of Abstract class :-

1. Security.
2. we can give various implementation to particular Feature.
3. Code Reuseability & Readebility
4. Increase Scalability of Code.
5. Easy Bug Fixing.

* Advantage of Runtime Polymorphism

1. We can achive ~~#~~ generalization through Run time Polymorphism.
2. we can achive abstraction through —II—
3. In run time polymorphism the method block which gets executed depends upon who's & which class object has been created.
4. we can achive code Reusability with the help of run time polymorphism.
5. wee can make a behavior (method) work differently with the help of diff. implementation.

* Interface :-

Interface is Compo ent of java which allows us to achive multiple inheritance & abstraction. Interface doesn't allow to create its own object.

~~Int~~ Because interface doesn't support Constructor.

* features of Interface :-

1. we can have static method inside an interface.
2. (main method can be created inside a interface)
3. we can execute an inteface ~~im~~ explicitly also.
4. we can't have Constructor inside interface.
5. we can't create object of ~~of~~ interface.
6. we can't have intializer inside a interface. (intitalizer - static, non-static block, Constructor)
7. In inteface we cannot have non-static method.
8. methods without non-access modifier are considered as abstract by default.

8/01/2024

M T W T F S S
Page No.
Date
YOUVA

* object class:-

hashCode is a method which is written in C & C++ if you use java to find hexadecimal language.

① toString()
② getClass()
③ hashCode()
④ equals()
} method Imp.

object class :-

object class which is present in lang Package of system library & Considered as Supermost Parent class - every class will inherit or extend object class directly or indirectly.

Q. why object class is mandatory to inherit?

① It has Constructor which Predefined by the developer & does not Contain super() statement in it thus, stopping the Constructor chaining. Process will be keep going.
② it Contain important method such as toString, hashcode, getClass & other 11 methods which is responsible for running or executing class.

* Method & Constructor

① declaration of object class

```
Public class object
{

}
```

② Constructor of object class

```
Public object() {

}
```

③ getClass():-

Declaration :- Public final native <u>class<?></u> getClass();
return type

work:- ① it gives the class info inside getClass

② also it returns the refrence of object address representation of

<u>class</u> A.
keyword Class name

③ through which we can access non-static methods of class of getclass through Concept of method chaining.

getclass returning refrence of class

④ hashCode :-

Declaration :- Public native int hashCode();

work:- ① it return unique represntation of object in form of int.

② it return representation of object it does not return exact address of object.

③ hashCode implementation has been given in C ↓ Ctt as it is a native method.

4/01/2024

M T W T F S S
Page No.:
Date:
YOUVA

* equals method ():-

It is used to compare string values because doubles equal to operator (==) compares ref. of object rather than comparing actual value.

Ex.

```
String a = "Hii";
String b = "Hii";
String c = new string("Hii");
S.o.p(a==b); // true
S.o.p (a==c); // false
S.o.p (a.equals(c)); // true
```

* toString() :-

```
Public String toString(){
    return getclass().getName() +"@" + Integer.toHexString
                                        (hashCode());
}
```

getclass(): present inside object class, return type class
getName(): return name of class
hashCode(): int type formal argument

- It is a method which return representation of object address.
- By using getclass method then getName method
  +"@" +
  Integer.toHexString(int a); → static method which present inside Integer class
  Wrapper class
  return hexstring representation of hashcode.

```
Public int hashCode()
{
    return price + tax;
}
}
```

01/2024

Throwable ~~class~~ :-

Throwable is a parent class of Expection & Error which is responsible for running or throwing errors & exception, they are thrown by JVM implicitly

Throwable class
- Exception
- Error

Error :- It is a predefined class which is child class of throwable class ~~of~~ & error class is responsible for throwing / giving errors.
* at runtime

Public Error extends Throwable.

Example :-
1. Stackoverflow → when stack area memory is full then we get stack overflow error.
2. Abstract method Error → when we do not give implement° of abstract method of inherited class.
3. illegal access Error :- when we try to get / access private member we get illegal access error
4. class Formal Error :- when JVM cannot read class file it throw class format error.

Akash

'02'

* Exception:-

Exception is a aptrupt or sudden stop in program while executing it.

Exception
- ① Compile Time Exception (checked)
- ② RunTime Exception (unchecked)

1) Run-Time ~~Exep~~ Exception (Unchecked Exception) :-

It is a subclass of exception which gives exception at runtime.

In this exception the exception are thrown by JVM implicitly

Example :-

1) Arithmetic Exception

```
class Crow
{
    p-s-v-main(String[] args)
    {
        s.o.p(10/.1);
        s.o.p(10/.2);
        s.o.p(10/.3);
        s.o.p(10/.0); // JVM throws ArithmeticException
    } // can't execute next line of code
} // exception by using try & catch block.
```

09/01/2024

M T W T F S S
Page No.:
Date:
YOUVA

* Customized Exception :-

It is a process of creating user defined exception where user can name ~~in~~ is his exception as well as he can add some ~~meaing~~ meaning full msg in exception. Using keyword throw we can create ~~Example:~~ our Customized Exception (user defined Exception)

- We need to define our own class which must extend exception or any child of Exception
  ~~Public~~ ↳(the name of user defined Exception will your defined class names.)

Example :-

```
Public class M2 extends M3
{
 Public Static void main(String[]args)
 {
    try{
     int a= 10;
     int b= 0;
     if(a*b == 0)
     {
        throw new M3("Can't Multiply by zero");
     }
    }
    Catch (Exception e)
    {
        S.o.p (e getMessage());
    }
    So.p ("Done");
 }
}
```

```
Public class M3 extends RuntimeException
{
   M3 (String msg)
   {
      super(msg);
   }
```

```
   M2(String b)
   {
     super(b);
   }
```

M T W T F S S
Page No.:
Date:
YOUVA

* try Block :-

syntax:
try{
}

① It is a block of code which can be executed to check whether the given code in block (try) is throwing any Exception or not.
② If it does not throw any Exception try block will be executed successfully else the line in which we get exception will stop the termination. the program there & then its goes for catch block if it is present. else its goes for finally block.

* Catch Block :-

syntax:-
catch()
{
}

① It is block of code which is used to catch exception which is thrown by JVM from try block
② we can use catch block only in combination of with try block.

* Finally Block :-

① It is a block of code which is used in combination with try block the code which we write in finally block ~~if we have~~ will execute even if we have any kind of Exception.
② It can't be use with catch block, it always we with try block

Syntax :-

```
finally
{
}
```

* throw :-

It is a keyword which is used to throw object of a class which extend exception or any of their subclasses such that to create user defined / customized Exception.

* Checked Exception :-

Checked Exception when Compiler is aware of Exception which is going to occur at runtime, it displays a message file for handling the specified Exception.

Example :- (1) FileReader → FileNotFound.

* throws :-

It is a keyword which is used method declaration to declare exception or propoget exception so that we can handle exception at run time & Compile time.

* Diffrence betn throws & try and catch & throw

	throws	throw	try and Catch
①	It is used with method declaration.	It is used with object of class.	It can be used within any block.
②	It is used to handle exception as well as declare.	It is used for creating user defined Exception	It is also used to handle exception.
③	It is not 100% dependable.	-	It is 100% Dependable

④	syntax :-
	P.S.V. add()throws FileNotFoundExc.
	{
	//write your
	// Code
	}
		syntax :-
		Class M2
		{
		~~P.S.V.~~
			syntax :-
			try{ // code.
			}
			Catch( )
			{ // code.
			}

Instanceof operator:-

It is used for checking if the refrence which we have provided contains object of given class

Syntax :-

object refrence instance of classname.

return type of instance of operator is boolean.

Example.

```
class AC {
  Public static void main(String[] args)
   {
      AC ref = new AC();
      boolean result = ref instance of AC;
        S.o.p(result);
      S.o.p. (ref instance of AC);
   }
}
```

* ClassCast Exception :-

It is sudden stop in Program when we try to downcast an object refrence which is not upcasted

Example :-

```
Package Wert;
Public class Name2
 {
    int b=10;
    Nam2()
    {

    }
 }
```

```
Package Wert;
Public class Name extends Name2
```

12/01/2024

* Drawbacks of Variable :-

① Variable Can't store multiple value it is very complex to perform mathematical operation with more number of variable.
② It is very ~~tuff~~ tough for Programmer to remember variable name when Variables are in large no./huge no.
③ It increase Code Complexity
④ It reduce Code readability
⑤ It reduce Code. maintainability
⑥ Also it Consume more line of Code.

* Array :-

To overcome disadvantage of variable we have Concept of Array.

M T W T F S S
Page No.:
Date:
YOUVA

What is an array?

- Array is Data structure of Continues memory blocks. (Memory Allocation) @which can be used for storing diffrent number of values of similar datatype.
- Also It is fixed in size.
- Syntax of array :-

datatype [] variable ; → (String [] args)

① Declaration → datatype variable [] ; → (String args [])

② Initialization

→ ① Single line initialization

syntax :- int a [] = {1, 2, 3, 4, 5, 6, 5, 8, 9} ;

→ ② datatype variablename [] = new datatype [size] ;

int array [] = new int [4] ;

array [0] = 100; | array [3] = 300;

10 Create an array of int type an store 10 element in array

* Jagged Array :-

A Array which is having diffrent number.

24/01/24 * String :-

→ ① What is String?
→ ② Properties of String class
→ ③ What is immutability
→ ④

* What is String?

- String is a datatype of non-primitive type (class type). mainly string is used to store Combination of multiple Caharacter
- ~~For eg.~~ String should be coritten in double queats

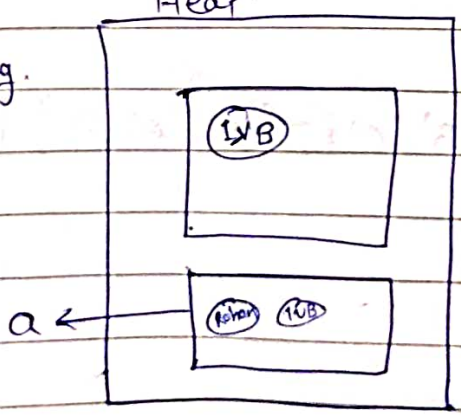
For. eg. "BuzzNumber" = String a ;
↳ Combination of various character.

* How to store value in String :-

→ ① String a = "Rohan."; → literals (String Constant) → object will store in SCP
→ ② String className = new String ('IVB'); → object will be created inside heap area also as scp if value is not present in scp

Heap.

Eg.



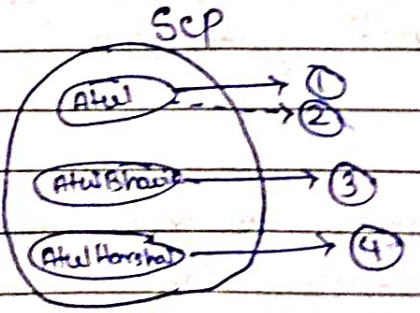
* Properties of string class:

① String class is final class.
② String is immutable.
③ mainly all members of string class are also final.
④ It is threadsafe.

* What is immutability?

- Immutability is a Property of string due to which we Cannot update the previous or the current object.
- Example

String name = "Atul"; new object created in SCP.
String nameStud = "Atul"; → Point to previous object because it has same value as previous object
String name2 = name + "Bhavik"; new object will be created. with the value name+Bhavik. inside SCP.
~~String~~
name_Stud = name_Stud + "Harshal";
Due to updation of value. / Concatination of value will create new object name_stud Point this object.



25/04/2024

* String :-
  - ① why string is immutable
  - ② Advantages of immutability
  - ③ Disadvantage.

* Why String immutable.

String is immutable because it is mandatory to create a new object while doing updation hence in string updation is done only in newly created object this allows string to be constant through multiple operations.

* Advantage of immutability :-

① Thread safe :-
② Predictable behaviour
③ Easy error fixing
④ Reusability.

* Disadvantages :-

① It is not recommended to use immutable object for preparing frequent operation
② It is not synchronized.

① write a program to take first username & last name from user. in fname concate the

* concat() method :-

29/01/2024

M T W T F S S
Page No:
Date:
YOUVA

	String	String Buffer	String Builder
Object	It has immutable object	It has mutable object	It has mutable object.
Space	It Consumes more space as it is immutable	It will Consume less space as Compared to string because it is mutable	It will Consume less space as Compared to String because it is mutable.
Thread Safe	String is threadsafe bc's of immutability	It is also threadsafe and synchronized	It is not threadsafe and a synchronized.
Efficiency	String is efficient ~~you~~ when you take previous obj.	It is efficient when we frequently update the value of object.	It is efficient when we use frequently updation of value.
	string is slower as Compare to String Buffer	String Buffer is Slower as Compare to String Builder	String Builder is faster as Compared to String Buffer.

* Collection :-

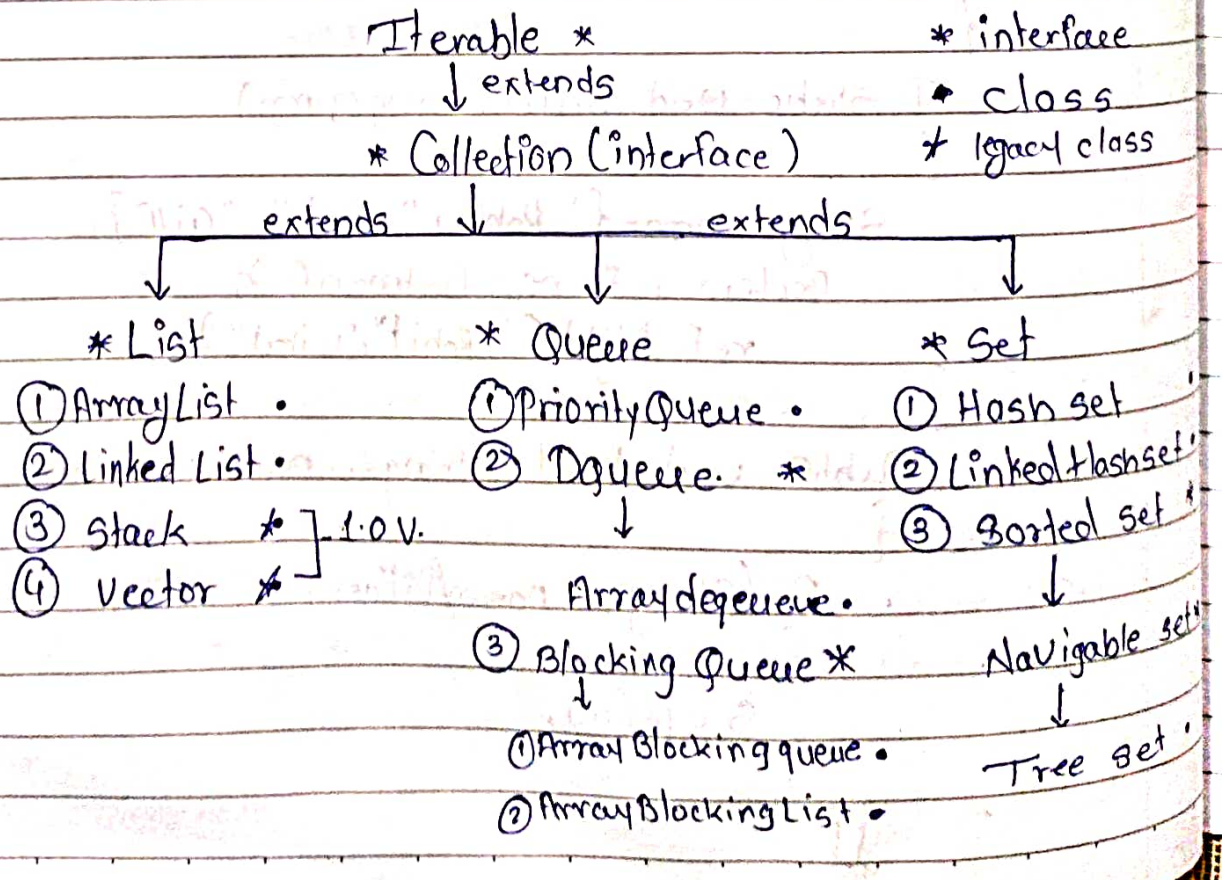
Collection in java is nothing but representation of group object

* why do we need Collection:-

Collection overcome all the draeobacks of array

i) Drawbackes :- Collection can be graeable in nature (array fixed in size).
ii) Collection can Contain heterogeneaes data.
iii) Collection have predefined helping method.
iv) Collection can be used to increase the readability of the program.

* Hierarchy of Collection :-



3/02/2024

M T W T F S S
Page No.:
Date:
YOUVA

* Collection Framework :-

- It is combination of various class & interface which can be used to contain diffrent type of object. (heterogenous data)

* Interface $\xrightarrow{\text{implements}}$ class
* class $\xrightarrow{\text{extends}}$ class
* Interface $\xrightarrow{\text{extends}}$ Interface

* Collection is interface which is used for providing common behaviour (methods) for its child interfaces.

Common behaviour (Methods)

① size :- int size();
returns size of Collection of int data.

② add :-

① boolean add(Object e) :- return type boolean → True (if successfully added.) / → False.. else.
It is used to add object in Collection.

② boolean addAll(Collection c) :-
It is used for adding one Collection to another Collection.

6/02/2024

* Implementation DS of ~~for~~ ArrayList is array (growable array)
* The array in ArrayList grows implicitly by using grow() method.
* Using ArrayList we can perform fetching or retrival task very much efficiently.
* this is because of ArrayList implements RandomAccess interface. Since it is always recomanded the use array list for any feething or retrival operation.

* There are 3 Constructor of ArrayList.

① Public ArrayList() → It will create ArrayList (growable array) of size 10.
② Public ArrayList(int Capacity) → It will create arraylist of given (capacity) size.
Ex. ArrayList ref = new ArrayList(20);
③ Public ArrayList(Collection c); → used for Constructing another type of Collection to Array List.
Ex. ArrayList ref = new ArrayList(ref2);

* ref. ensureCapacity(10); :- Increase. To given Capacity as a Parameter.

* ref. removeIF(Condition) → Predicate Fun^c.

07/02/2024

* LinkedList :-

LinkdList is implementation class of list Interface and internal DS of clinkelist is doubly linked list.
We use link list (Recommended) for addition & Removal of element from Collection.

* How to Declare Link List :-

① LinkList ref = new LinkList();
② LinkList <Integer> ref1 = new LinkList();
③ ~~int~~ List ref2 = new LinkList();
List <String> ref3 = new LinkList();

* Constructor of Link List :-

① Public LinkList();

② Public LinkList(Collection c);

* Method of Dqueue :-

① offer(object e) :- Return type b → Add element at last.

② offerFrist (object e) :- Return type boolean → Add element at frist po.

③ offerLast (object e) : Return type → boolean → Insert the specified element at the last of List

④ Peek() :- Return type → object → for fetching value. from link list

⑤ PeekFrist() :- Return type → object → For fetching ~~va~~ frist value from list

⑥ PeekLast() :- Return type → object → For fetching last value. from list.

⑦ poll() :- fetch 1st element from list after fetching it will remov that element from list. → Return type → object.

⑧ PollFrist() : Return type → object →

⑨ poll Last() : Fetch last element from list after fetching remove it. Return type → object.

⑩ Pop() :- Return type → obj → Retrive frist element in list and return it to the programmer as well as remove that element from list.

⑪ Push() :- Return type → obj → Add frist element in list

③ Split Iterator
④ List Iterator
① Iterable
② Iterator

Write a program add marks of students of class & retrive the marks ~~grea~~ grade As per the given grade value.

90 - 100 → A*
80 - 90 → A
70 - 80 - B
60 - 70 → B
50 - 60 → C
450 - 50 → d
35 - 40 → P
0 - 35 → f

08/02/2024

* Vector :-

- Vector is a legacy class of java which is present from version 1.0 also it implements list interface hence containing all implementations of list interface method as well as its own method.
- Internelly it implements growable array as its DS.
- Vector is synchronized.

* ~~Stack~~ :-

Constructor of Vector class

① Public Vector();
② Public Vector(int Capacity);
③ Public Vector(int intialCapacity, int incrementCapacity);
④ Public Vector(Collection e);

* Stack :-

2/02/2024

* Queue :-

Def^n :- Queue interface allows data to be stored in format of FIFO Concept. Queue is child interface of Collection.

- Queue Does not support Null value.
- Does Queue allows duplicate value :- yes
- Does it maintain insertion order :- No
- Does it allow heterogenous data :- No.

* Queue Method :-

we can add only homogeneous type data.

① boolean add(object a) :- This method is used to add element in Queue. return type → boolean. if null is added in queue you will get NullPointerException.

② boolean offer(object b) :- offer method also use for adding values in queue. return type → boolean.

③ object remove() :- It removes and retrive head element of queue. In case if queue is empty it return NoSuchElementFoundException.

④ object Poll() :- It removes and retrive head element of queue. In Case if queue is empty it returns null.

⑤ object Peek() :- It retrive head element of queue. In Case if queue is empty it returns null.

⑥ object Element() :- It retrive head element of queue. In Case if queue is empty

* How to Create object of Queue.

① PriorityQueue ref = new priorityQueue();

② Queue ref1 = new PriorityQueue();

14/02/2023

* Cursor Classes :-

① Enumeration ③ List iterator

② iterator ④ split iterator

* What does it do?

Def^n :- It works similar to that of looping statement and are applicable for all collection-classes.

1. ~~Enumeration~~ Enumeration
   ↳ package java.util.

Applicable → collection → vector & sh

Method of Enumeration

① hasMoreElement ():-

② nextElement ():-

```
class EnumerationEx.
{
 p.s. void main(String[] a)
 {
   Vector ref = new Vector();
      ref.add('a');
      ref.add('b');
   Enumeration ref1 = ref.elements();
```

```
// for(; ref1.hasMoreElement();)
   while (ref1.hasMoreElement())
   {
      char b = (character) ref1.nextElement();
         s.o.p.(b);
   }
 }
}
```

Drawback :-

* Applicable for legcy classes & it contains only 2 methods which is helpful in iteration but it does not allow to update & remove values.

* Enumeration is unidirectional.

2. Iterator :-
   - Applicable for any collection.
   - It was similar to that of Enumeration but it allow to remove values from Collection.

Methods of Iterator

① hasNext() :- return type → boolean

② next() :- returntype → object.

③ remove() :- return type → void.

```
class Iterator Ex
{
 p.s. void main (String[] a)
 {
    ArrayList ref = new ArrayList();
        ref.add(10);
        ref.add(12);
        ref.add(13);
        ref.add(14);
  Iterator ref2 = ref1.iterator();
   while(ref2.hasNext())
   {
      int a = (Integer) next();
       if(a%2!=0)
       {
          ref2.remove();
       }
    }
    s.o.p.(ref);
  }
}
```

15/02/2023

List Iterator :-

* What is List iterator?
- It is Interface ~~where~~ which extends Iterator and perform iteration in forward & backward direction and also can be used to perform updation of list. (Applicable only for list implementing classes).

* Methods of List iterator :-

RT

1. hasNext() → boolean
2. next() → object
3. hasPrevious() → boolean
4. Previous() → object
5. remove() → void

RT

6. set(object) → void
7. add(object) → void
8. nextIndex() → int
9. ~~Pre~~PreviousIndex() → int

* Note for add() :- We cannot use add() method inside any loop.

Ex. class Example {
P.S.V.m.(String [] a)
{
ArrayList ref = new ArrayList();
for(int i=10; i<=20; i++)
{
ref.add(i);
}
ListIterator ref1 = ~~new~~ ref.~~ite~~listIterator();
ListIterator ref2 = ref.listIterator(3);
while(ref1.hasNext())
{
S.o.p.(ref1.nextIndex());
ref1.next();
}
while(ref2.hasPrevious())
{
S.o.p.(ref2.~~previous~~PreviousIndex());
ref2.Previ
}

17/02/2024

List	Queue.
1. Store heterogenous element	Store homogenus element.
2. Insertion order is maintain	Insertion order is not maintain.
3. Duplicate are allowed.	Duplicate are allowed.
4. Random Access. (Indexing Allow).	No indexing (FIFO)

* ~~DQueue~~ :- Deque :-

Dqueue is a child interface of queue which allows elements to be in sorted order until its unique.

It also allows to add & remove element from last & from frist.

Dqueue standed Doubleendeed Queue.

Dqueue allows duplicate element.

Implementing class of Dqueue is Array Deque.

* ~~Whyen~~ When to use ~~Dqueue~~. ~~Dq~~ Deque.

we can use dequeue when

(1) we need to store homogeneus date.

(2) we can also use it storing data in sorted manner

(3) we can Dqueue to add element & remove element both side.

17/02/2024

List	Queue
1. Store heterogenous element	Store homogenous element.
2. Insertion order is maintain	Insertion order is not maintain.
3. Duplicate are allowed.	Duplicate are allowed.
4. Random Access. (Indexing Allow).	No indexing (FIFO)

19/0

* ~~DQueue~~ Deque :-
Deque is a child interface of queue which allows elements to be in sorted order until its unique.
It also allows to add & remove element from last & from frist.
Dqueue standed Doubleendeed Queue.
Dqueue allows duplicate element.
Implementing class of Dqueue is Array Deque.

20/0

* ~~*~~ Whyes to use ~~Dqueue~~. ~~Dq~~ Deque.
we can use dequeue when
(1) we need to store homogeneus data.
(2) we can also use it storing data in sorted manner
(3) we can Dqueue to add element & remove element both side.

* Deque Methods :-

① addFrist (obj)	⑤ ~~Remo~~ remove frist ()
② addlast (obj)	⑥ remove last()
③ offerFrist (obj)	⑦ PollFrist ()
④ offer Last (obj)	⑧ Poll Last()
⑨ get Frist ()	⑩ getlast()
⑪ PeckFrist ()	⑫ Peek last
⑬ Remove FristOccurence (obj)	⑭ Remove LastOccurence (obj)

10/02/2024

	List	Queue	~~Set~~ set.
①	insertion order main-tain.	It maintain only when it has only unique ele.	It doesn't maintain insertion order.
②	Duplicate are allowed	Duplicate are allowed	NO Duplicate allow
③	null is also allowed	No null values.	null is allowed only!
④	Hetrogeneus	Homogeneus data	(*) Heterogeneus data
⑤	Indexing Concept	FIFO & LIFO Concept.	

20/02/2024

* Set :-

Set interface is a child interface of Collection which does not allow add duplicate elements and also can be used for
① Sorting element (ASC/DESC)
② It Can be used for maintaining inerstion order (LinkHashSet)
③ In set HashSet, LinkHashSet both are not sorted.
④ In HashSet the elemnt order according hashCode.
- In LinkHash set it doen't maintain order because it follow insertion order.

In tree Set we can Sort the ~~acco~~ element according to our requirement (Asc / DEsc)

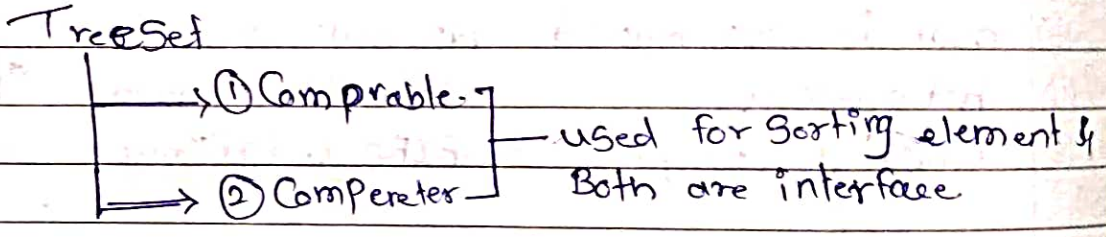
④ By default element in tree Set are ~~arrange~~ sorted in Asc order.

* Implementing class of Set interface :-

1) Treeset 2) ~~LinkH~~ LinkedHashSet 3) HashSet

1) Tree Set :-

- Tree set is implementing class of Set which allows to Store unique element in either Asc order or DEsc order
- In TreeSet there are two interfaces which are used for sorting element in TreeSet.



Example

```
import SetInterface;
import java.util.*;
Public class TreeSetExample
public static void main(String[] args)
{ Set ref = new TreeSet();
    ref.add(10);
    ref.add(11);
    ref.add(1);
    ref.add(14);
    ref.add(6);
    ref.add(8);
    ref.add(9);
    ref.add(10);

    S.o.p(ref);
}
}
```

* Comparable :-
- It is used for sorting element in Asc order.
- Comparable interface is used implicitly by tree set to sort the element in Asc order.
- The sorting of element are done by CompareTo() method.
- It is present inside java.lang Package.

* ~~Comparator~~ :-

* Comparator :-
- Comparator is a interface which present in util package which enables user or programer to sort element either Asc or DSFC order.
- we have two abstract method Compare and equals
  ① int Compare (object1, object2)
  ② boolean equals(object obj)
- The implementation of Compare method is defined by user in implementing class but implementation of equals() method is not define or given in implementing class because implementing class inherit equals() method implementation from object class

* What is load Factor?
- Load factor is ratio/percentage of set capacity at which new set with increased capacity is created & previous elements are stored & get provid larger capacity to add more element.
- Load factor (fill ratio) is always within the rage of 0-1. & it can be return in form of decimal. foreg. 0.75%.
- The deault Fill ratio of HashSet is 0.75%

* Constructor of HashSet

① Public HashSet (int initialCapacity)
② Public HashSet(int initialCapacity, float load factor);
③ Public HashSet (Collection a). ④ Public HashSet()

* LinkedHashSet :-
- LinkedhashSet is a implementing class of set interface which allocas to maintain insertion order
- It does not sort the element in any order.

* Constructor of LinkedhashSet.

① Public LinkedHashSet (int initialCapacity, Float loadFactor)
② Public LinkedHashSet (int initialCapacity)
③ Public LinkedHashSet()
④ Public LinkedHashSet(Collection a)

* Method of LinkedHashSet:

① splitIterator()
② addFirst(obj) → RT. → void
③ addLast(obj) → RT. → void.
④ getFirst()
⑤ getLast()
⑥ RemoveFirst()
⑦ removeLast() → RT. → object
⑧ reverse()

* Map:-
Map is interface which allows to add elements in form of key & value Pairs.

* What is key?
- key is unique element which is present inside entry which contains or points to another refrence of value.

* What is Value:-
- values are data which is associated with individual key.
- values can be duplicated.

* where to use map? → when we have to store value in Pair.

* Implementing class of MAP. → interface.
  - → ① HashMap
  - → ② LinkedHashMap
  - → ③ TreeMap

* HashMap :-
- It is a implementing class of map interface which allows to store elements in key-value pair.
- Internal DS of HashMap is HashTable.

Constructor of HashMap :-

1. Public HashMap()
2. public HashMap (initialCapacity)
3. Public HashMap(Map m);

* Methods of HashMap :-

	Return type
① int size()	int
② boolean isEmpty()	boolean
~~③ boolean get~~	
③ get(object k)	object
④ Containskey(ob	

* Method of HashMap

	Return type
① int size()	int
② boolean isEmpty()	boolean
③ obj get (object k)	object
④ boolean Containskey(object o)	boolean
⑤ obj put (k, v)	object
⑥ void putAll (map m)	void
⑦ obj remove (object key)	object
⑧ void clear()	void
⑨ boolean Contains(object value)	boolean
⑩ Set keySet	set

⑪ Collection values() Collection

* why we return Collection ?
values method return o/p in Collection because the values of a map can be duplicate.

29/02/24

	HashMap	LinkedHashMap	TreeMap
①	Interal Data Structure is HashTable	Internal Data structure is HashTable & Deque.	Internal data structure is tree (Red Black tree)
②	no duplicate keys can have duplicate values.	no duplicate keys duplicate values can are allowed.	no duplicate keys
③	null is allow only once, values can be duplicate so null is allowed.	null is allowed.	No null is allowed.
④	It does not follow insertion order.	It main tain the insertion order.	NO insertion order.
⑤	Heterogenious data is stored	Heterogenious data. (object)	Homogenious as well as Heterogenious.