Systems Programming and Concurrency
Course 22/23
Home task on GUI with Java

System description

We want to compare the time taken to sort lists of N numbers (stored in arrays of type a[0..N-1]) using the sorting algorithm by selection and the bubble algorithm, which are described below:

Selection sort

```
for (int i=0;i<N-1;i++){
      // We look for the smallest element of the array a[i.. N-1]
      // and we swap it with a[i]
      int lowest = i
      for (int j=i+1; j<N; j++)
            if (a[j]<a[lowest]) lowest = j;
      int aux = a[i];
      a[i] = a[lowest];
      a[lowest] = aux;
      // Inv: a[0..i] contains the i+1 lowest elements of a[0..N-1]
      //Inv: a[0..i] is ordered
}
```

Bubble sort

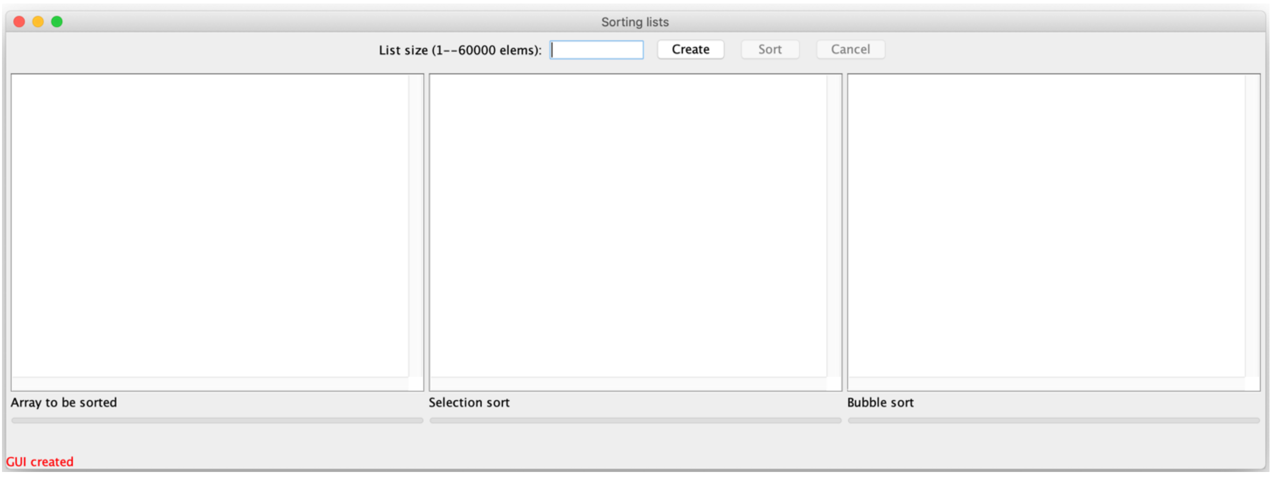
```
for (int i=0;i<N-1;i++){
// we run the array a[i.. N-1] from position N-1 to i.
// We compare the elements two by two and exchange them if
// are unsorted
      for (int j=N-1; j>i; j--) {
            if (a[j]<a[j-1]){
                  int aux = a[j];
                  a[j] = a[j-1];
                  a[j-1] = aux;
            }
      }
      // Inv: a[0..i] contains the i+1 lowest elements of a[0..N-1]
      // Inv: a[0..i] is ordered
}
```

This algorithm can be optimized following several approaches. One of them is to stop iterating the outermost loop when it detects that no pair of items has been swapped in an iteration (meaning that the list is sorted).

Note that in the two algorithms described, in each iteration of the outermost loop the sublist a[0..i] is already ordered.

You have to develop a GUI that, given a list of numbers, generated randomly, shows it ordered using the two algorithms described above. In the Virtual Campus you have the template of the **Panel** class that will help you start the project. You have to make three versions of this project and deliver each of them in a compressed file.

The provided **Panel** class allows you creating an user interface with the layout shown in the figure:



The main elements of the interface are:

1. A text field (JTextField size) in which the user enters the size N of the list to be sorted.
2. A button (JButton createButton) with the text "Create" that, once pressed, randomly generates a list of N numbers between 0 and 99999 and displays the list in the leftmost area.
3. A text area (JTextArea area) where the random list generated after pressing the "Create" button is displayed.
4. A button (JButton sortButton) with the text "Sort" that, once pressed, sorts the list created using the two algorithms (selection and bubble).
5. A button (JButton cancelButton) with the text "Cancel" that, once pressed, allows you to cancel the sorting of the list.
6. Two text areas (JTextArea areaSelection, areaBubble) in which the lists sorted using each of the algorithms are displayed.
7. Three labels (JLabel messageArea, messageSelection, messageBubble) below the text areas that are used to display the status of tasks being performed.
8. Two progress bars (JProgressBar progressSelection, progressBubble) below the respective text areas that are used in the second part of the exercise to show the progress of each sort algorithm.
9. A red label (JLabel comment) at the bottom of the GUI that allows you to give information to the user.

Exercise 1 (0.5 points): GUI with SwingWorker with done():

1. Include the **Panel** class in your project, which will look alike the one shown in the previous figure.
2. Create a **Controller** class that implements the interfaces needed to handle the events of the "Create" and "Sort" buttons.
3. When the user presses the "Create" button, a random list of numbers of the size indicated in the size text field must be created. The list should be displayed in the text area on the left (called area), and the message label should be updated with the message "List created". If the size

entered by the text field is not in the range [1..60000] the number must be deleted and the user informed, through the comment tag, that the number entered is invalid. When you create a new list, the areaSelection and areaBubble text areas must be cleaned up.

4. Creates a **WorkerSelection** class that sorts the previous list using the selection sorting algorithm. The list to be sorted is passed as a parameter in the constructor and, in the constructor, **a copy of the list must be created** (**Note**: the input list should not be sorted).
5. Create a **WorkerBubble** class that gives a list sorts the list using the bubble algorithm. The list to be sorted is passed as a parameter in the constructor and, in the constructor, **a copy of the list must be created** (**Note**: the input list should not be sorted).
6. When the user presses the "Sort" button, the list created must be sorted by the two workers and then the result must be displayed in the corresponding text areas. While each worker is sorting the list, the message "Sorting the list" should appear below each text area. When a worker has finished, a message of the type "List sorted in xxx ms." should appear below its text area. To calculate the time taken to perform sorting, you can use instructions such as:

```
long startTime = System.currentTimeMillis(); //Beginning of sorting
.........
total= (System.currentTimeMillis()-startTime); // end of sorting
```

7. Create a **Main** class that constructs a dispatcher thread from which the GUI is created.

Note: In this first version it is not necessary to control the progress bar, nor to make use of the "Cancel" button

Exercise 2 (0.3 points): GUI with SwingWorker with publish and progress bar.

This version of the GUI uses the fact that, after each iteration of the outermost loop of each of the algorithms, the sublist a[0..i] is already sorted:

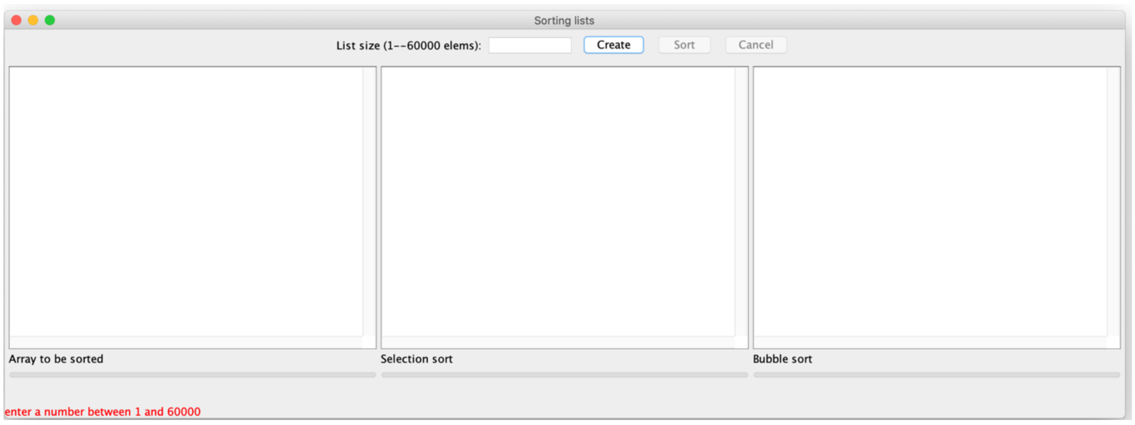
1. Modify the **WorkerSelection** and **WorkerBubble** classes so that the ordered sublists are displayed in the respective text areas as the outermost loops of the algorithms are iterated. In addition, workers have to update their progress so that each progress bar shows the percentage of work done by each worker.
2. Modifies the **Controller** class to implement the interfaces necessary to handle worker progress change events.

Exercise 3 (0.2 points): Make the necessary modifications to the **Panel**, **Controller**, **WorkerSelection**, and **WorkerBubble** class so that:

1. It is possible to cancel the sorting executions of the two workers when the user presses the "Cancel" button.
2. Only the "Create" button is enabled if a list has not been created.
3. The "Sort" button is only enabled if a list has been created but not yet sorted.
4. Only the "Cancel" button is enabled while the lists are being sorted.
5. After sorting the list, or if sorting has been cancelled, only the "Create" button should be enabled.

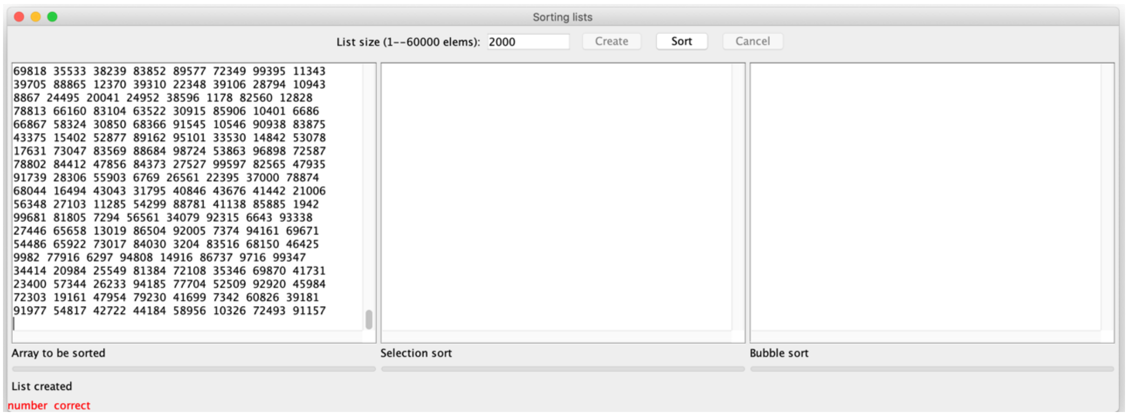
Examples of the view in the first part of the project:

1.- GUI after pressing the "Create" button without entering any number



```
enter a number between 1 and 60000
```

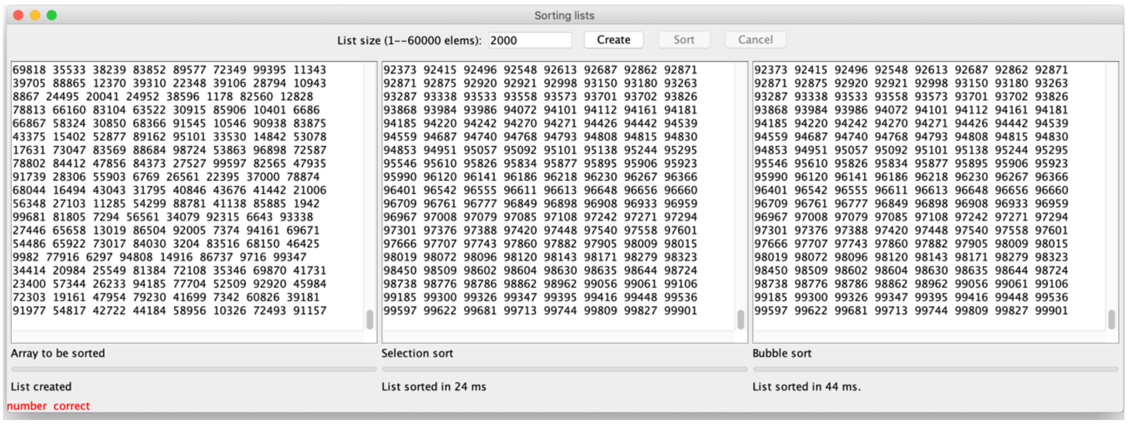
2.- GUI after pressing "Create" and entering the value 2000



```
69818 35533 38239 83852 89577 72349 99395 11343
39705 88865 12370 39310 22348 39106 28794 10943
8867 24495 20041 24952 38596 1178 82560 12828
78813 66160 83104 63522 30915 85906 10401 6686
66867 58324 30850 68366 91545 10546 90938 83875
43375 15402 52877 89162 95101 33530 14842 53078
17631 73047 83569 88684 98724 53863 96898 72587
78802 84412 47856 84373 27527 99597 82565 47935
91739 28306 55903 6769 26561 22395 37000 78874
68044 16494 43043 31795 40846 43676 41442 21006
56348 27103 11285 54299 88781 41138 85885 1942
99681 81805 7294 56561 34079 92315 6643 93338
27446 65658 13019 86504 92005 7374 94161 69671
54486 65922 73017 84030 3204 83516 68150 46425
9982 77916 6297 94808 14916 86737 9716 99347
34414 20984 25549 81384 72108 35346 69870 41731
23400 57344 26233 94185 77704 52509 92920 45984
72303 19161 47954 79230 41699 7342 60826 39181
91977 54817 42722 44184 58956 10326 72493 91157
```

```
List created
number  correct
```

3.- GUI after pressing "Sort"



Array to be sorted:

```
69818 35533 38239 83852 89577 72349 99395 11343
39705 88865 12370 39310 22348 39106 28794 10943
8867 24495 20041 24952 38596 1178 82560 12828
78813 66160 83104 63522 30915 85906 10401 6686
66867 58324 30850 68366 91545 10546 90938 83875
43375 15402 52877 89162 95101 33530 14842 53078
17631 73047 83569 88684 98724 53863 96898 72587
78802 84412 47856 84373 27527 99597 82565 47935
91739 28306 55903 6769 26561 22395 37000 78874
68044 16494 43043 31795 40846 43676 41442 21006
56348 27103 11285 54299 88781 41138 85885 1942
99681 81805 7294 56561 34079 92315 6643 93338
27446 65658 13019 86504 92005 7374 94161 69671
54486 65922 73017 84030 3204 83516 68150 46425
9982 77916 6297 94808 14916 86737 9716 99347
34414 20984 25549 81384 72108 35346 69870 41731
23400 57344 26233 94185 77704 52509 92920 45984
72303 19161 47954 79230 41699 7342 60826 39181
91977 54817 42722 44184 58956 10326 72493 91157
```

Selection sort:

```
92373 92415 92496 92548 92613 92687 92862 92871
92871 92875 92920 92921 92998 93150 93180 93263
93287 93338 93533 93558 93573 93701 93702 93826
93868 93984 93986 94072 94101 94112 94161 94181
94185 94220 94242 94270 94271 94426 94442 94539
94559 94687 94740 94768 94793 94808 94815 94830
94853 94951 95057 95092 95101 95138 95244 95295
95546 95610 95826 95834 95877 95895 95906 95923
95990 96120 96141 96186 96218 96230 96267 96366
96401 96542 96555 96611 96613 96648 96656 96660
96709 96761 96777 96849 96898 96908 96933 96959
96967 97008 97079 97085 97108 97242 97271 97294
97301 97376 97388 97420 97448 97540 97558 97601
97666 97707 97743 97860 97882 97905 98009 98015
98019 98072 98096 98120 98143 98171 98279 98323
98450 98509 98602 98604 98630 98635 98644 98724
98738 98776 98786 98862 98962 99056 99061 99106
99185 99300 99326 99347 99395 99416 99448 99536
99597 99622 99681 99713 99744 99809 99827 99901
```

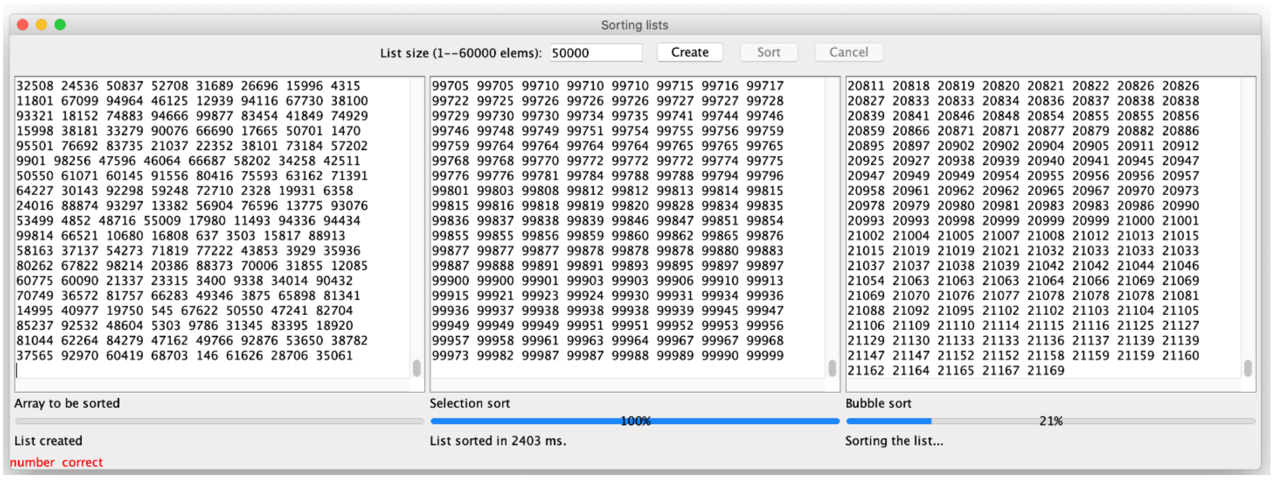
Bubble sort:

```
92373 92415 92496 92548 92613 92687 92862 92871
92871 92875 92920 92921 92998 93150 93180 93263
93287 93338 93533 93558 93573 93701 93702 93826
93868 93984 93986 94072 94101 94112 94161 94181
94185 94220 94242 94270 94271 94426 94442 94539
94559 94687 94740 94768 94793 94808 94815 94830
94853 94951 95057 95092 95101 95138 95244 95295
95546 95610 95826 95834 95877 95895 95906 95923
95990 96120 96141 96186 96218 96230 96267 96366
96401 96542 96555 96611 96613 96648 96656 96660
96709 96761 96777 96849 96898 96908 96933 96959
96967 97008 97079 97085 97108 97242 97271 97294
97301 97376 97388 97420 97448 97540 97558 97601
97666 97707 97743 97860 97882 97905 98009 98015
98019 98072 98096 98120 98143 98171 98279 98323
98450 98509 98602 98604 98630 98635 98644 98724
98738 98776 98786 98862 98962 99056 99061 99106
99185 99300 99326 99347 99395 99416 99448 99536
99597 99622 99681 99713 99744 99809 99827 99901
```

```
List created                List sorted in 24 ms                List sorted in 44 ms.
number  correct
```

Example of the view in the second part of the project:

1.- GUI during list sorting



List size (1--60000 elems): 50000

```
32508 24536 50837 52708 31689 26696 15996 4315
11801 67099 94964 46125 12939 94116 67730 38100
93321 18152 74883 94666 99877 83454 41849 74929
15998 38181 33279 90076 66690 17665 50701 1470
95501 76692 83735 21037 22352 38101 73184 57202
9901 98256 47596 46064 66687 58202 34258 42511
50550 61071 60145 91556 80416 75593 63162 71391
64227 30143 92298 59248 72710 2328 19931 6358
24016 88874 93297 13382 56904 76596 13775 93076
53499 4852 48716 55009 17980 11493 94336 94434
99814 66521 10680 16808 637 3503 15817 88913
58163 37137 54273 71819 77222 43853 3929 35936
80262 67822 98214 20386 88373 70006 31855 12085
60775 60090 21337 23315 3400 9338 34014 90432
70749 36572 81757 66283 49346 3875 65898 81341
14995 40977 19750 545 67622 50550 47241 82704
85237 92532 48604 5303 9786 31345 83395 18920
81044 62264 84279 47162 49766 92876 53650 38782
37565 92970 60419 68703 146 61626 28706 35061
```

Array to be sorted

List created

number correct

```
99705 99705 99710 99710 99710 99715 99716 99717
99722 99725 99726 99726 99726 99727 99727 99728
99729 99730 99730 99734 99735 99741 99744 99746
99746 99748 99749 99751 99754 99755 99756 99759
99759 99764 99764 99764 99764 99765 99765 99765
99768 99768 99770 99772 99772 99772 99774 99775
99776 99776 99781 99784 99788 99788 99794 99796
99801 99803 99808 99812 99812 99813 99814 99815
99815 99816 99818 99819 99820 99828 99834 99835
99836 99837 99838 99839 99846 99847 99851 99854
99855 99855 99856 99859 99860 99862 99865 99876
99877 99877 99877 99878 99878 99878 99880 99883
99887 99888 99891 99891 99893 99895 99897 99897
99900 99900 99901 99903 99903 99906 99910 99913
99915 99921 99923 99924 99930 99931 99934 99936
99936 99937 99938 99938 99938 99939 99945 99947
99949 99949 99949 99951 99951 99952 99953 99956
99957 99958 99961 99963 99964 99967 99967 99968
99973 99982 99987 99987 99988 99989 99990 99999
```

Selection sort

100%

List sorted in 2403 ms.

```
20811 20818 20819 20820 20821 20822 20826 20826
20827 20833 20833 20834 20836 20837 20838 20838
20839 20841 20846 20848 20854 20855 20855 20856
20859 20866 20871 20871 20877 20879 20882 20886
20895 20897 20902 20902 20904 20905 20911 20912
20925 20927 20938 20939 20940 20941 20945 20947
20947 20949 20949 20954 20955 20956 20956 20957
20958 20961 20962 20962 20965 20967 20970 20973
20978 20979 20980 20981 20983 20983 20986 20990
20993 20993 20998 20999 20999 20999 21000 21001
21002 21004 21005 21007 21008 21012 21013 21015
21015 21019 21019 21021 21032 21033 21033 21033
21037 21037 21038 21039 21042 21042 21044 21046
21054 21063 21063 21063 21064 21066 21069 21069
21069 21070 21076 21077 21078 21078 21078 21081
21088 21092 21095 21102 21102 21103 21104 21105
21106 21109 21110 21114 21115 21116 21125 21127
21129 21130 21133 21133 21136 21137 21139 21139
21147 21147 21152 21152 21158 21159 21159 21160
21162 21164 21165 21167 21169
```

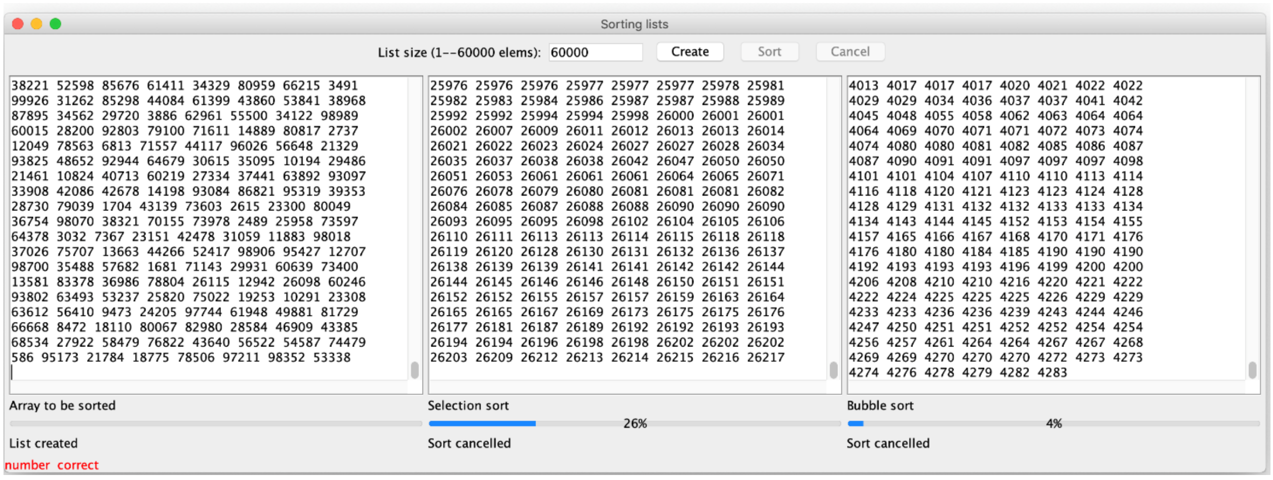
Bubble sort

21%

Sorting the list...

Example of the view in the third part of the project:

1. GUI after you have cancelled the sorting process.



List size (1--60000 elems): 60000

```
38221 52598 85676 61411 34329 80959 66215 3491
99926 31262 85298 44084 61399 43860 53841 38968
87895 34562 29720 3886 62961 55500 34122 98989
60015 28200 92803 79100 71611 14889 80817 2737
12049 78563 6813 71557 44117 96026 56648 21329
93825 48652 92944 64679 30615 35095 10194 29486
21461 10824 40713 60219 27334 37441 63892 93097
33908 42086 42678 14198 93084 86821 95319 39353
28730 79039 1704 43139 73603 2615 23300 80049
36754 98070 38321 70155 73978 2489 25958 73597
64378 3032 7367 23151 42478 31059 11883 98018
37026 75707 13663 44266 52417 98906 95427 12707
98700 35488 57682 1681 71143 29931 60639 73400
13581 83378 36986 78804 26115 12942 26098 60246
93802 63493 53237 25820 75022 19253 10291 23308
63612 56410 9473 24205 97744 61948 49881 81729
66668 8472 18110 80067 82980 28584 46909 43385
68534 27922 58479 76822 43640 56522 54587 74479
586 95173 21784 18775 78506 97211 98352 53338
```

Array to be sorted

List created

number correct

```
25976 25976 25976 25977 25977 25977 25978 25981
25982 25983 25984 25986 25987 25987 25988 25989
25992 25992 25994 25994 25998 26000 26001 26001
26002 26007 26009 26011 26012 26013 26013 26014
26021 26022 26023 26024 26027 26027 26028 26034
26035 26037 26038 26038 26042 26047 26050 26050
26051 26053 26061 26061 26061 26064 26065 26071
26076 26078 26079 26080 26081 26081 26081 26082
26084 26085 26087 26088 26088 26090 26090 26090
26093 26095 26095 26098 26102 26104 26105 26106
26110 26111 26113 26113 26114 26115 26118 26118
26119 26120 26128 26130 26131 26132 26136 26137
26138 26139 26139 26141 26141 26142 26142 26144
26144 26145 26146 26146 26148 26150 26151 26151
26152 26152 26155 26157 26157 26159 26163 26164
26165 26165 26167 26169 26173 26175 26175 26176
26177 26181 26187 26189 26192 26192 26193 26193
26194 26194 26196 26198 26198 26202 26202 26202
26203 26209 26212 26213 26214 26215 26216 26217
```

Selection sort

26%

Sort cancelled

```
4013 4017 4017 4017 4020 4021 4022 4022
4029 4029 4034 4036 4037 4037 4041 4042
4045 4048 4055 4058 4062 4063 4064 4064
4064 4069 4070 4071 4071 4072 4073 4074
4074 4080 4080 4081 4082 4085 4086 4087
4087 4090 4091 4091 4097 4097 4097 4098
4101 4101 4104 4107 4110 4110 4113 4114
4116 4118 4120 4121 4123 4123 4124 4128
4128 4129 4131 4132 4132 4133 4133 4134
4134 4143 4144 4145 4152 4153 4154 4155
4157 4165 4166 4167 4168 4170 4171 4176
4176 4180 4180 4184 4185 4190 4190 4190
4192 4193 4193 4193 4196 4199 4200 4200
4206 4208 4210 4210 4216 4220 4221 4222
4222 4224 4225 4225 4225 4226 4229 4229
4233 4233 4236 4236 4239 4243 4244 4246
4247 4250 4251 4251 4252 4252 4254 4254
4256 4257 4261 4264 4264 4267 4267 4268
4269 4269 4270 4270 4270 4272 4273 4273
4274 4276 4278 4279 4282 4283
```

Bubble sort

4%

Sort cancelled