CORE -JAVA-THEORY

# CORE- JAVA

1. Interview Questions:
   1. Reverse a Given String:

* create a string to be reversed and assign it to the variable.
* use decremental for-loop.
* use str.length() & str.charAt() methods.
* logic.
  1. Count the number of words in the given String:
* Create a string (bug if there is space at the beginning then first trim and then split)
* Trim and split based on space and assign it to a string array.
* And use string array reference variable with length method to print number of words.
* Use for each loop statement to print all the words.
  1. To check number of opening and closing parenthesis and if same number print no error if not same print error.
* First use scanner class to take an input.
* Give 2 conditions to count ‘(‘ & ‘)’.
* Use for statement and use charAt method to compare and if it matches then increase the condition , continue as follows.
* Put if else condition for count1==count2 🡪 no error like that.
  1. Build A square of 5x5:
* Use nested for loop
* 1st loop has I <5; and it is used to go to next line.
* 2nd loop has i<5; and \* is printed.
  1. Build a right-angle triangle:
* It is same as square
* Here in 2nd loop just put if condition where all it should be blank that’s it.
  1. String to Uppercase and Lowercase:
* Built in method is there.
* First store sentence or word to a String variable.
* Then use ref variable.toLowerCase(); // for converting into lower case.
* Then use ref variable.toUpperCase(); // for converting into upper case.
  1. Remove the white space:
* Store sentence in string variable.
* Use referencevariable.trim(); // method used to trim 1st and last white space.
  1. Check string if it starts/ends with any letter:
* First store all diff letters in the string variable.
* Then use inbuilt method appended with reference variable.startsWith(“letter”);
* Then use inbuilt method appended with reference variable.endsWith(“letter”);
  1. To check length of the string:
* First store all diff letters in the string variable.
* Then use inbuilt method appended with reference variable.length();
  1. Convert any type of data type into string:
* Use String.valueOf(variable);
* Ex: char[] I = {a,b,c}; String str = String.valueOf(i);
  1. Write a program to find how many a’s and b’s are there in (aabaaaabaa):
* First store all diff letters in the string variable.
* Put 2 conditions to count a and b’s and assign 0 to them.
* Use for loop i<str.length() .
* Inn if loop Use str.charAt(i)==’a’ increase a condition a++;
* Else if loop Use str.charAt(i)==’b’ increase a condition b++;
* Print 2 conditions a & b .
  1. Right angle triangle where as follows 1-23-456-78910-1112131415:
* Put one condition variable and assign 1 to it.
* Use nested for loop.
* Outer for loop i=1; i<7 🡪 print to next line using soprintln().
* Inner for loop j =1; j<i🡪print k and k++.
  1. Find Duplicate element in array and remove it:

1. MUTABLE AND IMMUTABLE:
   1. Mutable: is something where, class object keeps on changing.
   2. Immutable: here once object is created then its state cannot be altered.
2. Steps to create an immutable Class:

* Create a final class.
* Set the values of the variables/properties using only PUBLIC constructors.
* Make the properties as private and final.
* Do-not provide any setters.
* Example:
  + **final** **public** **class** Ex2 {

**private** **final** **int** age;

**private** **final** String name;

**public** Ex2(**int** age, String name) { // Initializing properties values.

**this**.age = age;

**this**.name = name;

}

**public** **static** **void** main(String[] args) {

Ex2 ex = **new** Ex2(20, "pankaj");

// object creation to assign values by calling constructor.

}

**public** String getName() { // getters

**return** name;

}

**public** **int** getAge() { // getters

**return** age;

}

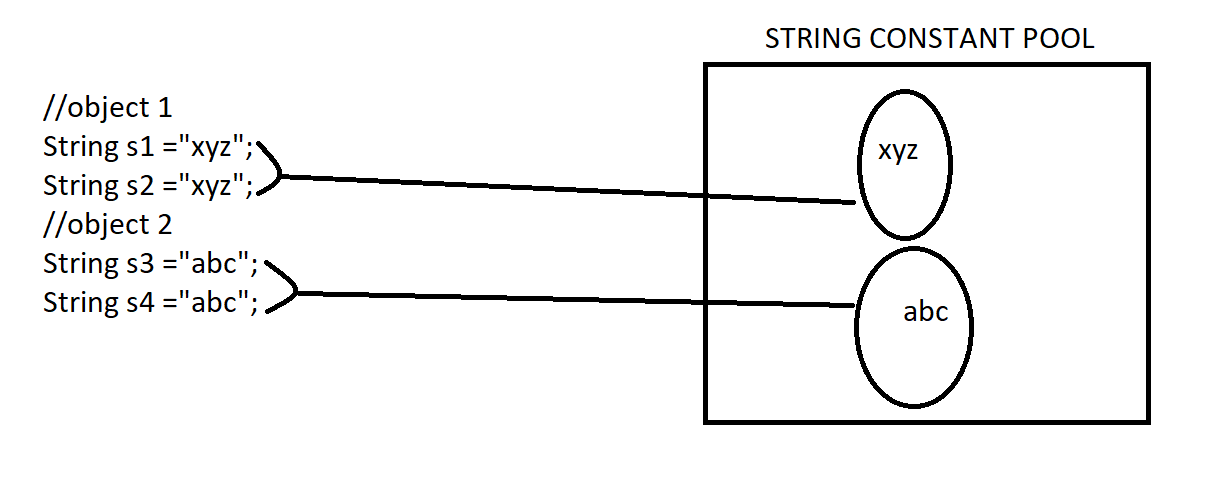
}

1. String is immutable:

* Even if we reassign value to String’s reference variable it won’t change its values there itself, instead it will create new object and store in it. whereas past object will go for garbage collection so String class is IMMUTABLE.
* always use s1.equals(s2) to compare values of 2 Strings.
* Example 1(IMP):
  + String s1 = "pankaj"; //object 1 and stores pankaj in it.
  + String s2 = "pankaj"; // as values are same it will point to object 1 only
  + String s3 = "Pankaj"; // as java is case sensitive this will create new object2 and stores Pankaj.
* Example 2(IMP):
  + String s1 = new String("pankaj”) ;// creates a new object 1 and stores pankaj.
  + String s2 = "pankaj"; // even though s1 and s2 values are same still as s1 is manually created in new object so here it will create another object 2 and stores pankaj in it.
  + String s3 = "pankaj"; // as values are same it will point to object 2 as here no new object created manually.
  + String s4 = new String("pankaj"); // creates a new object 3 and stores pankaj.
    - s1 == s4 -> false
    - s2 == s1 -> false
    - s2 == s3 -> true
    - s2 == s4 -> false
    - s3 == s4 -> false
    - s3 == s1 -> false

1. String Constant pool:

* Where object of string is created and stored, it is called as string constant pool:



1. Threads:

* Multitasking done at program level is called as threads.
* the main purpose of thread is to improve the performance of application by reducing execution time.
* Example: file searching logic between 3 folders to search a file is the best example.

1. Threads:
   1. Thread:

* Thread is a class
* consists of 2 methods
  + - a)->run(); --> we override this method and write what we want to multi-task with main method.
    - b)->start(); --> we use this to start the threading operation when it is appended with object reference variable (a1.start()).
* Class A extends Thread.
* Note:
* 1)make sure in main method first object should be created should call the start method and next we have to write different logic.
* 2)here in output we can see clearly that main method runs for some time and run method runs for some time and it repeats and order it executes is unpredictable.
* 3)main method is also considered as thread so here total of 2 threads are there.
* 4) first object created then use reference variable.start()→ this is the order of execution.
  1. Runnable:
* Runnable it’s an interface.
* consist of only run() method.
* Class A Implements Runnable.
* Note:
* 1) here we don't have start() method by default so we create object and also create thread class where we put reference variable of created object to get start() method.
* 2)here in output we can see clearly that main method runs for some time and run method runs for some time and it repeats and order it executes is unpredictable.
* 3)main method is also considered as thread.

1. Draw Backs of Threads:

* 1)Bank where money is deposited from one end and withdrawn simultaneously on other end without even complete full deposit where output will be corrupted so here, we won’t use thread concept.
* 2) 2 people booking only one remaining ticket at the exact same time even here data will be corrupted, so we don't use thread concept.
* 3) redeeming of same coupon.

1. Thread Synchronization:

* When 2 threads are operating on common data the data might get corrupted because of multitasking, to make the thread operates one after another we use synchronized keyword, the thread which is acquired the lock can only execute the block whereas the other thread would be in wait states only when the first thread release the lock the other thread will get opportunity and execute the lock.

1. Thread Pool(program):
2. Thread Priority:

* 1) it decides which thread runs first and which thread runs later(.setPriority(1-10)).
* 2) if we set the priority then it is the request made to thread scheduler there is no assuring that it will be processed of approved.
* 3) the minimum thread priority is 1 and maximum thread priority is 10, normal priority is 5, however we can set thread priority with number between 1-10.

1. Setting & getting Name of Thread:

* Use inbuilt method in same order 🡪.setName(“name”) , .getName(); .start().

1. Thread Pool (Theory):

* 1)when you need to limit the number of threads running your application at same time, this will help us to improve the performance of the application, i.e., instead of starting new thread for every task executed, current task can be passed to a thread pool.
* 2)Thread pool contains collection of thread, as soon as pool has any ideal thread task is assigned to one then and gets executed.
* 3) Thread pool are often used in servers, each connection arriving from network wrapped as task and passed to thread pool.
* 4)The thread in the thread pool will process the request on the connections done currently, this is how we can use existing threads instead of creating a new thread and there by improve the performance in-turns of execution.

1. Enum:

* Enum is a collection of constants.
* Syntax: public Enum A () {

Write all constants; // make sure you won’t initialize any variables.

}

* No inheritance is needed.
* here we can only access constants created Enum class other than that we can't access anything

1. Wrapper Class:

* 1) Here the values are stored in the object.
* 2) The process of storing the value inside an object is called as wrapping or boxing.
* 3)Reading the value from the object is called as unboxing.
* 4) Inbuilt methods:-
  + Integer.***MAX\_VALUE🡪*** // gives max value of integer.
  + i.longValue()🡪// converts integer into long
  + i.hashCode()🡪 // converts integer into hash code
  + Integer.***SIZE 🡪*** // gives size of the Integer i.e,

4 bytes🡪32bits(1byte=4bits)

* + i.toString() 🡪 // converts integer into string
  + i.doubleValue()🡪 // converts integer into double
  + i.byteValue()🡪 // converts integer into byte value.

1. Types of Wrapper Class:

* Byte b = 34;
* Short s = 20;
* Integer i = 30;
* Long l = 40l;
* Float f = 50.8f;
* Double d = 70.89;
* Character c = 'a';
* Boolean o = **true**;

1. Finalize:

* 1) is a method present inside object class.
* 2) garbage collection logic is implemented in Finalize method.
* 3) program.

1. Throws Keyword:

* It is applied on method.
* If any exception occurs in a method, then exception will be passed on to calling statement of the method.
* EX: psvm() throws Exception {

}

* After throws keyword we can give multiple class names.
* EX: psvm() throws IOException, SQLException {

}

1. Throw Keyword:

* It basically helps us to create customized exception as per the requirements of the developer.
* Basically, create a class with Customized Exception name and extends Throwable to same class.
* And in any class of same package we just have to write try and catch block where we want to use it and inside try we have to write throw new customizedException(); and in catch (customizedException e), here customizedException can be any name which we want to use of our own.

1. Regular Expression:

* Steps to follow for every regular Expression:
  + Pattern p = Pattren.compile(“[write anything which is to be searched]”);
  + Matcher m = p.matcher (“input what we have to refer to search”);
  + While (m.find()) {

Sysout(m.start()+”…….”+m.group());

}

* + Here m.find()→ it will search the given in matcher if its true it will enter loop.
  + Here m.start()→ will print the index number of matching letter
  + Here m.group()→ will print the letter of given index.
* (“[write anything which is to be searched]”) 🡪Whatever we write in square brackets will be searched.
* (“[a-z]”); 🡪To search all the words from a-z lower cases
* ("[0-9]"); 🡪here it is searching all the numbers from 0-9.
* ("[A-Z]"); 🡪 here it is searching all the alphabets from A-Z all upper-cases.
* ("[a-zA-Z0-9]"); 🡪 here it is searching all the alphabets from A-Z all upper&lower-case and also all the digits from 0-9.
* ("[^a-z]"); 🡪here it will search everything other than lower-case alphabets.
* ("[^a-zA-Z0-9]"); 🡪here it is searching anything other than all the alphabets from A-Z all upper & lower-case and also all the digits from 0-9.
* ("[0-9]{2}"); 🡪Anything we write in {} will form the number of groups of given string →if 2 then 5 groups, if 5 then 2 groups if 11 then 0 groups.
* ("\\s");🡪 give me white space from given string.
* ("\\S"); 🡪 upper-case S gives anything other than white spaces.
* ("\\d");🡪lower-case d gives every numbers/digits.
* ("\\D"); 🡪upper-case D gives everything other than numbers/digits.
* ("\\w"); 🡪gives everything other than special characters and not even white spaces other than that everything.
* ("\\W");🡪gives special characters and even white spaces other than that everything will be not printed.
* Programs:
  + Write a program to validate name input without any whitespaces.
  + Write a program to validate a given field where input should not contain any special characters using regular expression.
  + Validating Name using Regular Expression.
  + easy way to validate mobile number using matches method.
* Wild Cards in Regular Expression:
  + char\*-> it gives us zero occurrences or group of occurrences of particular character.
  + char+-> group of occurrences of a and not any zero occurrence of a.
  + char? -> only single and zero occurrences of 'a' and not any group occurrence of 'a'.

1. String Tokenizer:

* Major importance is to split the string.
* Program:
  + Write a program to count and split the sentence into words based on white spaces of any type using String Tokenizer, instead of using split() use this so that no bugs are present.
  + Split given date/string based on delimiter using String Tokenizer.
  + Using Regular Expression write a program to split the given string based on spaces.

1. Cloning:

* Cloning: The process of creating a replica of a particular object by copying the content of one object completely into another object.
* Study one example.

1. Hash code in Java:

* HASH CODE IN JAVA: will return integer representation of memory address as shown below.
* Ex8 ex8 = **new** Ex8();
* System.***out***.println(ex8.hashCode()); // converts hexadecimal values into integer representation.

1. Annotation’s:

* Annotations: communication to java complier to do something/command the compiler.
* 3 Types:
  + @Override 🡪 refer core java classes.
  + @SupressWarning🡪**@SupressWarning**: me instructing compiler to tell to suppress the warning.
    - If we write it outside method then everything inside method will supress warnings.
    - If we write inside method then on specified line of code warning will be supressed.
  + @Depricated →**@deprecated**: it tells which features are out-dated.

1. JDBC:
   1. Refer Core Java Classes.
2. Collection:

* Collection stores group of objects in it.
* In java collection is a framework which has readily available logic to deal with different Data Structures.

1. Types of Collection:
2. LIST:

* ArrayList:
  + <\*> internally it is implemented as dynamic array.
  + <\*> initial size of ArrayList is 10.
  + <\*> when we exceed the initial size automatically arrayList size increases by 1.5 times.
  + <\*> ArrayList maintains insertion order.
  + <\*> it can consists of duplicate elements.
  + <\*> the advantage of array list is reading of data would give us best performance.
  + <\*> disadvantage, insertion of data in b/w of the array list will result in worst performance,
  + as the shuffle and data adjustments has to be done so performance error.
  + <\*> we can store heterogeneous data but it is not a good practice, so we use generic here.
* Method inbuilt:

1. X.add(10); // add() method is present in arrayList to add values it is also called as boxing as here objects are created in each.
2. x.add(1,500);//x.add(index,value); // adding objects in-between.
3. x.addAll(2, y);//x.addAll(index,collection/array);//Adding 2 arrays.
4. if(x.contains(/\*object\*/500)) {}// to search weather it contains or not.
5. x.remove(1); //x.remove(index of object); //removing value based on index
6. int size = x.size(); // size of given arrayList.
7. System.out.println(x.get(2));// x.get(index number of values to be fetched);
8. read data using iterator:

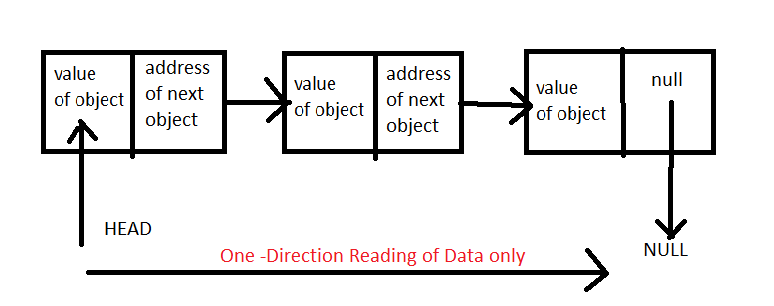
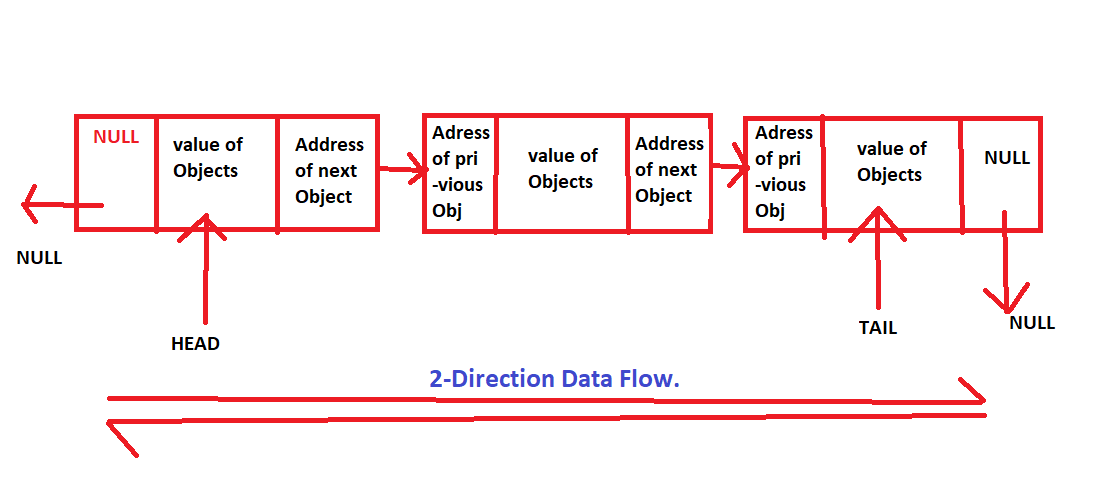
Iterator<Integer> itr = x.iterator();

while(itr.hasNext()) {

System.out.println("value of x :"+ itr.next());

}

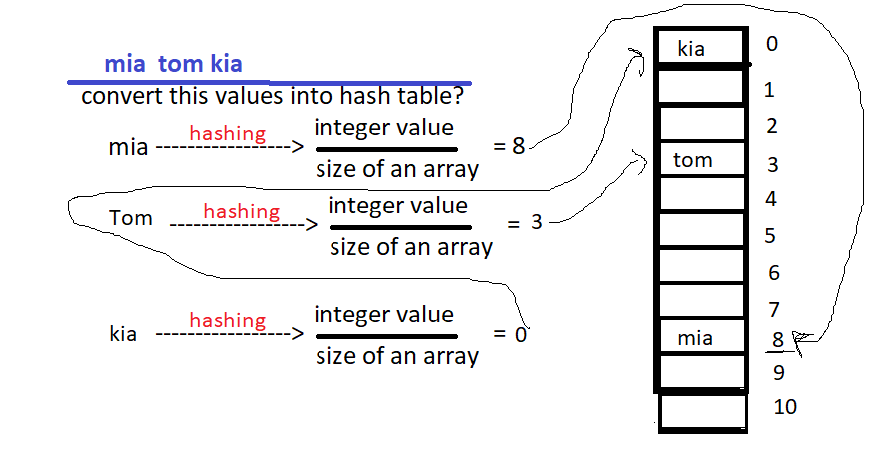
* LinkedList:

1. Singly Linked List:
   * 
   * Disadvantage is takes more memory.
   * We can understand that we reached last data by reading null value at the end.
   * This is in one direction data flow i.e., head to tail, so it is Singly linked list.
   * At beginning adding of data performance is good, but at the end or in between data adding there will be performance problem.
   * In JDK LinkedList is internally implemented as doubly linked list.
2. Doubly-Linked List:
   * 
   * Disadvantages is more memory is taken as more data each node has i.e., 3-Fields.
   * Adding data to end and beginning performance is very good but adding data in between performance will be worst.
   * Inbuilt Methods:
3. All methods are same as that of list but some extra methods are there explicitly.
4. LinkedList<Integer> x = **new** LinkedList<Integer>();
5. addFirst()🡪every times a new element is added to first it becomes first element i.e., if 20 and 30 are added respectively then 30,20 in this way it is stored.
6. X.add()🡪 );// always will be the last object as addFirst is in combination with it.
7. addLast()🡪// every times a new element is added to last it becomes last element i.e., if 20 and 30 are added respectively then 20,30 in this way it is stored.
8. x.add(5);// always will be the first object as addLast is in combination with it.
9. Study one example(student name and id).
10. Note: Linked list and Array list can consist of duplicate elements and maintains insertion order.

* Vector:
  + Vector is not used much as it gives bad performance as all methods present in are synchronized and due to this there is no thread operation so bad performance.

1. Hash Table:

* Stores value in key, value pair (Ex🡪 Pankaj, 6361317502).
* It is an legacy class.
* Variable.put(key,value);
* Variable.get(key); // get values based on key provided.
* Variable.values();// get all values and stores in array.
* Variable.keyset(); // get all keys and stores in array.
* Here pankaj is key and 6361317502 is value for the reference to key.
* Hashing→ is a technique where when representing any entity in the form of integer and it is done in java using hash-code/hashing method.
* Hashcode→ is a method present in object class in java.
* Hash tables → is an associated array where in values are stored as key, value pair.
* Collisions→ when 2 values are being stored at the same index number it is called as collision, to solve this problem in hash table we store the data as list, mapped to the same index number.
* Ex:



* Initial size of hash table is 16, when load ratio becomes 75% i.e., out of 16, 12 elements are injected into the table then size of the table automatically doubles.
* Hash table is synchronized.

1. Set:

* SET:
* It is interface.
* Do not maintain insertion order.
* Cannot contain any duplicate values.
  + 1)Hash set:
    - It uses hash table internally.
    - It uses hashing concept to inject data into database.
    - It will contain only unique element.
    - Doesnot maintain insertion order.
    - This is not synchronized.
    - This will permit null elements to store in it.
    - LinkedHashSet:
      * It maintains insertion order
      * Can contains only unique elements, no duplicate values.
  + 2)Tree Set:
    - Contains unique elemts only.
    - It sorts the data in ascending order automatically.

1. MAP – Interface:

* It does not come under collection concept.
* Hash Map:
  + Internally uses hash table.
  + To inject data into hash table it uses hashing technique.
  + A hashmap stores data as key value pair.
  + Hash map is not synchronized.
  + Example : shopping mall barcode→product.
  + Variable.put(key,value);
  + Variable.get(key); // get values based on key provided.
  + Variable.values();// get all values and stores in array.
  + Variable.keyset(); // get all keys and stores in array.

1. Difference between HashTable v/s HashSet v/s HashMap:

|  |  |  |
| --- | --- | --- |
| * Hash Table | * Hash Set | * Hash Map |
| * It is a legacy class. | * It implements hash table. | * It implements hash table. |
| * It is synchronised. | * It is not synchronised. | * It is not synchronised. |
| * It stores values as key, value pairs. | * It stores values as objects. | * It stores values as key, value pairs. |
| * Maintains Insertion order. | * Does not Maintain an Insertion order. | * Maintains Insertion order. |
| * Uses hashing technique to store data | * -------------- | * ------------------ |
| * Cannot have duplicate elements | * -------------- | * ------------------ |

1. Comparator:

* it is used to order objects of user defined classes, if in sorting if obj1 comes first and then obj2 then it will return -ve values. but if we sorting obj2 comes first and then obj1, then it will return +ve value, if both objects are same then it will return zero.
* Give one simple example (compare 2 strings).

1. Generics:

* Gives us the flexibility to dynamically allocate the data type in program.
* Example.

1. J-SHELL:

* JDK9 a new feature J-shell was added that helps us to quickly build java code in it, to access J-shell go to java jdk, bin , then double click on jshell.exe file.
* Example-Program of writing uni-code directly. jdk 9 and above.