JAVA

Different Kind of Java Edition

* Java Standard Edition
* Java Micro Edition
* Java Entrepreneur Edition

@ methods Are named block of codes used to reduce code complexity and reduce number of lines of codes

@ 3 steps in java file execution

* Source file creation \*(.java file)
* Compiling the source file checking for errors. This will give compile time error \*(.class file).
* Executing the complies file which is of “byte codes”

@ Class keyword in java says that it is the beginning of the java code.

@ Byte code generated during the compile time, is neither understandable by humans nor by the computers

@ instead they are understood by the “JRE”

@JRE acts as a interpreter

@ Once the file is complied it can be run in any os due to Byte Codes and JRE hence it is called as “Platform Independent Language”

OBJECT

* Objects are elements of program that has some data which is also known as “States”.
* Objects also have behaviors which means that they can perform some operations intern called as methods

KEYWORDS

* Words which have predefined meanings
* There are 50(keywords) or 53(3 literals eg: true, false and null )
* Java is a case sensitive language

IDENTIFIERS

* Giving names to the elements of the program
* RULES

@ It should start with an alphabet

@ It can’t start with the numbers

@ Identifiers are also case sensitive

@ we can’t use key words ah identifiers

@ \_ and & are the only special characters

@ Space between the identifiers are not allowed

NOTE : If we declare class as public the we have to save the file name with its class name

VARIABLES

* These are name memory locations
* It is a piece of memory with a name
* Variables should have data type

TO-TYPES

* Local and global variable
* Local = primitive & reference
* Global = primitive & reference

Static & Non-static static & non-static

* Variable created in side the scope of class = global variable
* Variable declared in the scope of method is called local variable

DATA-TYPE

* It is used to specify which kind of data the variable is storing
* It is used for validation purpose
* Two kinds of data types are Primitive and Reference Data Type

@ Primitive Data Type

* Data type which is specified by the compiler or by the inbuilt programming is called as primitive data type
* 8 types of primitive data type

@ byte : 4-bits , range = -128 to 127

@ short : 2bytes , 8-bits, range = -32768 to 32767

@ long : 8 byte , 64-bits

@ int : 4 byte, 32-bits

@ double : 8 bytes ,64-bits

@ float : 4 bytes , 32-bits

@ char : 2-byte

@ Boolean : true or false

>Declaring the variables : Syntax : data type variable ;

> initialization Variable = value;

>utilization is using the variable inside the program

OPERATORS

* An operator is a special symbol or keyword that is used to designate a numerical operation or some other type of operation
* These operations can be performed on one or more than one values called as operands

@ Arithmetic operation

+ : Addition [ Additive arithmetic operator ]

- : Subtraction [------‘’ -----------‘’ ----------------- ]

\* : Multiplication [ Arithmetic Multiplicative Operator ]

/ : Division [------------------------‘’------------------------]

% : modules [ -------------------‘’----------------------------]

Note : ASCII values from ‘A’ to ‘Z’ is 65 to 90 and ‘a’ to ‘z’ is 97 to 122

COMPONED OPERATOR :

+= is used to add the numbers and store the value in the corresponding value Eg: X+= Y,-= ,/= ,%=

INCRIMENT (++)AND DECRIMENT(- -):

@they are called as unary operator

@ if they are used before the variable then it is called as PRE - INCRIMENT or PRE-DECREMENT( Eg : ++x)

PRE – INCREMENT

1. Increment
2. Substitute
3. Utilize

POST – INCREMENT

1. Substitute
2. Utilize
3. increment

@ if they are used after the variable then it is called as POST – INCRIMENT or POST – DECREMENT

RELATIONAL OPERATOR

* == : Equal to (true when both the sides are equal)
* != : Not equal to (both the sides are not equal to)
* < : less than (Returns true the RHS is greater the LHS)
* > : Grater than (Returns true when LHS is greater than RHS)
* >= : Grater than or equal to (Returns true when LHS is greater than or equal to RHS)
* <= : Less than or equal to (Returns true when LHS is less than or Equal to RHS)

LOGICAL OPERATORS

* Used to validate to or more conditions at a time
* Kinds of logical operators are
* @ NOT – it is a unary operator ; the condition will be true if the input is false

@ AND – it is a binary operator ; if both the condition are true then the output is true

@ OR – it is a binary operator ; if any one of the condition is true the output will be true

BIT – WISE OPERATOR

* Output of bitwise operator is integer
* @ bit\_wise OR = If any one is true the output is true

@ bit\_wise AND = if both the operators are true then output is true

@ bit\_wise XOR = if both the bits are different then output is true

FLOW CONTROLE STARTEMENTS

1] IF-ELSE : If the condition is true , conditions inside the if-block will b executed ;else the codes outside the if

block will be executed

To the codes outside the if block

Codes in the IF block

True(if)

Conditi-on

False(else)

Code outside the if else block

Codes outside the IF block

2] ELSE-IF-LADDER : if there is multiple conditions to be evaluated then we use if else-if ladder

=> we go for if else-if when we have more than one Boolean condition to be satisfied

condition

Code outside the if else-if block

true

false

true

Code inside if else-if block

Code in sife if block

Code inside else block

3] SWITCH : if the Boolean condition to be checked of “=” then we can go for switch statements

Break statement: It is used to break the execution flow when ever needed

Default statement : => it is not mandatory that it should be specified at the last if in case we are giving it in the middle then we have to give brake statement after that

LOOPING-STATMENT

1] FOR – LOOP :

Syntax : for ( initialization ; test condition ; increment/decrement )

{

Statement-1;

Statement-2;

}

* Writing a statement after in finite loop it throws a error saying “unreachable statement”.

2] WHILE : Block of codes will be executed only if the condition is satisfied

Without any increment or decrement condition while loop is a infinite loop

Syntax : while (condition)

{

Statement-1;

.

.

Statement - n ;

}

3] DO-WHILE : if we want to execute the condition at least once and later to check for the condition

Syntax : do

{

Statements ;

} while ( condition )

4] forEach : loops the this which are sub classes of irritable

METHODS

* Which has both method definition and method declaration
* Methods usually are made up of access-specifiers and access-modifier and return type with method name and argument list
* Syntax : Access-Specifier Access-modifier Return-type Method-name (argument - list)

{

statments

}

* Return type can be of any kind of primitive data type
* Access modifier is used to give visibility of the class
* At the end we have to give return statement in the created method
* Naming convention for method is it should start with lowercase alphabet
* A method will only execute if it is been called by passing the required arguments
* A method which is been called is known as called method
* A method which is calling is known as calling method
* It creates problems when ever we try to create a method inside a method it is a good practice to declare a method in a class
* CLASS LOADER :it will check for named method

ARRAYS

* It is a collection of homogeneous data
* Index of an array will always start from zero
* Maximum value of index is (Size-1)

DECLARATION

* Declaration the array : Syntax : data type []Array\_name ;

Data\_type[] Array\_name;

Data\_type Array\_name[] ; // mostly not used

CREATION

* Syntax : Array\_name = new data\_type [Size]

INITIALIZATION

* Synatx : Array\_name [index] = value ;

ARRAY MANUPILATION

* Is to perform some operations on the arrays
* Array Out Of Bound Exception is the error we will get when the array is overflowed or if we try to print a variable which is not present in the array
* Length() = it is used to calculate the length of the array
* PROGRAME: If that element is present it should print the index

Else it should print the its not present

STRINGS

* Sting is sequence of character
* String values are immutable
* String is not considered in primitive data type where as it is considered on among user definied data type
* Storing data in the string internally creates a character array
* It is to overcome the storing limit prob in array
* But string can store n number of characters but for best practice character more than 2000 is not preferred
* Syntax :1] String variable = “ value”;

2]String Str ;

Str = “hello”;

3] String str = new String(“Hello”);

* To character array is a method which is used to convert the string to a character

REFERENCE-TYPES

* It is based on java class rather than a primitive data type
* A reference type can be ba
* sed on pre-defined classes in java or classes defined by programmer or developer
* New keyword used to create the reference variable has the address of that particular object
* Eg : String str1 = new String(“ ”)

STATIC AND NON-STATIC

STATIC

* Any member of the class that has be declared by the key word static then it is called as a static member
* To call a static members in another static class we don’t have to create a object of that class
* They are accessible just by their class name and variable name

NON-STATIC

* Members of a class which are not specified as static we call them as non-static members of the class
* To call the non static members we have to create a instance of that member
* Creating static members in static class in not allowed as well as in non static class in sot allowed
* If a variable declared in class it has accessibility throughout the class

CONSTRUCTORS

* It is a special kind of method without return type
* Every class should have a constructor
* If we are not creating it compliler will create a inbuilt constructor
* These have the name same as the defined class name and without return type specified
* Constructor can’t be static
* To view .class file we use “javap” file name
* These are used to create a objects
* No argument constructor and default constructor both are not same

CONSTRUCTOR-OVERLOADING

* Creating same constructor with same name but with different input arguments

NOTE : In order to call a method from another package we have to import

Public keyword is used to control the visibility of the class members

ACCESS-SECIFIERS

1. PUBLIC
   * It can be used by any other class or package
2. PROTECTED
   * It can not be used to access in any other package
3. DEFULT(Package level)
   * If there is no access specifier then it is by default package level
   * It can’t be used in any other package but only inside its own package
4. PRIVATE
   * It can’t be used in any other class expect its own

ASSOCIATIONS

* This is java is relationship between to different classes
* Types of association
  + One to one
  + One to many and Many to one
  + Many to many
* HAS – RELATION and IS- RELATION
* HAS-A REALTION
  + Aggregation
  + Composition

HAS-A-RELATION

AGGREGATION

* One class is not dependent on other class

COMPOSITION

* Existence of one class is very much dependent on other class

IS-A- RELATION (INHERITANCE)

* The process by which one class acquires the properties and functionalities of another class is called inheritance
* The aim of inheritance is to provide re-usability of code.
* Class whose properties or functionalities whose properties are inherited by other class is known as “parent-class or Super-class or Base-class”
* Class that inherits properties or functionalities from another class is known as “child class or Sub-class or Derived- class”
* For inheritance in java we have to make use of “Extends ” keyword
* You can access the properties of both Super-class and sub-class using object of sub-class
* FINAL CLASSES can’t be inherited, Final members of a super class can be inherited but can’t be changed
* PRIVATE members and constructers of super-class can’t be inherited
* Types of Inheritance
  + Single-inheritance
    - One class acquiring the properties from one single class
  + Multilevel-inheritance
    - One class inheriting from another class and that class inheriting from another class
  + Multiple-inheritance
    - It is not supported in java, because of diamond-problem
  + Hierarchical-inheritance
    - It is a mixture of to or more type of inheritance

SUPER-KEYWORD:

* It is used to access the key word from the immediate super class
* It refers to the object of immediate parent or super-class
* To access the data member of a parent class when both child and parent class have the data member with the same name
* If there is no super statement to call it automatically invoke a zero argument constructor from parent class

METHOD-OVERRIDING

* Declaring a method in a child class(over riding method) which has already been declared in parent class(over ridden method) is known as method over-riding
* Method over ridding is done to provide implementation specific to a child class
* Over-riding method should have same number of arguments and order of arguments

RULES

* Private methods can’t be printed
* Final members can be inherited but not overridden
* Static members can’t be inherited

ADVANTAGES

* You can provide implementation to the child class method without changing the code in the parent class

METHOD-OVERLOADING

* This is a feature in java which allows us to have same methods (with respect to name) in the single class more than once provided the arguments list differ
  + In number of parameters
  + Order of parameters
  + Data types of parameters
* They can have different return types and also different access specifier’s
* Method-overloading can be done irrespective of the static or non static methods
* Private methods can be overloaded
* Final method can be overloaded

@Program to perform basic arithmetic operations having different argument list

POLYMORPHISM

* Same function acting differently at different instence of time
* Two types
  + Run time polymorphism
  + Compile time polymorphism
* Method over-riding it is a type of run-time polymorphism(LATE-BINDING)
* Method overloading is one of the example for compile time polymorphism(EARLY-BINDING)

ABSTRACTION

* Hiding the implementation details and providing the services or functionalities
* Without caring about the actual functionality and concentrating on the output is called abstraction
* For example invoking a object, calling a println method, creating a main method etc are few examples for abstraction

ABSTRACT- CLASS

* Any class which is prefixed by the “Abstract” key-word is called as abstract class
* Any method that is declared with the abstract key word they are called as abstract method
* These does not have a method definition
* Abstract classes can have both abstract methods as well as concrete methods”
* Any class having a abstract method should be declared as ABSTRACR but vise versa is not true
* If a class has been declared as abstract then that class should be extended by any sub-class
* We can’t create object for abstract class but it can have a constructor
* To give the definition for a abstract methods we have to over ride but if the child class is also abstract then implementation of abstract methods becomes optional
* If a abstract class is extending another abstract class then there is no need to implement all the methods
* If a concrete class is extending another abstract class then we have to give implementation for all the abstract methods

INTERFACE

* These are kind of abstract class by default it will be abstract and public
* Implantation for the interface is given in the child class
* Class “IMPLIMENTS” another interface
* Methods inside a interface are by default abstract
* Interfaces does’t contain constructors
* It is possible to create a concrete class either it should be static or default keyword

Difference between interface and a abstract class

|  |  |
| --- | --- |
| ABSTRACT CLASS | INTERFACE |
| * It is a class but with abstract key word | * It is a type of class |
| * To declare a concert class inside abstract method there is no need for static or default | * If we want to declare a concrete class then it should be pre-fixed with static or default |
| * Multiple inheritance is not allowed | * Multiple inheritance is possible |
| * We can create constructor for abstract class | * We can’t have constructor for interface |
|  | * Data members of interface if final static and public |
|  |  |

* Types of interface
  + Functional interface
    - It should contain one abstract method
    - @FunctionalInterface is the decorator used
  + Marker interface
    - If there is no methods inside it then it is marker interface
    - Eg : cloneable ,Serializable,remote
  + Interface
    - Which has n number of abstract and concert methods
* Cloneable is a in built interface only used with clone method in java
* Serializable is used with the concepts in file handling

NOTE: Package other than java.lang should be imported in our class file

ENCAPSULATION

* Encapsulation is a method where we wrap up the data (data and function members) to single object
* Hiding the actual implementation
* Getters and setters are the public methods which are used to manipulate the data
* Return type of setter are always void were as getters will be of its data type
* They are called as plane old java object(POJO) or Been

PAKAGES

* Java API is also a set of packages Eg: java.lang
* Fully qualified call name : package name followed by the class name
* If we want to access a class with same name from different packages we have to import it once and the other we have to create object using “fully qualified class name”

FINAL

* One the value is initialized then the value of the variable can not be changed
* If we want to declare the final variables then they should be in block letters
* Final methods can’t be overridden and variables can’t be re-initialized
* Final classes can be inherited but can inherit from other classes(final classes can be sub class but not super class)

OBJECT-CLASS

* In java each and every class directly and indirectly inherites the properties of object class that is
* Object class is the super most class in java
* Each and every class either a pre-defined class or a user defined class is a child class of object
* Class is a blue-print of object

METHODS OF OBJECT-CLASS

* getClass() => returns the name of the class
* clone() =>using cloneable interface we can create a clone of the object
* equals() => before using this we have to override the hascode method
* toString()
* finalize() => used with garbage collection
* wait()-----------
* wait(long,int)
* wait(long) used in threads
* notify()
* notify(All)----------
* hashCode()=> gives the address of memory location of object
* class named as class inside a java API
* to access static members of a call we can use static import statement with \* next to it

STRINGS

* it is a mot value type, reference type it is not primitive type
* String literals are the inputs to the strings
* Concatenation can be done using + operator (Eg for polymorphism )
* Operator overloading is done in strings
* They are immutable
* The string contains two parts
  + Contestant pool => duplicate values are not applicable
  + Non constant pool=> duplicates values are accepted

STRING BULIDER AND STRING BUFFER

* The string is a very powerful class but not so efficient. Because string are immutable any methods of string class which modifies the string creates a new string object
* To overcome this problem java offers two alternatives those are “string buffers and string builders”
* Both are mirror images which has same methods and perform same functions
* The only difference between them is that string buffer is “THREAD SAFE“ where as string builders is not “THREAD SAFE”
* String builder efficiency is more than buffer we usually use builders in case of thread safety we go for buffer

SRTING BUFFER

* Public StringBuffer() and public StringBuffer()

Its constructor’s with zero arguments initial capacity is 16 characters this is same in case of “String-bilider”also.

* Public StringBuffer(int capacity) and public StringBuilder(int capacity)

Contructor a string buffer with no characters in it and the specified initial capacity

It will be creating the buffer object

* + @param capacity the initial capacity
  + @exception string buffer value with negative throghs a exception called as negativeArraySizeException
* Capacity() methods says the actual capacity of the string buffers (length of string + 16(which is initial capacity))
* Public StringBuffer(String str) and Public StringBuilder (String str)
  + Constructor a string buffer initialized to the contents of the specified string initial size of the string buffer is length of string + 16
* Public StringBuffer(CharSequence seq) and Public StringBuilder(CharSequence seq)
  + This is same as string but instead of string it is used char sequence and initial capcity is 16 + length of the string

|  |  |  |
| --- | --- | --- |
| STRING | STRING-BUILDER | STRING-BUFFER |
| * immutable | * mutable | * mutable |
|  | * Used when strings should mutable and concatenation of several characters should be done | * Used when we have to obtain synchronized |
|  |  |  |

EXCEPTION-HANDLING

* It is a block of code that can process the exception object only at the “RUN TIME”
* That is an error which happens at the run time it starts the exception handler for the method where error occurred and it moves on to the calling method and same continues
* Exception handling is overcoming the problems which comes during run time
* When ever exception is created it creates a “exception object” after that the program execution is stopped and JRE finds a program which can solve that program
* While displaying the error it follows method hierarch with file name exception name, with line number
* Creating an abject and showing it on the run time environment is called” throwing the exception”
* If the appropriate exception is fond then the exception objet is passed to the handler to process it . That is said as “catching the exception”
* NOTE: java exception is a framework that is used to handle runtime errors only complete time errors are not handles by the exception handling in java

Difference between exception handler and errors

* Exception usually have constructors but not methods

NOTE : to make our custom exceptions unchecked exceptions we extend “runtime exception”

|  |  |
| --- | --- |
| ERRORS | EXCEPTION |
| * included in java.lang.Erroe.pakages | * included in java.lang.ExceptionPakages |
| * they can not be handled | * they can be handles |
| * thses are caused mostly due to the environment in which it is running | * programs are it self responsible for the errors |
| * errors are unchecked type | * they are checked and unchecked type |

NOTE: checked type belongs to the exceptions which are checked during compile time. Unchecked belongs to the exceptions which are checked during run time

EXCEPTION HANDLING KEY-WORDS

* Exceptions will be thrown using “THROW ”keyword for this type we have to specify the condition about where it has to show the exception

“Throws” is used to throw the exception prier to the occurrence of the error. This might say the calling method that it might throw an error to the called method using ”THROWS”(class will be extend with throws key word)

* This is only used to show that the compiler might show an exception
* TRY –CATCH
  + Used for exception handling try block will be first at the end of the try block catch block begins
  + We have have multiple catch blocks after try block
  + If we are specifying the exceptions in catch blocks then the exception with exception catch block should be written at the last
* FINALLY
  + We can have try and catch block inside finally block also
  + This is only used with try and catch block the exception blocks the flow of execution but still if we want to continue the flow of execution we write the block of code inside the finally block
* TRY AND RESOURCES
  + Syntax :

Try( )

{

}catch( ){

}

THROWABLE

* Parent class of the all the exception and its two child objects are Errors and Exceptions
* Java exceptions are hierarchical and inheritance is used to categorized
* Two types are
  + Errors
    - It won’t be able to handled using programming
    - Eg: hardware failure, JVM crash or out of memory error so its not able to anticipate and recover from them
  + Exceptions
    - Checked Exceptions
      * These can be anticipate and can be overcome
      * We go for try and catch block
    - Runtime or Unchecked Exceptions
      * We go for try and finally exception
* Public String getMessage()
  + Present in throwable class
  + Returns a null message
* Public getLocalizedMessage()
  + To get the coustomozed exception

COLLECTONS

* It is a group of multiple items in a single unit example of collection classes are Vector, Stack, Hashtable, Array
* In Java 1.2 collection frame work was introduced
* Parts in collection
  + Interface
    - Collection frame work interface provide abstract data type that is
      * List
      * Set
      * Queue
      * Map
  + Implementation classes
    - collections provide core implementation classes using these we can create collections in java program
    - we can also create custom collection classes
* “Java.util.collection” is the root interface of collection framework . it contains few methods such as Size() ,iterator() ,add(),remove(),clear()
* Benefits of collection framework
  + Reduced development effort
  + Increases quality
  + Reusability and interoperability

JAVA COLLECTION INTERFACE

* Core collections are interfaces are genric(<E>) eg: public interface Collection<E>
* This is the syntax for generics and when we declare collection,we should use it to specify the type of object it can contain
* Generics helps in reducing run-time errors .
* Collection interface
  + This is the root of the collection hierarchies
  + We can find out the size of the collection eg: size and isEmpty(which gives Boolean values) to check weather collection has the element we have the method called as contains() add() to add the element and remove() to remove the element
  + We can also perform some bulk oparations such as
    - addAll(), removeAll(), clear() toArray() containsAll()
* Set interface
  + Is a collection which does’t contain duplicate values and contains atmost 1 null value
  + Eg: HashSet ,TreeSet, LinkedhashSet
* List Interface
  + It is the ordered collections also has the index value to that precise input and output operation can be carried out
  + they allow null values and duplicate elements more formally ,lists typically allow pairs of elements e1 and e2 such that e1.equals(e2),and they typically allow multiple null elements if they allow null elements at all
  + eg : ArrayList and linkeLlist we can add remove and replace the elements from the collection based on their index value unlike set
* Queue Interface
  + A collection designed for holding elements prior to processing
  + Besides basic collection operation other operations are insertion , extraction and inspection
  + They come in two forms they are
    - one throws an exception
    - other gives a special value in case if exception that is null or false
  + Different methods inside queue are
    - Boolean add(E e)
      * inserts the specified element into this queue if it possible to do so immediately violating capacity restriction
      * Exceptions are ClassCastException,NullPointerException,IlegalArgumentExceptin
    - offer(),
    - remove() ,
    - E poll()
      * Retrives and removes the element the head of this queue. This method differs frompeek only in that it throws an exception if the queue is empty
    - E element()
      * Retrives the element but not remove the head of this queue, and it throws an exception if the queue is empty
    - Peek()
      * Same as element but it does not throw any exception
* Map Interface
  + It helps us to create key and value pairs
  + Public interface Map<K,V> = key and value
  + It can not contain duplicate keys but can have duplicate values ,key can map to at most one value
  + There general purpose implementations are
    - HashMap
    - TreeMap
    - LinkedHashMap
  + Basics operations of map are get,put,containsKey,ContainsValue,size,isEmpty

JAVA COLLECTION CLASSES

* There are many implementation classes
* HashSet class
  + It is the basic implementation of the set interface that is backed by “HashMap”.
  + Inside the HashSet there is no iteration order and also permits null elements
  + We can set the initial capacity and load factor for this collections(load factor says that at what point of time it should increase the load capacity)
  + Constructors are
    - Public HashSet(){

Map = newHasMap<>();

}

Initial capacity is 16 and load factor is 0.75

* + - Public HashSet(Collection <? Extends E> c){

Map = new HashMap<>(Math.max((int)(c.size()/.75f)+1,16)) ; }

This constructs a new set containing

* + - Public HashSet(int initialCapacity , float loadFactor){

Map = new HashMap<>(initialCapcity,loadFactor); }

* + - Public hashSet(int initialCapcity)
    - Public HashSet(int initialCapacity , float loadFactor,Boolean dummy)
      * It will be backed by LinkedHashMap with specified initial capacity and load factor
* TreeSet
  + Comparable natural ordering in ascending order
  + All elemnt inserted into the set must be implemented in the interface
  + All such elements must be mutually cpmparable e1.compareTo(e2) must not throw a ClassCastException for any elements e1 and e2 in the set
  + Public Treeset()
  + Public Treeset(Comparator<? Super E > ccomparator){ this.(new TreeMap <>(comparator)); }
  + Public Treeset()

HashMap Class

* + Public V put (K key , v Value){

}

If we are using the key for the 1st time the return output is null

LIST

* ArrayList defult size of ArrayList is 10 and it is unsyncronized

Public ArrayList(int initialcapacity)

Public ArrayList()

Public void trimToSize =extra ize of the array will be trimed

Public void ensurecapacity(int initialCapcity) = to increase the capcity of the arrayList

* VectorList
  + It is a synchronized list and used for thread safety

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| collection | ordering | Key value | Duplicate Elements | Null Element | Thread Safe | Load Factor | Initial Capacity |
| ArrayList | Yes | No | Yes | Yes | No | 1 | 10 |
| linkedList | Yes | No | Yes | Yes | No |  |  |
| HashSet | No | No | No | Yes | No | 0.5 | 10 |
| TreeSet | Yes | No | No | No | No |  |  |
| HashMap | No | Yes | No | Yes | No | 0.75f | 16 |
| Treemap | Yes | Yes | No | No | No |  |  |
| Hashtable | No | Yes | No | No | Yes | No of key stored | Size of array |
| Vector | Yes | No | Yes | Yes | Yes | 1 | 10 |

JAVA LAMBDA EXPRESSIONS

* This is one of the new features of java standard edition 8
* It is a way to represent “Functional interface”
* The lambda expression is used to provide the implementation of the functional interface
* In case of lambda expression we don’t need to define the method again for providing the implementation
* Java lambda function is treated as a function so compiler does not create .class file.
* This is used only in case of functional interface bez this might create ambiguity if it is used with a method with more than I method definition
* Synatx : (argument-list)🡪 {body}
* RUNABLE INTERFACE it is a functional interface which is sub class of throwables

THREADS

* It is a single sequence of executable code in large program(small part of a bigger program)
* Main is also a thread which is executed 1st . But in java we can have multiple threads that can run parlellay (Eg: while browsing in a web page viewing the page and downloading the page simultaneously )
* This is called as “multithreading” in java
  + Multithread program have different parts of program which can run simultaneously
  + Allows concurrent execution of two or more parts of program
  + Its advantage is max usage of cpu
* THREAD CREACTION
  + They can be done in two ways(both are present in java.lang package)
    - By extending a class
      * We create a class which extends java.lang.Thread class that will override the run() available in the thrgreas class. We create an object of our new class and call start() to start the execution of the thread start() invokes the run() and the thread object
    - By implementing a runnable interface
      * We create a new class which implemnys java.lang.Runnable interface and override………
* 3 important property of thread
  + Thread name= used for identifying the threads
  + Thread id = its is a unique number which is created and assign by the “thread scheduler” to every single thread inorder to identify them uniquely
  + Thread priority = I used by the thread scheduler to decide the order of the execution by using setPriority( ).

Garbage collector: After execution of the program the de-referenced objects are discarded

Note: when to or more threads try’s to use same resource that is known as “RACE CONDITION”. This leads to data inconsistency

Race condition🡪 seen in multi environment condition one resource is used by more than one thread at a time.

THRAED-LIFE-CYCLE:

NOTIFY();NOTIFYALL();

YIELD();

SLEEP();

WAIT()

RUNNING

JOIN();START();

* A THREAD CAN BE I ONE FOLLOWING STATES:
  + NEW 🡪 a thread that has not yet started is in this state
  + RUNNABLE🡪 a thread executing in java is in the machine (start())
  + BLOCKED🡪 A thread that is blocked waiting for a monitor lock is in the

blocked (sleep(),join(),yield()).

* + WAITING🡪 a thread is said to be waiting when it is dead lock condition(wait(),notify(),notifyAll())
    - Dead lock 🡪 when to or more threads are locked or locked forever. To overcome this problem by using ITC “Inter Thread Communication” that is using (wait(),notify(),notifyAll()).
      * WAIT()🡪 it will pause the execution of thread and release the object lock
      * Notify()🡪 wakes up the first thread that called wait() on the same object
  + TERMINATED🡪 a thread that has exited is in this state. To stop explicitly we go for stop().

SYNCRONIZATION :

* To overcome problem of multi threading there is some kind of lock primitive. Only one thread can acquire a particular lock at any particular time. This can be achieved by using a keyword ”Synchronization ”

1] Difference b/w run() and start()

|  |  |
| --- | --- |
| RUN() | START() |
| * But if you call a run() directly the code inside the run() is executed without creating a new thread | * This creates a new thread and code inside run() is executed |
| * It can be called n number of times This does’t give any exception | * You can not call start() twice on thread object it gives IllegalStartException |

2] Difference b/w sleep() and wait()

|  |  |
| --- | --- |
| Wait() | Sleep() |
| * Wait releases the lock or monitor | * It does not |
| * Used in inter-thread communication | * Used to pause the execution |

REGEX(REGULAR EXPRESSION)

* Used only on strings and to serch,edit or manupilate text
* They are present in java.util.regex
* In regex “\_”is considered as a charecter
* It contains 3 classes they are
  + pattrenSyntaxException:
    - is a unchecked exception thrown whent the syntax of regular exception is wrong
  + pattern
    - pattern object is the complied version of the regular expression
    - pattern class does not have any public constructor
    - to create a pattern object we use public static method complie() and pass a regular expresition
  + matcher
    - it is a java regex engine object. This will try to match the input string with the pattern object created
    - it does not have any public constructor but to create a matcher object we 1st have to create pattern object and then create a matcher object input argument is string
    - matches() return type is Boolean

|  |  |
| --- | --- |
| \d | * Single digit of number |
| \D | * Matches any thing except digit |
| \d+ | * Any number of digits can be matches |
| \d{2},\d{2,4} | * Only 2 digits will be the pattern; least number is 2 and max will be 4 will be matched |
| \s,\s+ | * Single space, any number of space |
| \S,\S+ | * Any thing except space; multilple things except space |
| \w,\w+ | * Single character ;multiple charecters |
| \W | * Anything except numbers |

FILE-HANDLING:

* File🡪maven🡪last-option in that🡪click next🡪group id :give pakage name🡪Artifactname :for the progect🡪in package: in last column remove the package name