

ArrayList and LinkedList:

1. Create an ArrayList to store the names of students in a class. Add, remove, and print the list of students.

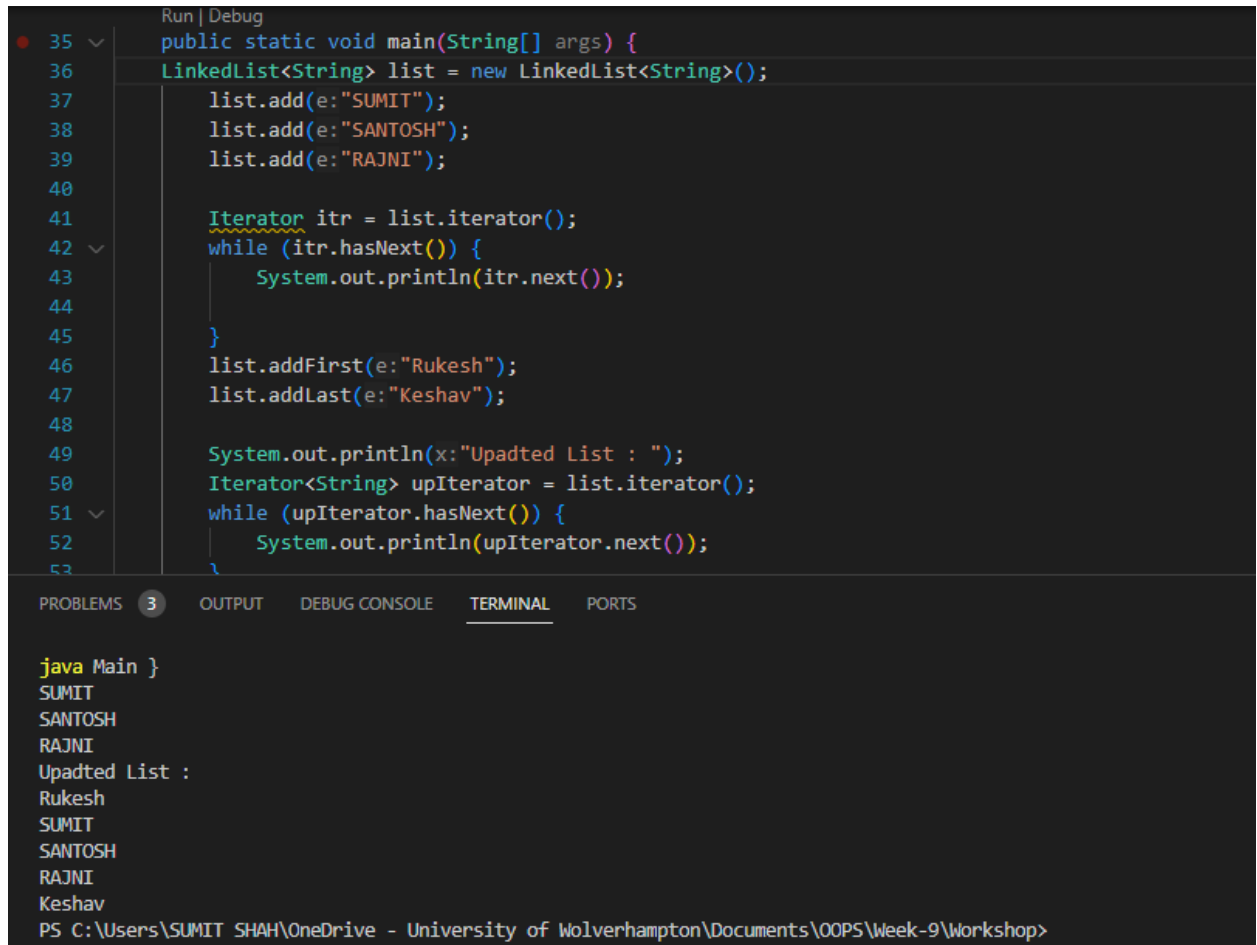
- Initialize an empty ArrayList to store examinee names.
- Add the names of five examinee participating in the exam to the ArrayList.
- Remove the name of the examinee who withdrew from the exam.
- Print the updated list of participants.

```
9  import java.util.ArrayList;
10
11  public class Main {
12      public static void main(String[] args) {
13          ArrayList<String> studentList = new ArrayList<>();
14          studentList.add(e:"Santosh");
15          studentList.add(e:"Sumit");
16          studentList.add(e:"Arayan");
17          studentList.add(e:"Rajni");
18          studentList.add(e:"Barsigh");
19
20          System.out.println("Initial List : " + studentList);
21
22          String withdrawnStudent = "Arayan";
23          studentList.remove(withdrawnStudent);
24
25          System.out.println("Updated List : " + studentList);
26      }
27  }
28
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Updated List : [Santosh, Sumit, Rajni, Barsigh]
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop\" ; if (\$?) { javac Main.java } ; if (\$?) { java Main }
Initial List : [Santosh, Sumit, Arayan, Rajni, Barsigh]
Updated List : [Santosh, Sumit, Rajni, Barsigh]
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>

2. Write a program to insert elements into the linked list at the first and last positions. Also check if the linked list is empty or not.



```
Run | Debug
35 public static void main(String[] args) {
36     LinkedList<String> list = new LinkedList<String>();
37     list.add(e:"SUMIT");
38     list.add(e:"SANTOSH");
39     list.add(e:"RAJNI");
40
41     Iterator itr = list.iterator();
42     while (itr.hasNext()) {
43         System.out.println(itr.next());
44     }
45     list.addFirst(e:"Rukesh");
46     list.addLast(e:"Keshav");
47
48     System.out.println(x:"Upadted List : ");
49     Iterator<String> upIterator = list.iterator();
50     while (upIterator.hasNext()) {
51         System.out.println(upIterator.next());
52     }
53 }
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
java Main }
SUMIT
SANTOSH
RAJNI
Upadted List :
Rukesh
SUMIT
SANTOSH
RAJNI
Keshav
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

3. Rotate the elements of an ArrayList to the right by a given number of positions. For example, if the ArrayList is [1, 2, 3, 4, 5] and you rotate it by 2 positions, the result should be [4, 5, 1, 2, 3].

```
64 // 3. Rotate the elements of an ArrayList to the right by a given number of positions.
65 // For example, if the ArrayList is [1, 2, 3, 4, 5] and you rotate it by 2 positions, the result should be [4, 5, 1, 2, 3].
Run|Debug
66 public static void main(String[] args) {
67     ArrayList<Integer> arrayList = new ArrayList<>();
68     arrayList.add(e:1);
69     arrayList.add(e:2);
70     arrayList.add(e:3);
71     arrayList.add(e:4);
72     arrayList.add(e:5);
73
74     int rotateBy = 2;
75
76     System.out.println("Original List : " + arrayList);
77
78     Collections.rotate(arrayList, rotateBy);
79
80     System.out.println("Rotated List : " + arrayList);
81
82
83 }
84 }
85
86
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Original List : [1, 2, 3, 4, 5]
Rotated List : [4, 5, 1, 2, 3]
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

5. Write a program to declare a linkedList, colors to store String. Insert five colors into the linked list.
- Iterate and print all the colors.
 - Check if “Red” exists in the linkedList or not.
 - Shuffle the elements of the list and print them.
 - Print the LinkedList in ascending order

```
import java.util.LinkedList;
import java.util.Collections;
import java.util.Iterator;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        // Create a LinkedList to store colors
        LinkedList<String> colors = new LinkedList<>();

        // Insert five colors into the linked list
        colors.add(e:"Red");
        colors.add(e:"Blue");
        colors.add(e:"Green");
        colors.add(e:"Yellow");
        colors.add(e:"Orange");

        // a. Iterate and print all the colors
        System.out.println(x:"All colors:");
        for (String color : colors) {
            System.out.println(color);
        }
    }
}
```

```
// a. Iterate and print all the colors
System.out.println(x:"All colors:");
for (String color : colors) {
    System.out.println(color);
}

// b. Check if "Red" exists in the linked list or not
if (colors.contains(o:"Red")) {
    System.out.println(x:"Red exists in the linked list.");
} else {
    System.out.println(x:"Red does not exist in the linked list.");
}

// c. Shuffle the elements of the list and print them
Collections.shuffle(colors);
System.out.println(x:"\nShuffled colors:");
for (String color : colors) {
    System.out.println(color);
}
```

```
// d. Print the LinkedList in ascending order
Collections.sort(colors);
System.out.println(x:"\nColors in ascending order:");
for (String color : colors) {
    System.out.println(color);
}
}
```

Stack:

6. Create a Stack to manage a sequence of tasks. Implement the following operations:
 - a. Push the tasks "Read", "Write", and "Code" onto the stack.
 - b. Pop a task from the stack.
 - c. Push tasks "Debug" and "Test" onto the stack.
 - d. Peek at the top task without removing it.
 - e. Print the stack.

```
import java.util.Stack;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        // Create a Stack to manage tasks
        Stack<String> tasks = new Stack<>();

        // a. Push the tasks "Read", "Write", and "Code" onto the stack
        tasks.push(item:"Read");
        tasks.push(item:"Write");
        tasks.push(item:"Code");

        // b. Pop a task from the stack
        String poppedTask = tasks.pop();
        System.out.println("Popped task: " + poppedTask);

        // c. Push tasks "Debug" and "Test" onto the stack
        tasks.push(item:"Debug");
        tasks.push(item:"Test");
```

```
174         // c. Push tasks "Debug" and "Test" onto the stack
175         tasks.push(item:"Debug");
176         tasks.push(item:"Test");
177
178         // d. Peek at the top task without removing it
179         String topTask = tasks.peek();
180         System.out.println("Top task: " + topTask);
181
182         // e. Print the stack
183         System.out.println(x:"\nTasks on the stack:");
184         for (String task : tasks) {
185             System.out.println(task);
186         }
187     }
188 }
189
190
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Tasks on the stack:

Read
Write
Debug
Test

PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>

7. Write a program that reverses the order of words in a sentence using a Stack. For example, if the input is "Hello World", the output should be "World Hello".

```
// 7. Write a program that reverses the order of words in a sentence using a Stack. For example,  
// if the input is "Hello World", the output should be "World Hello".  
import java.util.Stack;  
  
public class Main {  
    Run | Debug  
    public static void main(String[] args) {  
        String input = "Hello World";  
        String reversedSentence = reverseSentence(input);  
        System.out.println("Reversed sentence: " + reversedSentence);  
    }  
  
    public static String reverseSentence(String sentence) {  
        // Split the sentence into words  
        String[] words = sentence.split(regex: " ");  
  
        // Create a stack to store the words  
        Stack<String> wordStack = new Stack<>();  
        for (String word : words) {  
            wordStack.push(word);  
        }  
  
        // Reconstruct the sentence by popping words from the stack  
        StringBuilder reversedSentence = new StringBuilder();  
        while (!wordStack.isEmpty()) {  
            reversedSentence.append(wordStack.pop()).append(str: " ");  
        }  
    }  
}
```

```
214  
215     // Create a stack to store the words  
216     Stack<String> wordStack = new Stack<>();  
217     for (String word : words) {  
218         wordStack.push(word);  
219     }  
220  
221     // Reconstruct the sentence by popping words from the stack  
222     StringBuilder reversedSentence = new StringBuilder();  
223     while (!wordStack.isEmpty()) {  
224         reversedSentence.append(wordStack.pop()).append(str: " ");  
225     }  
226  
227     // Remove trailing space and return the reversed sentence  
228     return reversedSentence.toString().trim();  
229 }  
230  
231
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Reversed sentence: World Hello

PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> []

Queue

8. Imagine a scenario where a printer is managing print jobs. Create a Queue to handle these print jobs. Implement the following operations:

- Enqueue print jobs "Document1", "Document2", and "Document3" into the print queue.
- Dequeue a print job from the front of the queue.
- Enqueue print jobs "Document4" and "Document5" into the print queue.
- Peek at the next print job without removing it.
- Print the list of print jobs in the queue.


```
250 import java.util.LinkedList;
251 import java.util.Queue;
252
253 public class Main {
254     public static void main(String[] args) {
255         // Create a Queue to manage print jobs
256         Queue<String> printQueue = new LinkedList<>();
257
258         // Enqueue print jobs "Document1", "Document2", and "Document3" into the print queue
259         printQueue.add(e:"Document1");
260         printQueue.add(e:"Document2");
261         printQueue.add(e:"Document3");
262
263         // Dequeue a print job from the front of the queue
264         String dequeuedJob = printQueue.poll();
265         System.out.println("Dequeued job: " + dequeuedJob);
266
267         // Enqueue print jobs "Document4" and "Document5" into the print queue
268         printQueue.add(e:"Document4");
269         printQueue.add(e:"Document5");
270
271         // Peek at the next print job without removing it
272         String nextJob = printQueue.peek();
273         System.out.println("Next job: " + nextJob);
274
275         // Print the list of print jobs in the queue
276         System.out.println("\nPrint jobs in the queue:");
277         for (String job : printQueue) {
278             System.out.println(job);
279         }
280     }
281 }
282
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Dequeued job: Document1
Next job: Document2

Print jobs in the queue:
Document2
Document3
Document4
Document5

PS C:\Users\SUMIT_SHARMA\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop >

Set Operations

9. Implement a TreeSet to store unique names in alphabetical order.
10. Consider a scenario where you have two sets, each representing a group of animals. Implement a Java program to perform set operations (Union, Intersection, and Difference) on these sets:
 - Initialize two HashSet objects: **set1** with elements "Dog," "Cat," "Elephant," and "Lion," and **set2** with elements "Cat," "Giraffe," "Dog," and "Monkey."
 - Implement a method performUnion that takes two sets and returns their union.

- Implement a method `performIntersection` that takes two sets and returns their intersection.
- Implement a method `performDifference` that takes two sets and returns the difference of the first set from the second set.
- Print the original sets, the union, intersection, and difference of the sets.

```

329 import java.util.HashSet;
330 import java.util.Set;
331
332 public class Main {
333     public static void main(String[] args) {
334         Set<String> set1 = new HashSet<>();
335         Set<String> set2 = new HashSet<>();
336
337         set1.add(e: "Dog");
338         set1.add(e: "Cat");
339         set1.add(e: "Elephant");
340         set1.add(e: "Lion");
341
342         set2.add(e: "Cat");
343         set2.add(e: "Giraffe");
344         set2.add(e: "Dog");
345         set2.add(e: "Monkey");
346
347         System.out.println("Original set1: " + set1);
348         System.out.println("Original set2: " + set2);
349
350         Set<String> union = performUnion(set1, set2);
351         System.out.println("Union: " + union);
352
353         Set<String> intersection = performIntersection(set1, set2);
354         System.out.println("Intersection: " + intersection);
355
356         Set<String> difference = performDifference(set1, set2);
357         System.out.println("Difference (set1 - set2): " + difference);
358     }
}

360 public static <T> Set<T> performUnion(Set<T> set1, Set<T> set2) {
361     Set<T> union = new HashSet<>(set1);
362     union.addAll(set2);
363     return union;
364 }
365
366 public static <T> Set<T> performIntersection(Set<T> set1, Set<T> set2) {
367     Set<T> intersection = new HashSet<>(set1);
368     intersection.retainAll(set2);
369     return intersection;
370 }
371
372 public static <T> Set<T> performDifference(Set<T> set1, Set<T> set2) {
373     Set<T> difference = new HashSet<>(set1);
374     difference.removeAll(set2);
375     return difference;
376 }
377 }
378
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Document
s\OOPS\Week-9\Workshop\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Original set1: [Cat, Elephant, Lion, Dog]
Original set2: [Cat, Monkey, Dog, Giraffe]
Union: [Cat, Elephant, Monkey, Lion, Dog, Giraffe]
Intersection: [Cat, Dog]
Difference (set1 - set2): [Elephant, Lion]

```

Map(HashMap, LinkedHashMap, TreeMap):

11. Write a program that uses a HashMap to store contact information (name and phone number).

```
392 import java.util.HashMap;
393 import java.util.Map;
394
395 public class Main {
    Run | Debug
396     public static void main(String[] args) {
397         Map<String, String> contacts = new HashMap<>();
398         contacts.put(key:"Sumit Shah", value:"9818257300");
399         contacts.put(key:"Rajni Shah", value:"9861446442");
400         contacts.put(key:"Santosh Shah", value:"9818257400");
401
402         System.out.println(x:"Contact Information:");
403         contacts.forEach((name, phoneNumber) -> System.out.println("Name: " + name + ", Phone Number: " + phoneNumber));
404
405         String sumitPhoneNumber = contacts.get(key:"Sumit Shah");
406         System.out.println("\nPhone number for Sumit Shah: " + sumitPhoneNumber);
407
408         String nameToCheck = "Rajni Shah";
409         System.out.println((contacts.containsKey(nameToCheck) ? nameToCheck + " exists in contacts." : nameToCheck + " does not exist in contacts");
410     }
411 }
412
413
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Name: Rajni Shah, Phone Number: 9861446442
Name: Sumit Shah, Phone Number: 9818257300
Name: Santosh Shah, Phone Number: 9818257400

Phone number for Sumit Shah: 9818257300
Rajni Shah exists in contacts.
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

12. Imagine a scenario where you are managing information about countries and their capitals using a HashMap. Perform the following tasks:

- Initialize a HashMap called countryCapitals to store the capitals of different countries. Add at least five country-capital pairs.
- Implement a method called printMap that takes a HashMap and prints all the key-value pairs.
- Implement a method called getCapital that takes a country name as a parameter and returns its capital from the countryCapitals map.

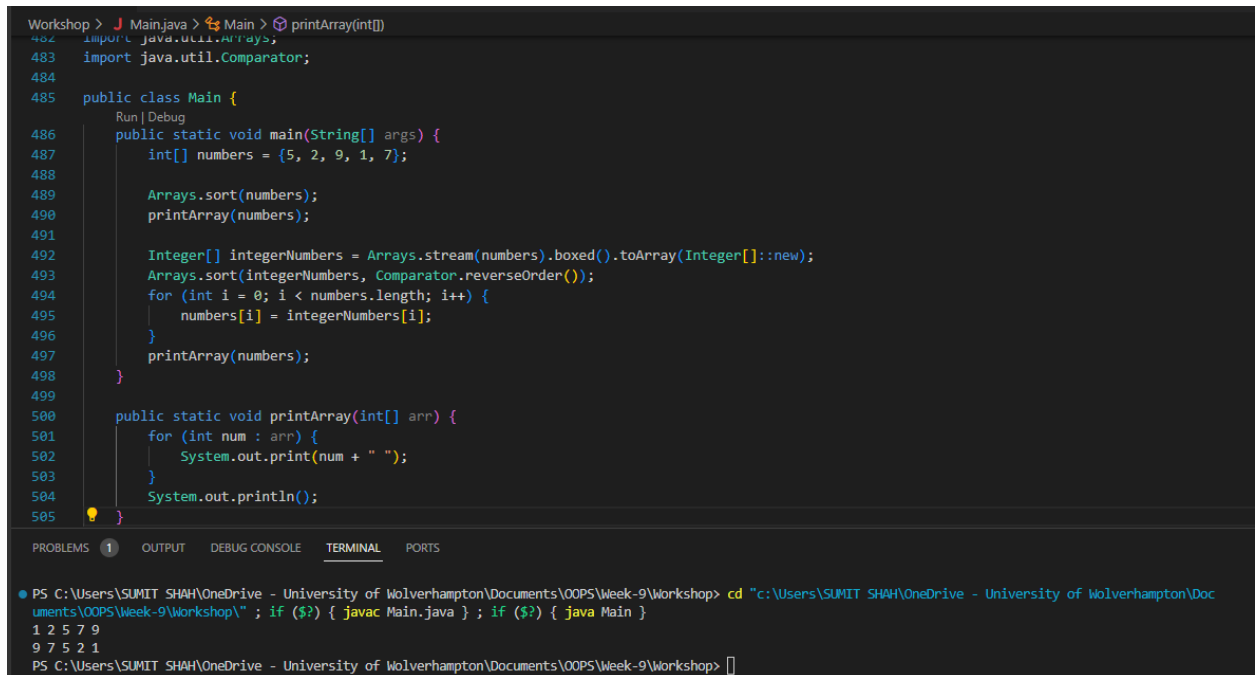
- Implement a method called `containsCapital` that takes a capital name as a parameter and returns whether that capital exists in the `countryCapitals` map.
- Iterate through the `countryCapitals` map and print each country and its capital.

```
Search (Ctrl+Shift+F) 1c static void main(String[] args) {  
432     Map<String, String> countryCapitals = new HashMap<>();  
433     countryCapitals.put(key:"USA", value:"Washington, D.C.");  
434     countryCapitals.put(key:"UK", value:"London");  
435     countryCapitals.put(key:"France", value:"Paris");  
436     countryCapitals.put(key:"Germany", value:"Berlin");  
437     countryCapitals.put(key:"Japan", value:"Tokyo");  
438  
439     printMap(countryCapitals);  
440  
441     String capitalForUK = getCapital(country:"UK", countryCapitals);  
442     System.out.println("Capital of UK: " + capitalForUK);  
443  
444     String capitalToCheck = "Paris";  
445     System.out.println("Does '" + capitalToCheck + "' exist in the map? " + containsCapital(capitalToCheck, countryCapitals));  
446  
447     System.out.println("\nCountry-Capital Information:");  
448     for (Map.Entry<String, String> entry : countryCapitals.entrySet()) {  
449         System.out.println("Country: " + entry.getKey() + ", Capital: " + entry.getValue());  
450     }  
451 }  
452  
453 public static void printMap(Map<String, String> map) {  
454     for (Map.Entry<String, String> entry : map.entrySet()) {  
455         System.out.println(entry.getKey() + " - " + entry.getValue());  
456     }  
457 }  
458  
459 public static String getCapital(String country, Map<String, String> map) {  
460  
461 }  
462 }  
463 }  
464 }  
465 }  
466 }  
467 }  
468 }  
469 }  
470 }  
471 }  
472 }  
473 }  
474 }  
475 }  
476 }  
477 }  
478 }  
479 }  
480 }  
481 }  
482 }  
483 }  
484 }  
485 }  
486 }  
487 }  
488 }  
489 }  
490 }  
491 }  
492 }  
493 }  
494 }  
495 }  
496 }  
497 }  
498 }  
499 }  
500 }  
501 }  
502 }  
503 }  
504 }  
505 }  
506 }  
507 }  
508 }  
509 }  
510 }  
511 }  
512 }  
513 }  
514 }  
515 }  
516 }  
517 }  
518 }  
519 }  
520 }  
521 }  
522 }  
523 }  
524 }  
525 }  
526 }  
527 }  
528 }  
529 }  
530 }  
531 }  
532 }  
533 }  
534 }  
535 }  
536 }  
537 }  
538 }  
539 }  
540 }  
541 }  
542 }  
543 }  
544 }  
545 }  
546 }  
547 }  
548 }  
549 }  
550 }  
551 }  
552 }  
553 }  
554 }  
555 }  
556 }  
557 }  
558 }  
559 }  
560 }  
561 }  
562 }  
563 }  
564 }  
565 }  
566 }  
567 }  
568 }  
569 }  
570 }  
571 }  
572 }  
573 }  
574 }  
575 }  
576 }  
577 }  
578 }  
579 }  
580 }  
581 }  
582 }  
583 }  
584 }  
585 }  
586 }  
587 }  
588 }  
589 }  
590 }  
591 }  
592 }  
593 }  
594 }  
595 }  
596 }  
597 }  
598 }  
599 }  
600 }  
601 }  
602 }  
603 }  
604 }  
605 }  
606 }  
607 }  
608 }  
609 }  
610 }  
611 }  
612 }  
613 }  
614 }  
615 }  
616 }  
617 }  
618 }  
619 }  
620 }  
621 }  
622 }  
623 }  
624 }  
625 }  
626 }  
627 }  
628 }  
629 }  
630 }  
631 }  
632 }  
633 }  
634 }  
635 }  
636 }  
637 }  
638 }  
639 }  
640 }  
641 }  
642 }  
643 }  
644 }  
645 }  
646 }  
647 }  
648 }  
649 }  
650 }  
651 }  
652 }  
653 }  
654 }  
655 }  
656 }  
657 }  
658 }  
659 }  
660 }  
661 }  
662 }  
663 }  
664 }  
665 }  
666 }  
667 }  
668 }  
669 }  
670 }  
671 }  
672 }  
673 }  
674 }  
675 }  
676 }  
677 }  
678 }  
679 }  
680 }  
681 }  
682 }  
683 }  
684 }  
685 }  
686 }  
687 }  
688 }  
689 }  
690 }  
691 }  
692 }  
693 }  
694 }  
695 }  
696 }  
697 }  
698 }  
699 }  
700 }  
701 }  
702 }  
703 }  
704 }  
705 }  
706 }  
707 }  
708 }  
709 }  
710 }  
711 }  
712 }  
713 }  
714 }  
715 }  
716 }  
717 }  
718 }  
719 }  
720 }  
721 }  
722 }  
723 }  
724 }  
725 }  
726 }  
727 }  
728 }  
729 }  
730 }  
731 }  
732 }  
733 }  
734 }  
735 }  
736 }  
737 }  
738 }  
739 }  
740 }  
741 }  
742 }  
743 }  
744 }  
745 }  
746 }  
747 }  
748 }  
749 }  
750 }  
751 }  
752 }  
753 }  
754 }  
755 }  
756 }  
757 }  
758 }  
759 }  
760 }  
761 }  
762 }  
763 }  
764 }  
765 }  
766 }  
767 }  
768 }  
769 }  
770 }  
771 }  
772 }  
773 }  
774 }  
775 }  
776 }  
777 }  
778 }  
779 }  
780 }  
781 }  
782 }  
783 }  
784 }  
785 }  
786 }  
787 }  
788 }  
789 }  
790 }  
791 }  
792 }  
793 }  
794 }  
795 }  
796 }  
797 }  
798 }  
799 }  
800 }  
801 }  
802 }  
803 }  
804 }  
805 }  
806 }  
807 }  
808 }  
809 }  
810 }  
811 }  
812 }  
813 }  
814 }  
815 }  
816 }  
817 }  
818 }  
819 }  
820 }  
821 }  
822 }  
823 }  
824 }  
825 }  
826 }  
827 }  
828 }  
829 }  
830 }  
831 }  
832 }  
833 }  
834 }  
835 }  
836 }  
837 }  
838 }  
839 }  
840 }  
841 }  
842 }  
843 }  
844 }  
845 }  
846 }  
847 }  
848 }  
849 }  
850 }  
851 }  
852 }  
853 }  
854 }  
855 }  
856 }  
857 }  
858 }  
859 }  
860 }  
861 }  
862 }  
863 }  
864 }  
865 }  
866 }  
867 }  
868 }  
869 }  
870 }  
871 }  
872 }  
873 }  
874 }  
875 }  
876 }  
877 }  
878 }  
879 }  
880 }  
881 }  
882 }  
883 }  
884 }  
885 }  
886 }  
887 }  
888 }  
889 }  
890 }  
891 }  
892 }  
893 }  
894 }  
895 }  
896 }  
897 }  
898 }  
899 }  
900 }  
901 }  
902 }  
903 }  
904 }  
905 }  
906 }  
907 }  
908 }  
909 }  
910 }  
911 }  
912 }  
913 }  
914 }  
915 }  
916 }  
917 }  
918 }  
919 }  
920 }  
921 }  
922 }  
923 }  
924 }  
925 }  
926 }  
927 }  
928 }  
929 }  
930 }  
931 }  
932 }  
933 }  
934 }  
935 }  
936 }  
937 }  
938 }  
939 }  
940 }  
941 }  
942 }  
943 }  
944 }  
945 }  
946 }  
947 }  
948 }  
949 }  
950 }  
951 }  
952 }  
953 }  
954 }  
955 }  
956 }  
957 }  
958 }  
959 }  
960 }  
961 }  
962 }  
963 }  
964 }  
965 }  
966 }  
967 }  
968 }  
969 }  
970 }  
971 }  
972 }  
973 }  
974 }  
975 }  
976 }  
977 }  
978 }  
979 }  
980 }  
981 }  
982 }  
983 }  
984 }  
985 }  
986 }  
987 }  
988 }  
989 }  
990 }  
991 }  
992 }  
993 }  
994 }  
995 }  
996 }  
997 }  
998 }  
999 }  
1000 }
```

Collection Algorithm

Sorting

13. Write a program that sorts an array of integers using the `sort()` method. Also try sorting in reverse order.



```
Workshop > J Main.java > Main > printArray(int[])
482 import java.util.Arrays;
483 import java.util.Comparator;
484
485 public class Main {
486     Run | Debug
487     public static void main(String[] args) {
488         int[] numbers = {5, 2, 9, 1, 7};
489
490         Arrays.sort(numbers);
491         printArray(numbers);
492
493         Integer[] integerNumbers = Arrays.stream(numbers).boxed().toArray(Integer[]::new);
494         Arrays.sort(integerNumbers, Comparator.reverseOrder());
495         for (int i = 0; i < numbers.length; i++) {
496             numbers[i] = integerNumbers[i];
497         }
498         printArray(numbers);
499     }
500
501     public static void printArray(int[] arr) {
502         for (int num : arr) {
503             System.out.print(num + " ");
504         }
505         System.out.println();
506     }
507 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop" ; if ($?) { javac Main.java } ; if ($?) { java Main }
1 2 5 7 9
9 7 5 2 1
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

14. Write a program that sorts an array list of string of colors using the `sort()` method. Also try sorting in reverse order.

```

Run | Debug
526 public static void main(String[] args) {
527     ArrayList<String> colors = new ArrayList<>();
528     colors.add(e:"Red");
529     colors.add(e:"Blue");
530     colors.add(e:"Green");
531     colors.add(e:"Yellow");
532     colors.add(e:"Orange");
533
534     Collections.sort(colors);
535     System.out.println(x:"Sorted in ascending order:");
536     printList(colors);
537
538     Collections.sort(colors, Collections.reverseOrder());
539     System.out.println(x:"\nSorted in descending order:");
540     printList(colors);
541 }
542
543 public static void printList(ArrayList<String> list) {
544     for (String color : list) {
545         System.out.print(color + " ");
546     }
}

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Sorted in descending order:
Yellow Red Orange Green Blue
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Sorted in ascending order:
Blue Green Orange Red Yellow

```

Binary search

15. Write a program to initialize an ArrayList with a set of integers. Implement a binary search algorithm to find a particular integer.

```

550 // 15. Write a program to initialize an ArrayList with a set of integers. Implement a binary search algorithm to find a particular integer.
560 import java.util.ArrayList;
561 import java.util.Collections;
562
563 public class Main {
564     public static void main(String[] args) {
565         ArrayList<Integer> numbers = new ArrayList<>();
566         Collections.addAll(numbers, 2, 5, 8, 12, 16, 23, 38, 56, 72, 91);
567
568         int target = 23;
569         int index = binarySearch(numbers, target);
570
571         if (index != -1) {
572             System.out.println(target + " found at index " + index);
573         } else {
574             System.out.println(target + " not found in the list");
575         }
576     }
577
578     public static int binarySearch(ArrayList<Integer> arr, int target) {
579         int left = 0, right = arr.size() - 1;
580
581         while (left <= right) {
582             int mid = left + (right - left) / 2;
583             if (arr.get(mid) == target) return mid;
584             if (arr.get(mid) < target) left = mid + 1;
585             else right = mid - 1;
586         }
587         return -1;
588     }
589 }
590
591

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

s:\OOPS\Week-9\Workshop> if ($?) { javac Main.java } ; if ($?) { java Main }
23 found at index 5
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>

```