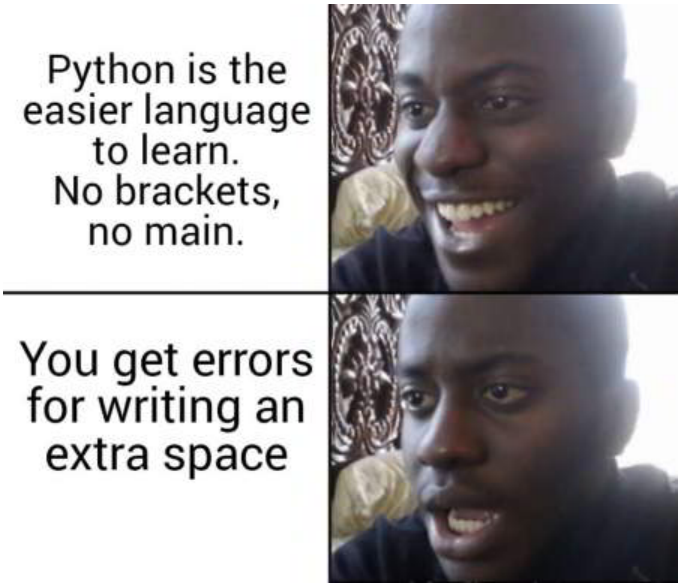
Sirf khwabh Nhi

25/08



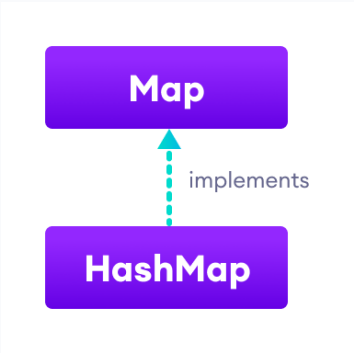
1] HashMap

HashMap class and its various operations with the help of examples.

The `HashMap` class of the Java collections framework provides the functionality of the hash table data structure.

It stores elements in key/value pairs. Here, keys are unique identifiers used to associate each value on a map.

The `HashMap` class implements the Map interface.



Java HashMap Implementation

Create a HashMap

In order to create a hash map, we must import the `java.util.HashMap` package first. Once we import the package, here is how we can create hashmaps in Java.

```
// hashMap creation with 8 capacity and 0.6 load factor

HashMap<K, V> numbers = new HashMap<>();
```

In the above code, we have created a hashmap named `numbers`. Here, K represents the key type and V represents the type of values. For example,

```
HashMap<String, Integer> numbers = new HashMap<>();
```

Here, the type of keys is `String` and the type of values is `Integer`.

Example 1: Create HashMap in Java

```
import java.util.HashMap;


class Main {

  public static void main(String[] args) {


    // create a hashmap

    HashMap<String, Integer> languages = new HashMap<>();


    // add elements to hashmap

    languages.put("Java", 8);

    languages.put("JavaScript", 1);

    languages.put("Python", 3);

    System.out.println("HashMap: " + languages);

  }
```

```
}
```

Output

```
HashMap: {Java=8, JavaScript=1, Python=3}
```

In the above example, we have created a `HashMap` named `languages`.

Here, we have used the `put()` method to add elements to the hashmap. We will learn more about the `put()` method later in this tutorial.

Basic Operations on Java HashMap

The `HashMap` class provides various methods to perform different operations on hashmaps. We will look at some commonly used arraylist operations in this tutorial:

- Add elements
- Access elements
- Change elements
- Remove elements

---

1. Add elements to a HashMap

To add a single element to the hashmap, we use the `put()` method of the `HashMap` class. For example,

```
import java.util.HashMap;



class Main {

  public static void main(String[] args) {



    // create a hashmap
```

```
    HashMap<String, Integer> numbers = new HashMap<>();



    System.out.println("Initial HashMap: " + numbers);

    // put() method to add elements

    numbers.put("One", 1);

    numbers.put("Two", 2);

    numbers.put("Three", 3);

    System.out.println("HashMap after put(): " + numbers);

  }

}
```

Output

```
Initial HashMap: {}

HashMap after put(): {One=1, Two=2, Three=3}
```

In the above example, we have created a `HashMap` named `numbers`. Here, we have used the `put()` method to add elements to numbers.

Notice the statement,

```
numbers.put("One", 1);
```

Here, we are passing the `String` value `One` as the key and `Integer` value `1` as the value to the `put()` method.

- Java HashMap put()
- Java HashMap putAll()
- Java HashMap putIfAbsent()

---

2. Access HashMap Elements

We can use the `get()` method to access the value from the hashmap. For example,

```
import java.util.HashMap;


class Main {
  public static void main(String[] args) {


    HashMap<Integer, String> languages = new HashMap<>();
    languages.put(1, "Java");
    languages.put(2, "Python");
    languages.put(3, "JavaScript");
    System.out.println("HashMap: " + languages);


    // get() method to get value
    String value = languages.get(1);
    System.out.println("Value at index 1: " + value);
  }
}
```

Output

```
HashMap: {1=Java, 2=Python, 3=JavaScript}
Value at index 1: Java
```

In the above example, notice the expression,

```
languages.get(1);
```

Here, the `get()` method takes the key as its argument and returns the corresponding value associated with the key.

We can also access the keys, values, and key/value pairs of the hashmap as set views using `keySet()`, `values()`, and `entrySet()` methods respectively. For example,

```
import java.util.HashMap;


class Main {
  public static void main(String[] args) {
    HashMap<Integer, String> languages = new HashMap<>();


    languages.put(1, "Java");
    languages.put(2, "Python");
    languages.put(3, "JavaScript");
    System.out.println("HashMap: " + languages);


    // return set view of keys
    // using keySet()
    System.out.println("Keys: " + languages.keySet());


    // return set view of values
    // using values()
    System.out.println("Values: " + languages.values());
```

```
    // return set view of key/value pairs
    // using entrySet()
    System.out.println("Key/Value mappings: " + languages.entrySet());
  }
}
```

Output

```
HashMap: {1=Java, 2=Python, 3=JavaScript}
Keys: [1, 2, 3]
Values: [Java, Python, JavaScript]
Key/Value mappings: [1=Java, 2=Python, 3=JavaScript]
```

In the above example, we have created a hashmap named `languages`. Here, we are accessing the keys, values, and key/value mappings from the hashmap.

- Java HashMap get()
- Java Hashmap getOrDefault()
- Java HashMap keySet()
- Java HashMap values()
- Java HashMap entrySet()

---

3. Change HashMap Value

We can use the `replace()` method to change the value associated with a key in a hashmap. For example,

```
import java.util.HashMap;

class Main {
  public static void main(String[] args) {
```

```
    HashMap<Integer, String> languages = new HashMap<>();
    languages.put(1, "Java");
    languages.put(2, "Python");
    languages.put(3, "JavaScript");
    System.out.println("Original HashMap: " + languages);

    // change element with key 2
    languages.replace(2, "C++");
    System.out.println("HashMap using replace(): " + languages);
  }
}
```

Output

```
Original HashMap: {1=Java, 2=Python, 3=JavaScript}
HashMap using replace(): {1=Java, 2=C++, 3=JavaScript}
```

In the above example, we have created a hashmap named `languages`. Notice the expression,

```
languages.replace(2, "C++");
```

Here, we are changing the value referred to by key 2 with the new value `C++`.

The `HashMap` class also provides some variations of the `replace()` method.

- Java HashMap replace()
- Java HashMap replaceAll()

---

4. Remove HashMap Elements

To remove elements from a hashmap, we can use the remove() method. For example,

```
import java.util.HashMap;



class Main {

  public static void main(String[] args) {



    HashMap<Integer, String> languages = new HashMap<>();

    languages.put(1, "Java");

    languages.put(2, "Python");

    languages.put(3, "JavaScript");

    System.out.println("HashMap: " + languages);



    // remove element associated with key 2

    String value = languages.remove(2);

    System.out.println("Removed value: " + value);



    System.out.println("Updated HashMap: " + languages);

  }

}
```

Output

```
HashMap: {1=Java, 2=Python, 3=JavaScript}

Removed value: Python
```

```
Updated HashMap: {1=Java, 3=JavaScript}
```

Here, the `remove()` method takes the key as its parameter. It then returns the value associated with the key and removes the entry.

We can also remove the entry only under certain conditions. For example,

```
remove(2, "C++");
```

Here, the `remove()` method only removes the entry if the key 2 is associated with the value C++. Since 2 is not associated with C++, it doesn't remove the entry.

Java HashMap remove().

Other Methods of HashMap

Method	Description
clear()	removes all mappings from the `HashMap`
compute()	computes a new value for the specified key
computeIfAbsent()	computes value if a mapping for the key is not present
computeIfPresent()	computes a value for mapping if the key is present
merge()	merges the specified mapping to the `HashMap`

clone()	makes the copy of the `HashMap`
containsKey()	checks if the specified key is present in Hashmap
containsValue()	checks if `Hashmap` contains the specified value
size()	returns the number of items in `HashMap`
isEmpty()	checks if the `Hashmap` is empty

Iterate through a HashMap

To iterate through each entry of the hashmap, we can use Java for each loop. We can iterate through keys only, vales only, and key/value mapping. For example,

```
import java.util.HashMap;

import java.util.Map.Entry;



class Main {

  public static void main(String[] args) {



    // create a HashMap

    HashMap<Integer, String> languages = new HashMap<>();

    languages.put(1, "Java");

    languages.put(2, "Python");
```

```
    languages.put(3, "JavaScript");
    System.out.println("HashMap: " + languages);

    // iterate through keys only
    System.out.print("Keys: ");
    for (Integer key : languages.keySet()) {
      System.out.print(key);
      System.out.print(", ");
    }

    // iterate through values only
    System.out.print("\nValues: ");
    for (String value : languages.values()) {
      System.out.print(value);
      System.out.print(", ");
    }

    // iterate through key/value entries
    System.out.print("\nEntries: ");
    for (Entry<Integer, String> entry : languages.entrySet()) {
      System.out.print(entry);
      System.out.print(", ");
    }
```

```
    }
}
```

Output

```
HashMap: {1=Java, 2=Python, 3=JavaScript}
Keys: 1, 2, 3,
Values: Java, Python, JavaScript,
Entries: 1=Java, 2=Python, 3=JavaScript,
```

Note that we have used the `Map.Entry` in the above example. It is the nested class of the `Map` interface that returns a view (elements) of the map.

We first need to import the `java.util.Map.Entry` package in order to use this class.

This nested class returns a view (elements) of the map.

Creating HashMap from Other Maps

In Java, we can also create a hashmap from other maps. For example,

```
import java.util.HashMap;
import java.util.TreeMap;


class Main {
  public static void main(String[] args) {


    // create a treemap
    TreeMap<String, Integer> evenNumbers = new TreeMap<>();
    evenNumbers.put("Two", 2);
    evenNumbers.put("Four", 4);
```

```
    System.out.println("TreeMap: " + evenNumbers);



    // create hashmap from the treemap

    HashMap<String, Integer> numbers = new HashMap<>(evenNumbers);

    numbers.put("Three", 3);

    System.out.println("HashMap: " + numbers);

  }

}
```

Output

```
TreeMap: {Four=4, Two=2}

HashMap: {Two=2, Three=3, Four=4}
```

In the above example, we have created a `TreeMap` named `evenNumbers`. Notice the expression,

```
numbers = new HashMap<>(evenNumbers)
```

Here, we are creating a `HashMap` named numbers using the `TreeMap`. To learn more about treemap, visit Java TreeMap.

Note: While creating a hashmap, we can include optional parameters: capacity and load factor. For example,

```
HashMap<K, V> numbers = new HashMap<>(8, 0.6f);
```

Here,

- 8 (capacity is 8) - This means it can store 8 entries.
- 0.6f (load factor is 0.6) - This means whenever our hash table is filled by 60%, the entries are moved to a new hash table double the size of the original hash table.

If the optional parameters not used, then the default capacity will be 16 and the default load factor will be 0.75.

IMP TECHKNOWLEDGE

1) Merge Sort

Merge sort is one of the most flexible sorting algorithms in java known to mankind (yes, no kidding). It uses the divide and conquers strategy for sorting elements in an array. It is also a stable sort, meaning that it will not change the order of the original elements in an array concerning each other. The underlying strategy breaks up the array into multiple smaller segments till segments of only two elements (or one element) are obtained. Now, elements in these segments are sorted and the segments are merged to form larger segments. This process continues till the entire array is sorted.

This algorithm has two main parts:

- mergeSort() – This function calculates the middle index for the subarray and then partitions the subarray into two halves. The first half runs from index left to middle, while the second half runs from index middle+1 to right. After the partitioning is done, this function automatically calls the merge() function for sorting the subarray being handled by the mergeSort() call.
- merge() – This function does the actual heavy lifting for the sorting process. It requires the input of four parameters – the array, the starting index (left), the middle index (middle), and the ending index (right). Once received, merge() will split the subarray into two subarrays – one left subarray and one right subarray. The left subarray runs from index left to middle,

while the right subarray runs from index middle+1 to right. This function then merges the two subarrays to get the sorted subarray.

Merge Sort Java Code:

```
class Sort

{

    void merge(int arr[], int left, int middle, int right)

    {

        int low = middle - left + 1;                    //size of the left
subarray

        int high = right - middle;                      //size of the right
subarray



        int L[] = new int[low];                             //create the
left and right subarray

        int R[] = new int[high];



        int i = 0, j = 0;



        for (i = 0; i < low; i++)                                   //copy
elements into left subarray

        {

            L[i] = arr[left + i];

        }
```

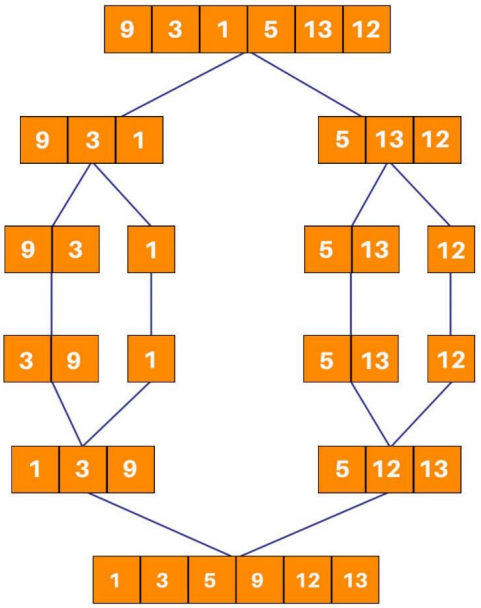
```
        for (j = 0; j < high; j++)                          //copy
elements into right subarray
        {
            R[j] = arr[middle + 1 + j];
        }



        int k = left;                                       //get
starting index for sort
        i = 0;                                          //reset loop
variables before performing merge
        j = 0;


        while (i < low && j < high)                     //merge the left and
right subarrays
        {
            if (L[i] <= R[j])
            {
                arr[k] = L[i];
                i++;
            }
            else
            {
                arr[k] = R[j];
```

```
                j++;

            }

            k++;

        }



        while (i < low)                                 //merge the remaining
elements from the left subarray

        {

            arr[k] = L[i];

            i++;

            k++;

        }



        while (j < high)                                //merge the remaining
elements from right subarray

        {

            arr[k] = R[j];

            j++;

            k++;

        }

    }
```

```
    void mergeSort(int arr[], int left, int right)       //helper function
that creates the sub cases for sorting
    {
        int middle;
        if (left < right) {                              //sort only if the
left index is lesser than the right index (meaning that sorting is done)
            middle = (left + right) / 2;


            mergeSort(arr, left, middle);                    //left subarray
            mergeSort(arr, middle + 1, right);               //right subarray


            merge(arr, left, middle, right);                //merge the two
subarrays
        }
    }


    void display(int arr[])                  //display the array
    {
        for (int i=0; i<arr.length; ++i)
        {
            System.out.print(arr[i]+" ");
        }
    }
```

```
    public static void main(String args[])

    {

        int arr[] = { 9, 3, 1, 5, 13, 12 };

        Sort ob = new Sort();

        ob.mergeSort(arr, 0, arr.length - 1);

        ob.display(arr);

    }

}
```

Explanation of how it works:



2) Heap Sort

Heap sort is one of the most important sorting methods in java that one needs to learn to get into sorting. It combines the concepts of a tree as well as sorting, properly reinforcing the use of concepts from both. A heap is a complete binary search tree where items are stored in a special order

depending on the requirement. A min-heap contains the minimum element at the root, and every child of the root must be greater than the root itself. The children at the level after that must be greater than these children, and so on. Similarly, a max-heap contains the maximum element at the root. For the sorting process, the heap is stored as an array where for every parent node at the index i, the left child is at index 2 * i + 1, and the right child is at index 2 * i + 2.

A max heap is built with the elements of the unsorted array, and then the maximum element is extracted from the root of the array and then exchanged with the last element of the array. Once done, the max heap is rebuilt for getting the next maximum element. This process continues till there is only one node present in the heap.

This algorithm has two main parts:-

- heapSort() – This function helps construct the max heap initially for use. Once done, every root element is extracted and sent to the end of the array. Once done, the max heap is reconstructed from the root. The root is again extracted and sent to the end of the array, and hence the process continues.
- heapify() – This function is the building block of the heap sort algorithm. This function determines the maximum from the element being examined as the root and its two children. If the maximum is among the children of the root, the root and its child are swapped. This process is then repeated for the new root. When the maximum element in the array is found (such that its children are smaller than it) the function stops. For the node at index i, the left child is at index 2 * i + 1, and the right child is at index 2 * i + 1. (indexing in an array starts from 0, so the root is at 0).

Heap Sort Java Code:

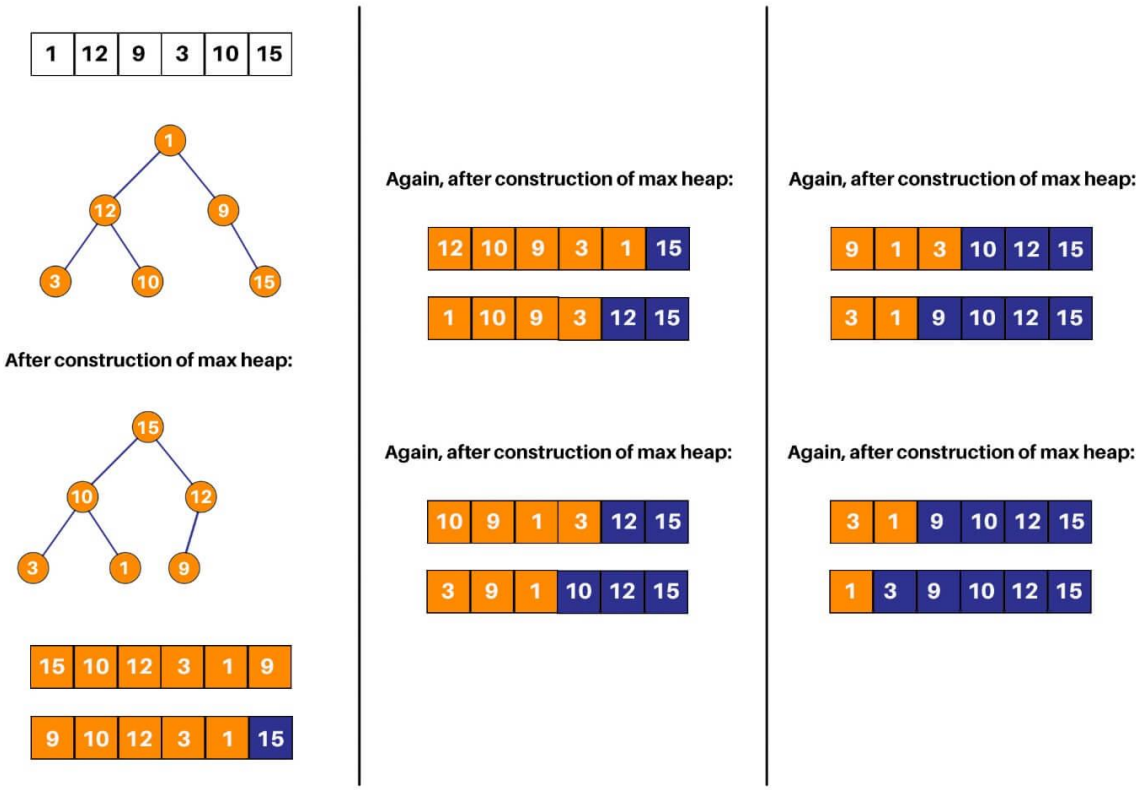
```
class Sort {

    public void heapSort(int arr[])

    {

        int temp;



        for (int i = arr.length / 2 - 1; i >= 0; i--)                    //build
the heap
```

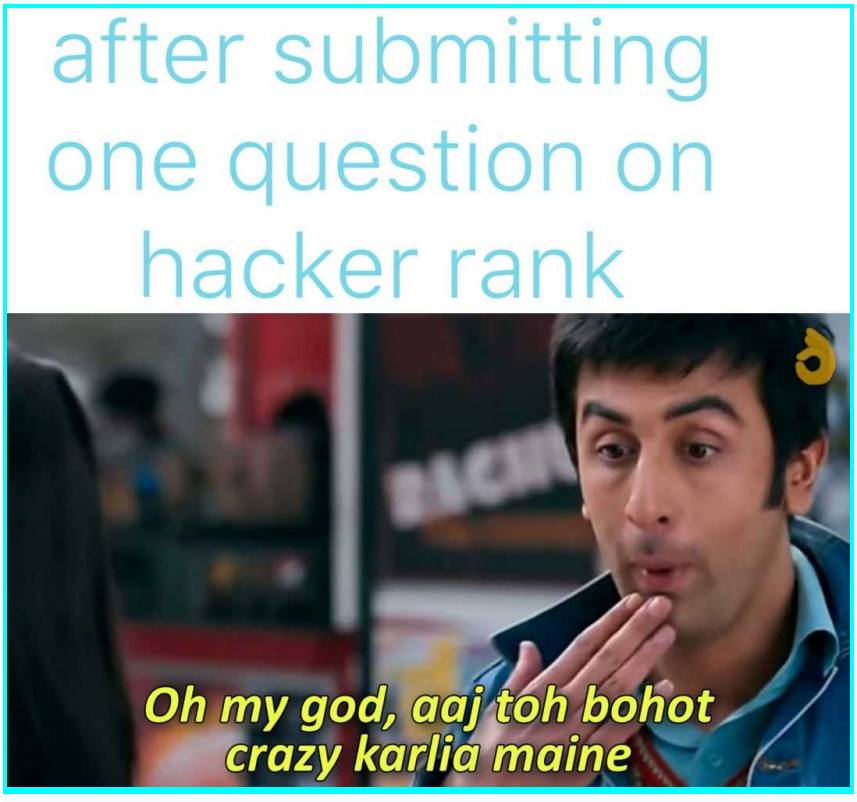
```
        {
            heapify(arr, arr.length, i);
        }

        for (int i = arr.length - 1; i > 0; i--)
//extract elements from the heap
        {
            temp = arr[0];
//move current root to end (since it is the largest)
            arr[0] = arr[i];
            arr[i] = temp;
            heapify(arr, i, 0);
//recall heapify to rebuild heap for the remaining elements
        }
    }

    void heapify(int arr[], int n, int i)
    {
        int MAX = i; // Initialize largest as root
        int left = 2 * i + 1; //index of the left child of ith node = 2*i + 1
        int right = 2 * i + 2; //index of the right child of ith node  = 2*i
+ 2
        int temp;
```

```
        if (left < n && arr[left] > arr[MAX])            //check if the left
child of the root is larger than the root
        {
            MAX = left;
        }

        if (right < n && arr[right] > arr[MAX])            //check if the
right child of the root is larger than the root
        {
            MAX = right;
        }

        if (MAX != i)
        {                                                  //repeat the
procedure for finding the largest element in the heap
            temp = arr[i];
            arr[i] = arr[MAX];
            arr[MAX] = temp;
            heapify(arr, n, MAX);
        }
    }

    void display(int arr[])                  //display the array
    {
```

```
        for (int i=0; i<arr.length; ++i)
        {
            System.out.print(arr[i]+" ");
        }
    }

    public static void main(String args[])
    {
        int arr[] = { 1, 12, 9 , 3, 10, 15 };

        Sort ob = new Sort();
        ob.heapSort(arr);
        ob.display(arr);
    }
}
```

Explanation of how it works:


1 12 9 3 10 15
After construction of max heap:
15 10 12 3 1 9
9 10 12 3 1 15
Again, after construction of max heap:
12 10 9 3 1 15
1 10 9 3 12 15
Again, after construction of max heap:
10 9 1 3 12 15
3 9 1 10 12 15
Again, after construction of max heap:
9 1 3 10 12 15
3 1 9 10 12 15
Again, after construction of max heap:
3 1 9 10 12 15
1 3 9 10 12 15


after submitting one question on hacker rank
Oh my god, aaj toh bohot crazy karlia maine

Statement and Assumption questions

Q 1.

Statement: The advisable age for a child to join a school is 5 years.

Assumption I: At this age, the child is familiar to adaptability

Assumption II: After this age, kids do not like to go to school

Assumption III: Schools do not take admission of children who are more than 5 years old

1. Only Assumption I follows
2. Both Assumptions I & III follow
3. Assumption I, II & III follow
4. Only Assumption II follows
5. None of the Above

Answer: (1) Only Assumption I follows

Explanation: In the given statement it is mentioned that 5 years is an advisable age for kids to join the school. The only sensible assumption which proves the statement to be true is that chances of kids getting more adaptable to the school at this young age are very high

Q 2.

Statement: Food poisoning due to the consumption of liquor is very common in rural areas

Assumption I: There are more illegal and unauthorised shops selling liquor in villages and rural areas

Assumption II: The ratio of people drinking liquor in villages is much more than that in towns

1. Both Assumption I and II follow
2. Neither Assumption I nor Assumption II follows
3. Only Assumption I follows
4. Assumption II follows but Assumption I does not follow
5. Either Assumption I or Assumption II follows

Answer: (3) Only Assumption I follows

Explanation: The statement is talking about food poisoning due to liquor so the number of people consuming liquor in towns or villages is not the main concern here. Which is why the only assumption I follow

Q 3.

Statement: Divya was advised by the Doctor that she should not take part in the dance competition

Assumption I: The Doctor did not want Divya to take part in the competition because he was afraid that she might lose

Assumption II: Divya had major surgery because of her injury

Assumption III: Divya did not have the money to go for the auditions

1. All Assumption I, II & III follow
2. Only Assumption I follows
3. Assumption II follows but Assumption I and III do not follow
4. None of the three assumptions follow
5. Only Assumption III follows

Answer: (3) Assumption II follows but Assumption I and III do not follow

Explanation: Based on the statement given, the Doctor would only advice a patient to not do certain things if they are unwell, in this case surgery. Which is the only suitable assumption why the doctor would advise Divya to not take part in the dance competition

Q 4.

Statement: In an election conducted in Village X, only 20% of the total number of women in the village came to vote.

Assumption I: The number of men in the village is more than the number of women in the village X

Assumption II: Women had to cook food and could not come to vote

1. Only Assumption I follows
2. Only Assumption II follows
3. Neither Assumption I nor Assumption II follows
4. Either Assumption I or Assumption II follows
5. Both Assumption I and Assumption II follow

Answer: (3) Neither Assumption I nor Assumption II follows

Explanation: The statement clearly indicates that out of the total number of women in the village only 20% came around to vote so the ratio between the number of men and women is not applicable here and the second assumption is not applicable as well.

Q 5.

Statement 1: The school has decided to cancel the summer camp this year

Assumption I: No entries have been received by the Institution for students willing to join the summer camp

Assumption II: It is being cancelled because the weather is too hot

1. Only Assumption I follows
2. Only Assumption II follows
3. Neither Assumption I nor Assumption II follows
4. Either Assumption I or Assumption II follows
5. Both Assumption I and Assumption II follow

Answer: (1) Only Assumption I follows

Explanation: Because the school was not receiving any student entries for participation in the summer camp, that is why it was cancelled. The second option is not implicit because the camp is being conducted in summers, so the weather is expected to be hot.

Statement & Conclusion questions

Q 1.

Statement: In a T20 match played between India and Australia, the total runs made by the Indian team were 200. 160 runs out of 200 runs were made by spinners.

Conclusion I: 80% of the team consists of spinners

Conclusion II: The opening batsmen were spinners

1. Only Conclusion I is true
2. Only Conclusion II is true
3. Both Conclusion I and II are true
4. Neither Conclusion I nor II is true
5. Either Conclusion I or II is true

Answer: (4) Neither Conclusion I nor II is true; Neither conclusion is logically correct with regard to the statement

Q 2.

Statement: The Team Manager humiliated Varun in front of his colleagues on a Monday morning.

Conclusion I: The Team Manager did not like Varun

Conclusion II: Varun was not as popular as his other colleagues

1. Only Conclusion I is true
2. Only Conclusion II is true
3. Both Conclusion I and II are true
4. Neither Conclusion I nor II is true
5. Either Conclusion I or II is true

Answer: (4) Neither Conclusion I nor II is true; None of the conclusions gives a valid reason for the statement to be correct.

Q 3.

Statement: A metal producing Company in India has moved from a position where there was a shortage of metal production to a position where there is the self-sufficiency of metal.

Conclusion I: India was previously importing metal

Conclusion II: at this speed of self sufficiency, India can soon become a foreign exchange earner

1. Only Conclusion I is true
2. Only Conclusion II is true
3. Both Conclusion I and II are true
4. Neither Conclusion I nor II is true
5. Either Conclusion I or II is true

Answer: (3) Both Conclusion I and II is true

Q 4.

Statement: Only good dancers are invited to the competition. No one without dancing techniques is a good dancer.

Conclusion I: All invited dancers in the competition dance well

Conclusion II: Those dancers who do not have good dancing techniques are not invited to the competition

1. Only Conclusion I is true
2. Only Conclusion II is true
3. Both Conclusion I and II are true
4. Neither Conclusion I nor II is true
5. Either Conclusion I or II is true

Answer: (3) Both Conclusion I and II are true

KHALI FHUKAT APTI MIX

1. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. Chess

B. Badminton

C. Cricket
D. Volleyball
E. Table tennis

Answer – A. Chess

Explanation: Remaining games are played in the field

2. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. Win
B. Compete
C. Triumph
D. Victory
E. Succeed

Answer – B. Compete

Explanation: Remaining words give the same meaning

3. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. Jackal
B. Horse
C. Cat
D. Dog
E. Wolf

Answer – B. Horse

Explanation: Horse alone omnivores

4. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. YB
B. AD
C. IK
D. MP
E. WZ

Answer – C. IK

Explanation: Remaining pairs have 2 letters between them

5. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. BCDF
B. KJNL
C. RSTU
D. WXYZ
E. FGHP

Answer – C. RSTU

Explanation: Remaining are contained only consonants

6. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. Avoid
B. Obstruct
C. Block
D. Hamper
E. Impede

Answer – A. Avoid

Explanation: Remaining words give the same meaning

7. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. Arvind Kejriwal
B. Siddaramaiah
C. Draupadi Murmu
D. Mukul Sangma
E. Nabam Tuki

Answer – C. Draupadi Murmu

Explanation:

Draupadi Murmu is governor of Jharkhand remaining all are CM 's

8. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. Wheat
B. Corn
C. Millet
D. Jowar
E. Cotton

Answer – E. Cotton **Explanation:**

Cotton is a fiber crop, remaining are food crops

9. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. Sparrow
B. Parrot
C. Swan
D. Koel
E. Vulture

Answer – C. Swan **Explanation:**

Only Swan is the water bird

10. **Four of the following five are alike in a certain way and so form a group. Which is the one that does not belong to that group?**

A. Manure
B. Ammonia
C. Urea
D. Potash
E. Nitrogen

Answer – E. Nitrogen **Explanation:**

All except Nitrogen are used as fertilizers

11. **Statements:**

I. The bank decided to give advances to the priority sector at the rate of interest at par with the corporate sector.

II. The percentage of bad loans given by the banks to the priority sector is very low compared to the corporate sector.

A. I is the cause and II is its effect.
B. II is the cause and I is its effect.
C. Both I and II are independent causes.
D. Both I and II are effects of independent causes.
E. Both I and II are effects of some common causes.

Answer – D. Both the I and II are effects of independent causes.

Explanation:

The bank can change its policy with the priority sector due to a change in the government policy. Thus, it is an effect. But the same cause cannot be the reason for the percentage of bad loans being less for the priority sector.

12. **Statements:**

I. Many seats in the private medical colleges in the state have remained vacant this year. **II.** The government medical colleges in the state could accommodate all the students who sought admission this year.

A. I is the cause and II is its effect.
B. II is the cause and I is its effect.
C. Both I and II are effects of independent causes.
D. Both I and II are effects of some common causes.
E. Both I and II are independent causes.

Answer – B. II is the cause and I is its effect.

Explanation:

Statement II is the cause and statement I is its effect. As the government medical college could accommodate all students who applied to it (and most medical students prefer government colleges which are less expensive), many seats in the private medical colleges in the state have remained vacant.

13. **Statements:**

I. The Ganges river is flooded due to large sheets of ice coming down from the Himalayas. **II.** Many people in the Himalayan ranges lost their houses under huge sheets of ice.

A. I is the cause and II is its effect.
B. II is the cause and I is its effect.
C. Both I and II are effects of some common causes.
D. Both I and II are independent causes.
E. Both I and II are effects of independent causes.

Answer – C. Both I and II are effects of some common causes.

Explanation:

Both statements are the effects of some common cause. In this case, the cause is ‘melting’ or ‘sliding down of ice’ from the Himalayan mountain ranges. Some common climatic cause is associated with the occurrence and hence the effects have been manifested in the following way.

14. **In a class of 20 students, Shreya is 5 from the top and Annie is 7 ranks below Shreya. Find Annie’s rank from the bottom.**

A. 3
B. 5
C. 6
D. 9
E. None of these

Answer - D. 9

Explanation:

Annie is 12 ranks from top, so from bottom = 20-12 + 1 = 9

15. **Rita is sitting 5th from the left end of the row and Sita is 11th to the right of Rita with Tina being 4th to left of Sita. Madhuri is 8th to the right of Tina. What is the total number of students in the row if Madhuri is sitting at the extreme end?**

A. 20
B. 28
C. 23
D. 33
E. 12

Answer - A. 20

Explanation:

Rita is 5th from left, Sita 11th to the right of Rita, so 16th from the left end, Tina is 4th to left of Sita so Tina is 12th from the left end. Now Madhuri is 8th to the right of Tina, this means 20th from left, so 20 students

16. **Karuna is sitting 25th from the left end and Preeti is sitting 26th from the right end. Preeti is at 20th to the left of Karuna. What is the total number of students sitting in the row?**

A. 21
B. 32
C. 26
D. 28
E. 30

Answer - E. 30 **Explanation:**

Karuna is 25th from the left end and Preeti is 20th to left of Karuna so Preeti is 5th from the left end and given 26th from the right end, so total = (5+26) – 1 = 30

17. **In a row of 30 children, A is 11th from the right end of row. If there are 4 children between A and B, What is the position of B from the left end of the row?**

A. 5
B. 8
C. 4
D. 6
E. Cannot be determined

Answer - E. Cannot be determined **Explanation:**

Since it is not given that B is left of A or right of A, cant is determined.

18. **Prerna is 5th from the left end and Charu is 4th from the right of the row. Charu interchanges her position with the one who is sitting 3rd to the right of Prerna and now Charu is 10th from the right end. How many children are there in the row?**

A. 18
B. 20
C. 15
D. 17
E. 16

Answer - D. 17 **Explanation:**

Let A is 3rd to the right of Prerna, so Charu comes to his place so now charge is (5+3) = 8th from the left end and also she is 10th from the right end, so in total (8+10)-1 = 17

19. **Judge: Court:: Doctor:?**

A. Municipality
B. Hospital
C. Tehsil
D. Factory
E. None of these

Answer – B. Hospital **Explanation:**

The judge sits in court similarly doctor sits in a hospital

20. **Carbon dioxide: Extinguish:: Oxygen:?**

A. Foam
B. Explode
C. Burn
D. Isolate
E. None of these

Answer – C. Burn **Explanation:**

Oxygen supports Burning similarly Carbon dioxide helps in extinguishing

21. **Kitten: Cat:: Cub:?**

A. Tiger
B. Dear
C. Giraffe
D. Horse
E. None of these

Answer – A. Tiger

Explanation:

Kitten is the young one of Cat similarly Cub is the younger one of Tiger

22. **Den: lion:: stable: ?**

A. Cat
B. Dog
C. Horse
D. Cow
E. None of these

Answer – C. Horse **Explanation:**

Lion lives in den similarly Horse lives in Stable

23. **Thermometer: Temperature:: Barometer:?**

A. Stress
B. Pressure
C. Strain
D. Force
E. None of these

Answer – B. Pressure **Explanation:**

The thermometer is used to measure to Temperature similarly Barometer is used to measure Pressure

24. **In which year was Sugan born?**

I. Sugan's present age is 20 years more than his child

II. Sugans have two children. The first child was born in 1993?

A. Only I
B. Only II

C. Both I and II D. Either I or II E. Neither I or II

Answer – 5.Neither I or II

Explanation:

From I:
S = 20+Which child(not mentioned) From II:
Only data about first child

25. **How many students did secure First class in a class of 50 students?**

I. Double the number of first-class students secured second class and all the remaining students failed

II. Number of students failed was twice the number of first class students in the class

A. Only I

B. Only II

C. Both I and II D. Either I or II E. Neither I or II

Answer – 3.Both I and II **explanation:** II

From I:
No of Students Failed = 50-3x From II:
No of Students Failed = 2x
2x = 50-3x
5x=50
X = 10

Doon Coding Questions

#1) Write a Java Program to reverse a string without using String inbuilt function.

Answer: Here, we are initializing a string variable str and making use of the string builder class.

The object of the string builder class str2 will be further used to append the value stored in the string variable str.

Thereafter, we are using the inbuilt function of the string builder (reverse()) and storing the new reversed string in str2. Finally, we are printing str2.

Following program code explains this:

```
public class FinalReverseWithoutUsingStringMethods {

      public static void main(String[] args) {
           // TODO Auto-generated method stub
           String str = "Automation";
           StringBuilder str2 = new StringBuilder();
           str2.append(str);
           str2 = str2.reverse();      // used string builder to reverse
          System.out.println(str2);
           }

 }
```

Output:

noitamotuA

Q #2) Write a Java Program to reverse a string without using String inbuilt function reverse().

Answer: There are several ways with which you can reverse your string if you are allowed to use the other string inbuilt functions.

Method 1:

In this method, we are initializing a string variable called str with the value of your given string. Then, we are converting that string into a character array with the toCharArray() function. Thereafter, we are using for loop to iterate between each character in reverse order and printing each character.

```
public class FinalReverseWithoutUsingInbuiltFunction {
    public static void main(String[] args) {
        String str = "Saket Saurav";
        char chars[] = str.toCharArray();  // converted to character array
and printed in reverse order
        for(int i= chars.length-1; i&gt;=0; i--) {
            System.out.print(chars[i]);
        }
    }
}
```

Output:

varuaS tekaS

Method 2:

This is another method in which you are declaring your string variable str and then using Scanner class to declare an object with a predefined standard input object.

This program will accept the string value through the command line (when executed).

We have used nextLine() which will read the input with the spaces between the words of a string. Thereafter, we have used a split() method to split the string into its substrings(no delimiter given here). Finally, we have printed the string in reverse order using for loop.

```
import java.util.Scanner;

public class ReverseSplit {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String str;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter your String");
        str = in.nextLine();
        String[] token = str.split("");    //used split method to print in
reverse order
        for(int i=token.length-1; i&gt;=0; i--)
        {
            System.out.print(token[i] + "");
        }

    }

}
```

Output:

Enter your String

Softwaretestinghelp

plehgnitseterawtfoS

Method 3:

This is almost like method 2, but here we did not use the split() method. We have used the scanner class and nextLine() for reading the input string. Then, we have declared an integer length which has the length of the input string.

Thereafter, we have printed the string in the reverse order using for loop. However, we have used the charAt(index) method which will return the character at any specific index. After each iteration, the character will be concatenated to reverse the string variable.

Finally, we have printed the reverse string variable.

```
import java.util.Scanner;

public class Reverse {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String original, reverse = "";
        System.out.println("Enter the string to be reversed");
        Scanner in = new Scanner(System.in);
        original = in.nextLine();
        int length = original.length();
        for(int i=length-1; i&gt;=0; i--) {
            reverse = reverse + original.charAt(i);   //used inbuilt method
charAt() to reverse the string
        }
        System.out.println(reverse);
    }

}
```

Output:

Enter the string to be reversed

automation testing

gnitset noitamotua

Fun Facts To Know

#KAAHANI

Elimination of Pointers from java

Pointers lead to confusion for a programmer.
Pointers may crash a program easily, for example, when we add two pointers, the program crashers immediately.
Pointers break security. Using pointers, harmful programs like Virus and other hacking programs can be developed.
Because of the above reasons, pointers have been eliminated from java.