**ST-2 Core Java (CSL5303) SET A**

**MCQs**

1. What is the name of the method used to start a thread execution?
2. init()
3. run()
4. **start()**
5. resume()
6. Determine the output of following java code.

import java.io.\*;

public class Mcq {

public static void main(String[] args) {

String obj = "java";

byte b[] = obj.getBytes();

ByteArrayInputStream obj1 = new ByteArrayInputStream(b);

for (inti = 0; i< 2; ++ i) {

int c;

while ((c = obj1.read()) != -1) {

if(i == 0) {

System.out.print((char)c);

}

}

}

}

}

1. **java**
2. j
3. ja
4. None of the above
5. Which among the following is valid constructor for Thread?
6. **Thread(Runnable r, String name)**
7. Thread(int priority)
8. Thread(Runnable r, ThreadGroup g)
9. Thread(Runnable r, int priority)
10. Determine the output of following java code.

public class Mcq{

public static void main(String[] args) {

TreeMap<String, Integer>obj = new TreeMap<>();

obj.put("DS", 1);

obj.put("Java", 2);

obj.put("SE", 3);

System.out.println(obj.entrySet());

}

}

1. **[DS=1, Java=2, SE=3]**
2. {DS=1, Java=2, SE=3}
3. [1,2, 3]
4. [DS, Java, SE]
5. Which of these methods deletes all the elements from invoking collection?
6. **clear()**
7. reset()
8. refresh()
9. None of the above
10. Which of these classes is not part of Java’s collection framework?
11. **Map**
12. Array
13. Stack
14. Queue
15. The default capacity of a Vector is:
16. **10**
17. 12
18. 8
19. 16
20. How many threads are created in following java code:

classTest extendsThread {

publicvoidrun()  {

        System.out.println("Run");

    }

}

classMyclass {

publicstaticvoidmain(String[] args)   {

        Test t = newTest();

        t.run();

    }

}

1. **One thread created**
2. Two threads created
3. Depend upon system
4. No thread created
5. Determine the output of following java code.

importjava.util.\*;

public class Mcq {

public static void main(String[] args) {

TreeSet<String> map = new TreeSet<>();

map.add("one");

map.add("two");

map.add("three");

map.add("four");

map.add("one");

Iterator<String> it = map.iterator();

while (it.hasNext() ) {

System.out.print(it.next() + " " );

}

}

}

1. one two three four
2. four three two one
3. **four one three two**
4. one two three four one
5. Determine the output of following java code.

public class Mcq {

public static void main(String[] args) {

ArrayList<String> obj1 = new ArrayList<>();

ArrayList<String> obj2 = new ArrayList<>();

obj1.add("A");

obj1.add("B");

obj2.add("A");

obj2.add(1, "B");

System.out.println(obj1.equals(obj2));

}

}

1. 0
2. 1
3. false
4. **true**
5. Determine the output of following java code.

public class Mcq extends Thread implements Runnable {

public void run() {

System.out.print("Program ");

}

public static void main(String[] args) throws InterruptedException {

Mcqobj = new Mcq();

obj.run();

obj.start();

}

}

1. Program
2. Runtime error
3. **Program Program**
4. Compilation error
5. Determine the output of following java code.

importjava.util.\*;

public class Mcq {

public static void main(String[] args) {

List<String> list1 = new LinkedList<>();

list1.add("Coding");

list1.add("in");

list1.add("Java");

list1.add("ST2");

list1.add("CodingInJava");

List<String> list2 = new LinkedList<>();

list2.add("Coding");

list1.removeAll(list2);

for (String temp : list1)

System.out.printf(temp + " ");

}

}

1. Codingin Java ST2 CodingInJava
2. **in Java ST2 CodingInJava**
3. Coding
4. in Java ST2
5. Determine the output of following java code.

public class Mcq implements Runnable {

int x, y;

public void run() {

for(inti = 0; i< 1000; i++)

synchronized(this) {

x = 20;

y = 20;

}

System.out.print(x + " " + y + " ");

}

public static void main(String args[]) {

Mcq1 run = new Mcq1();

Thread t1 = new Thread(run);

Thread t2 = new Thread(run);

t1.start();

t2.start();

}

}

1. 20 20
2. Compilation Error
3. Cannot determine output
4. **20 20 20 20**
5. What is the output of the following program?

importjava.util.\*;

public class priorityQueue {

public static void main(String[] args) {

PriorityQueue<Integer> queue = new PriorityQueue<>();

queue.add(11);

queue.add(10);

queue.add(22);

queue.add(5);

queue.add(12);

queue.add(2);

while (queue.isEmpty() == false)

System.out.printf("%d ", queue.remove());

}

}

1. 11 10 22 5 12 2
2. 2 12 5 22 10 11
3. **2 5 10 11 12 22**
4. 22 12 11 10 5 2
5. What is the output of the following program?

import java.io.\*;

public class Mcq {

public static void main(String[] args) {

String obj = "programming";

int length = obj.length();

char c[] = new char[length];

obj.getChars(0, length, c, 0);

CharArrayReader input1 = new CharArrayReader(c);

CharArrayReader input2 = new CharArrayReader(c, 0, 3);

inti;

try {

while ((i = input2.read()) != -1) {

System.out.print((char)i);

}

}

catch (IOException e) {

e.printStackTrace();

}

}

}

1. **pro**
2. programming
3. prog
4. None of the mentioned

***16. Sorting Apples By Multiple Characteristics***

Owner of an Apple farm wants to organize the apples for distribution in the market. First, he decided to sort the apple based on variety (type). While sorting, he realized there are many apples of same variety. So, he decided to include another property of apples in sorting. Thus, color of apples was included in the sorting criteria. Same problem arises in case of color also. At last, weight property was included to make the correct comparison among the apples.

Manually managing this task is extremely tedious. The owner decided to take help from java programmer so that he can design a code which could help him in performing this task automatically.

Input Format

The first line of sample input denotes N, Number of Apple types

Next three lines represent the variety (type) of apple, color & weight for each Apple type.

Output Format

Display output in sorted order according to variety (type) of apple or color or weight, depending upon the input and is separated by space.

*Explanation:*

Sample Input 1 is sorted according to variety (type) of apple.

Sample Input 2 is sorted according to color of apple as variety of apple is same.

Sample Input 3 is sorted according to weight of apple as variety and color is same.

Sample Input 4 is sorted according to variety (type) of apple and for same variety further sorted according to color.

*Sample Input 1*

3

RedDelicious

Red

120

Gala

YellowRed

100

Fuji

DarkRed

90

*Sample Output 1*

Fuji DarkRed 90

Gala YellowRed 100

RedDelicious Red 120

*Sample Input 2*

2

Gala

RedYellow

111

Gala

YellowRed

100

*Sample Output 2*

Gala RedYellow 111

Gala YellowRed 100

*Sample Input 3*

3

GoldenDelicious

YellowGreen

85

GoldenDelicious

YellowGreen

90

GoldenDelicious

YellowGreen

80

*Sample Output 3*

GoldenDelicious YellowGreen 80

GoldenDelicious YellowGreen 85

GoldenDelicious YellowGreen 90

*Sample Input 4*

3

RedDelicious

Red

100

RedDelicious

PinkRed

120

Fuji

DarkRed

90

*Sample Output 4*

Fuji DarkRed 90

RedDelicious PinkRed 120

RedDelicious Red 100

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3  Gala  YellowRed  100  GoldenDelicious  GoldenGreen  90  RedDelicious  Red  120 | 2  Gala  RedYellow  111  HoneyCrisp  YellowRed  111 | 4  CrippsPink  Red  95  CrippsPink  Red  99  HoneyCrisp  YellowRed  98  Gala  YellowRed  95 | 3  CrippsPink  Red  98  CrippsPink  Red  95  CrippsPink  Red  97 | 3  Ambrosia  PinkRed  95  Ambrosia  Red  98  Ambrosia  RedPink  97 |
| Gala YellowRed 100  GoldenDelicious GoldenGreen 90  RedDelicious Red 120 | Gala RedYellow 111  HoneyCrisp YellowRed 111 | CrippsPink Red 95  CrippsPink Red 99  Gala YellowRed 95  HoneyCrisp YellowRed 98 | CrippsPink Red 95  CrippsPink Red 97  CrippsPink Red 98 | Ambrosia PinkRed 95  Ambrosia Red 98  Ambrosia RedPink 97 |

***Solution:***

**import** java.util.\*;

**class** Apples **implements** Comparable<Apples> {

**private** String variety,color;

**private** **int** weight;

**public void setVariety(String variety){**

**this.variety=variety;**

**}**

**public void setColor(String color){**

**this.color=color;**

**}**

**public void setWeight(int weight){**

**this.weight=weight;**

**}**

**public String getVariety(){**

**return this.variety;**

**}**

**public String getColor(){**

**return this.color;**

**}**

**public int getWeight(){**

**return this.weight;**

**}**

**public int compareTo(Apples other) {**

**int result = this.variety.compareTo(other.variety);**

**if (result != 0) {**

**return result;**

**}**

**if (result == 0) {**

**System.*out*.println("0");**

**result = this.color.compareTo(other.color);**

**}**

**if (result != 0) {**

**return result;**

**}**

**if (result == 0) {**

**result = Integer.*compare*(this.weight, other.weight);**

**}**

**return result;**

**}**

**}**

**public** **class** file{

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

Scanner scan=**new** Scanner(System.*in*);

List<Apples> la=**new** ArrayList<Apples>();

String var,col;

**int** wt;

**int** n=scan.nextInt(); //No. of inputs

**for**(**int** i=0;i<n;i++){

Apples a=**new** Apples();

var=scan.next();

col=scan.next();

wt=scan.nextInt();

a.setVariety(var);

a.setColor(col);

a.setWeight(wt);

la.add(a);

}

Collections.*sort*(la);

**for**(**int** i=0;i<la.size();i++){

Apples ap=la.get(i);

System.*out*.println(ap.getVariety()+" "+ap.getColor()+" "+ap.getWeight());

}

}

}

***17. Second Smallest and Largest***

Write a java program to find the second largest and second smallest number in the list and compute the average of these two numbers. Then check if this average exists in the list or not. If exists then how many times it has occurred in the list.

Input Format

The input consists of n integers.

The first line corresponds to n, the number of elements to be added in the list.

The remaining integers correspond to the elements in the list.

*(Refer Sample Input below for explanation)*

Output Format

The first line of output denotes the second largest and second smallest element separated by space.

The second line display the average of these two numbers and next line must contain the count value i.e. 0 (zero) if average does not exist in the array otherwise the count value means how much time average value occurs in the array.

*(Refer Sample Output below for explanation)*

Sample Input 1

5

388

345

444

416

654

Sample Output 1

444 388

416

1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7  -122  198  675  -329  568  321  432 | 10  45  65  34  87  12  8  40  8  5  15 | 8  12  10  5  8  11  8  4  8 | 5  67  22  11  33  45 | 3  78  43  -13 |
| 568 -122  223  0 | 65 8  36  0 | 11 5  8  3 | 45 22  33  1 | 43 43  43  1 |

***Solution:***

**import**java.util.\*;

**publicclass**file{

**staticintreturnAverage(List<Integer>ls) {**

**Collections.*sort*(ls);**

**System.*out*.println(ls.get(ls.size()-2)+" "+ls.get(1));**

**intavg = ((ls.get(1) + ls.get(ls.size() - 2)))/ 2;**

**returnavg;**

**}**

**staticintfindAverageInList(List<Integer>ls, intav) {**

**int counter=0;**

**for(int i=0;i<ls.size();i++){**

**if(av==ls.get(i)){**

**counter++;**

**}**

**}**

**return counter;**

**}**

**publicstaticvoid** main(String args[]) {

Scanner scan=**new**Scanner(System.***in***);

List<Integer>li=**new**ArrayList<>();

**int**n=scan.nextInt();

**for**(**int**i=0;i<n;i++) {

**int**num=scan.nextInt();

li.add(num);

}

**int**avg=*returnAverage*(li);

System.***out***.println(avg);

**int**count=*findAverageInList*(li,avg);

System.***out***.println(count);

}

}

***18. Next Highest Element***

Given an array, print the Next Highest Element (NHE) for every element. The Next highest Element for an element x is the first greater element on the right side of x in array. Elements for which no greater element exist, consider next highest element as -1.

Examples:

a) For any array, rightmost element always has next highest element as -1.

b) For an array which is sorted in decreasing order, all elements have next highest element as -1.

c) For the input array [4, 5, 2, 25], the next greater elements for each element are as follows:

Element NHE

4 --> 5

5 --> 25

2 --> 25

25 --> -1

*Input Format*

First Line of the Sample Input denotes the number of elements, N of array.

Next N lines represents the array elements.

(Refer Sample Input below for the explanation)

*Output Format*

Display the Next Highest Element (NHE) for every element of array

(Refer Sample Output below for the explanation)

*Sample Input 1*

4

11

4

25

10

*Sample Output 1*

25

25

-1

-1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5  12  12  67  1  78 | 4  1  2  3  4 | 5  -1  7  11  -6  0 | 6  1  6  1  6  1  6 | 7  44  -128  23  240  0  -89  900 |
| 67  67  78  78  -1 | 2  3  4  -1 | 7  11  -1  0  -1 | 6  -1  6  -1  6  -1 | 240  23  240  900  900  900  -1 |

***Solution***:

**import**java.util.\*;

**public class** file{

**static** List<Integer>printNHE(List<Integer>list) {

**Stack<Integer> s = new Stack<>();**

**List<Integer>nge=newArrayList<>(list);**

**for (int i = list.size() - 1; i >= 0; i--) {**

**if (!s.empty()) {**

**while (!s.empty() &&s.peek() <= list.get(i)) {**

**s.pop();**

**}**

**}**

**if(s.empty()){**

**nge.set(i,-1);**

**}**

**else {**

**nge.set(i,s.peek());**

**}**

**s.push(list.get(i));**

**}**

**returnnge;**

}

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

Scanner sc=**new**Scanner(System.***in***);

**int**n=sc.nextInt();

List<Integer>list=**new**ArrayList<Integer>();

**for**(**int**i=0;i<n;i++){

**int**num=sc.nextInt();

list.add(num);

}

List<Integer>ls=*printNHE*(list);

**for**(**int**i = 0; i<ls.size(); i++)

System.***out***.println(ls.get(i));

}

}