

# CORE JAVA

## COLLECTION FRAMEWORK ASSIGNMENT

1. Given a `TreeMap<Long,Contact>` which has phone numbers for keys and contact objects for values.

Write solutions to

- a. Fetch all the keys and print them
- b. Fetch all the values and print them
- c. Print all key- value pairs

Note:

- a) Contacts should be stored in descending order of phone number
- b) Contact Class:
  - `PhoneNumber:<long>`
  - `Name:<String>`
  - `Email:<String>`
  - `Gender:<Enum>`

Code: -

```
ContactTree.java × HashSetExample.java EmployeeObject.java CheckLeapYear.java
1 package collection;
2 import java.util.TreeMap;
3 class Contact
4 {
5     long PhoneNumber;
6     String Name;
7     String Email;
8     String Gender;
9     public Contact(long phoneNumber, String name, String email, String gender)
10 {
11     super();
12     PhoneNumber = phoneNumber;
13     Name = name;
14     Email = email;
15     Gender = gender;
16 }
17 @Override
18 public String toString()
19 {
20     return "Contact [PhoneNumber=" + PhoneNumber + ", Name=" + Name + ", Email=" + Email + ", Gender=" + Gender + "]";
21 }
22 }
```

Problems Javadoc Declaration Console × Git Staging

<terminated> ContactTree [Java Application] C:\Users\MBALKRIS\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.0.v20211012-105

Fetching all the keys  
877991234  
8108764545  
8655454545

Fetching all the Values  
[Number=8876543901, Name=Siddhi, Email=sid@gmail.com, Gender=Female]  
[Number=9930697398, Name=Mrunal, Email=mru@gmail.com, Gender=Female]  
[Number=9967235410, Name=Kavish, Email=kav@gmail.com, Gender=Male]

Printing all the Key-Values pairs:{877991234=[Number=8876543901, Name=Siddhi, Email=sid@gmail.com, Gender=Female]  
8108764545=[Number=9930697398, Name=Mrunal, Email=mru@gmail.com, Gender=Female]  
8655454545=[Number=9967235410, Name=Kavish, Email=kav@gmail.com, Gender=Male]  
}

```

ContactTree.java x HashSetExample.java EmployeeObject.java CheckLeapYear.java
15 }
16 }
17 @Override
18 public String toString()
19 {
20     return "[Number=" + PhoneNumber + ", Name=" + Name + ", Email=" + Email + ", Gender=" + Gender + "]" + "\n";
21 }
22 }
23 public class ContactTree
24 {
25     public static void main(String[] args)
26     {
27         Contact obj1 = new Contact( 9930697398L, "Mrunal", "mru@gmail.com", "Female");
28         Contact obj2 = new Contact( 8876543901L, "Siddhi", "sid@gmail.com", "Female");
29         Contact obj3 = new Contact( 9967235410L, "Kavish", "kav@gmail.com", "Male");
30         TreeMap < Long , Contact> tr = new TreeMap<Long , Contact>();
31         tr.put(8108764545L, obj1);
32         tr.put(877991234L, obj2);
33         tr.put(8655454545L, obj3);
34         System.out.println("Fetching all the keys");
35         for(Long intk : tr.keySet())
36         {
37             System.out.println(intk);
38         }
39         System.out.println("Fetching all the Values");
40         for (Contact strV : tr.values())
41         {
42             System.out.println(strV);
43         }
44         System.out.println("Printing all the Key-Values pairs:"+ tr);
45     }
46 }
47

```

- Write an application to store 10 unique product objects. In case there is an attempt to add a duplicate product, it should be silently rejected. Hint: Use HashSet or TreeSet  
Extra(optional): Use ArrayList in the above solution. (This is optional)

Code: -

```

ContactTree.java HashSetExample.java x EmployeeObject.java Check
1 package collection;
2 import java.util.HashSet;
3 public class HashSetExample
4 {
5     public static void main(String[] args)
6     {
7
8         HashSet<Integer> myhashset = new HashSet<>();
9         myhashset.add(11);
10        myhashset.add(21);
11        myhashset.add(3);
12        myhashset.add(4);
13        myhashset.add(50);
14        myhashset.add(6);
15        myhashset.add(7);
16        myhashset.add(87);
17        myhashset.add(9);
18        myhashset.add(10);
19        myhashset.add(10);
20        myhashset.add(11);
21        System.out.println(myhashset);
22
23    }
24 }
25

```

Problems Javadoc Declaration Console x Git Staging

<terminated> HashSetExample [Java Application] C:\Users\MBALKRIS\p2\pool\plugir  
[50, 3, 4, 21, 6, 7, 87, 9, 10, 11]

- Store at least 10 Employee Objects in an TreeSet<Employee>. When the application runs the user should be asked to select one of the options upon which you will print the employee details in a sorted manner.

For e.g.

Run Application:

- ID
- Name
- Department
- Salary

Your choice: b

<Should print all the employees details sorted by name>

Code: -

```

ContactTree.java HashSetExample.java *EmployeeObject.java x CheckLeapYear.java
1 package collection;
2 import java.util.*;
3 public class EmployeeObject
4 {
5     public static void main(String[] args)
6     {
7         Employee emp_1 = new Employee(1, "Mrunal", "IT", 50000);
8         Employee emp_2 = new Employee(2, "Siddhi", "HR", 40000);
9         Employee emp_3 = new Employee(3, "Manasi", "IT", 300000);
10        Employee emp_4 = new Employee(4, "Vinod", "Admin", 50000);
11        Employee emp_5 = new Employee(5, "Vinayak", "IT", 60000);
12        Employee emp_6 = new Employee(6, "Prathamesh", "Event", 40000);
13        Employee emp_7 = new Employee(7, "Komal", "Event", 37000);
14        Employee emp_8 = new Employee(8, "Yash", "HR", 75000);
15        Employee emp_9 = new Employee(9, "Pooja", "IT", 24000);
16        Employee emp_10 = new Employee(10, "Jui", "Admin", 33000);
17
18        System.out.println("1.Enter a to sort according to id: ");
19        System.out.println("2.Enter b to sort according to Name: ");
20        System.out.println("3.Enter c to sort according to department: ");
21        System.out.println("4.Enter d to sort according to Salary:\n");
22        Scanner sc = new Scanner(System.in);
23        String ch = sc.nextLine();
24        TreeSet<Employee> set = new TreeSet<Employee>(new CustomSort(ch));
25
26        set.add(emp_1);
27        set.add(emp_2);
28        set.add(emp_3);
29        set.add(emp_4);
30        set.add(emp_5);
31        set.add(emp_6);
32        set.add(emp_7);
33        set.add(emp_8);
34        set.add(emp_9);
35        set.add(emp_10);
36
37        Iterator<Employee> it = set.iterator();
38        while(it.hasNext())
39        {
40            System.out.println(it.next());
41        }
42        sc.close();
43    }
44
45    class Employee
46    {
47        int id;
48        String name;
49        String dept;
50        long salary;
51        public Employee(int id, String name, String dept, long salary)
52        {
53            super();
54            this.id = id;
55            this.name = name;
56            this.dept = dept;
57            this.salary = salary;
58        }
59        @Override
60        public String toString()
61        {
62            return "Employee [id=" + id + ", name=" + name + ", dept=" + dept + ", salary=" + salary + "]\n";
63        }
64    }
65
66    class CustomSort implements Comparator<Employee>
67    {
68        String a;
69        public CustomSort(String a)
70        {
71            super();
72            this.a = a;
73        }
74        @Override
75        public int compare(Employee o1, Employee o2)
76        {
77            if(a.equalsIgnoreCase("a"))
78            {
79                return o1.id-o2.id;
80            }
81            else if(a.equalsIgnoreCase("b"))
82            {
83                return o1.name.compareTo(o2.name);
84            }
85            else if(a.equalsIgnoreCase("c"))
86            {
87                return o1.dept.compareTo(o2.dept);
88            }
89            else if(a.equalsIgnoreCase("d"))
90            {
91                if(o1.salary>o2.salary)
92                {
93                    return 1;
94                }
95                else if(o1.salary<o2.salary)
96                {
97                    return -1;
98                }
99                else
100                {
101                    return 0;
102                }
103            }
104            return 0;
105        }
106    }
107
108    }
109
110    }
111
112    }
113
114    }
115
116    }
117
118    }
119
120    }
121
122    }
123
124    }
125
126    }
127
128    }
129
130    }
131
132    }
133
134    }
135
136    }
137
138    }
139
140    }
141
142    }
143
144    }
145
146    }
147
148    }
149
150    }
151
152    }
153
154    }
155
156    }
157
158    }
159
160    }
161
162    }
163
164    }
165
166    }
167
168    }
169
170    }
171
172    }
173
174    }
175
176    }
177
178    }
179
180    }
181
182    }
183
184    }
185
186    }
187
188    }
189
190    }
191
192    }
193
194    }
195
196    }
197
198    }
199
200    }
201
202    }
203
204    }
205
206    }
207
208    }
209
210    }
211
212    }
213
214    }
215
216    }
217
218    }
219
220    }
221
222    }
223
224    }
225
226    }
227
228    }
229
230    }
231
232    }
233
234    }
235
236    }
237
238    }
239
240    }
241
242    }
243
244    }
245
246    }
247
248    }
249
250    }
251
252    }
253
254    }
255
256    }
257
258    }
259
260    }
261
262    }
263
264    }
265
266    }
267
268    }
269
270    }
271
272    }
273
274    }
275
276    }
277
278    }
279
280    }
281
282    }
283
284    }
285
286    }
287
288    }
289
290    }
291
292    }
293
294    }
295
296    }
297
298    }
299
300    }
301
302    }
303
304    }
305
306    }
307
308    }
309
310    }
311
312    }
313
314    }
315
316    }
317
318    }
319
320    }
321
322    }
323
324    }
325
326    }
327
328    }
329
330    }
331
332    }
333
334    }
335
336    }
337
338    }
339
340    }
341
342    }
343
344    }
345
346    }
347
348    }
349
350    }
351
352    }
353
354    }
355
356    }
357
358    }
359
360    }
361
362    }
363
364    }
365
366    }
367
368    }
369
370    }
371
372    }
373
374    }
375
376    }
377
378    }
379
380    }
381
382    }
383
384    }
385
386    }
387
388    }
389
390    }
391
392    }
393
394    }
395
396    }
397
398    }
399
400    }
401
402    }
403
404    }
405
406    }
407
408    }
409
410    }
411
412    }
413
414    }
415
416    }
417
418    }
419
420    }
421
422    }
423
424    }
425
426    }
427
428    }
429
430    }
431
432    }
433
434    }
435
436    }
437
438    }
439
440    }
441
442    }
443
444    }
445
446    }
447
448    }
449
450    }
451
452    }
453
454    }
455
456    }
457
458    }
459
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    }

```

- Given a LinkedList of Objects representing date of birth's (use any inbuild java class to represent date), print the date's along with the message: Your date of Birth is DD-MM-YYYY, and it (was or was not) a leap year.

E.g.

- For the date 23-12-2000

Your date of Birth is 23-12-2000 and it was a leap year

b) For the date 23-12-2001

Your date of Birth is 23-12-2001 and it was not a leap year

Note: - You need to access the Dates in the reverse order. i.e. start from the last object and move towards the first object.

Code: -

```
1 package collection;
2 import java.time.LocalDate;
3 import java.time.format.DateTimeFormatter;
4 import java.util.LinkedList;
5 import java.util.List;
6
7 public class CheckLeapYear
8 {
9     public static void main(String[] args)
10    {
11        Date date = new Date("22/08/1998");
12        Date date1 = new Date("12/10/1995");
13        Date date2 = new Date("13/04/2017");
14        Date date3 = new Date("15/09/2012");
15        Date date4 = new Date("12/10/2016");
16        Date date5 = new Date("26/08/2004");
17        Date date6 = new Date("10/12/2005");
18
19        List<Date> dobList = new LinkedList<>();
20        dobList.add(date);
21        dobList.add(date1);
22        dobList.add(date2);
23        dobList.add(date3);
24        dobList.add(date4);
25        dobList.add(date5);
26        dobList.add(date6);
27
28        dobList.add(date6);
29
30        DateTimeFormatter df = DateTimeFormatter.ofPattern("dd/MM/yyyy");
31
32        for(int i =0;i<dobList.size();i++)
33        {
34            LocalDate ld =LocalDate.parse(dobList.get(i).date,df);
35            String sd = (ld).format(df);
36
37            if (ld.getYear()%4 ==0)
38            {
39                System.out.println(sd + " "+"This is a leap year");
40            }
41            else
42            {
43                System.out.println(sd+" "+"This is not a leap year");
44            }
45        }
46    }
47
48    class Date
49    {
50        String date;
51        public Date(String date)
52        {
53            super();
54            this.date = date;
55        }
56        @Override
57        public String toString()
58        {
59            return " [date=" + date + " ]";
60        }
61        public String getDate()
62        {
63            return date;
64        }
65    }
66 }
```

```
46 class Date
47 {
48     String date;
49     public Date(String date)
50     {
51         super();
52         this.date = date;
53     }
54     @Override
55     public String toString()
56     {
57         return " [date=" + date + " ]";
58     }
59     public String getDate()
60     {
61         return date;
62     }
63 }
64
```