CS351 DATA ORGANIZATION AND MANAGEMENT

Programming Assignment #3

Replacement Selection Sort in JAVA

Due Date: December 24, 2009; 23:59

Assignment Definition

In this assignment, you are going to implement replacement selection sort using "Array" class in Java. You are going to sort the given records in increasing order. In the web, there are other implementation types of replacement selection sort using priority queues or heaps, but you are only allowed to use arrays for registers in this assignment. You are going to implement a program called **"Replacement.java"**, in which you are going to sort the given records (numbers) to you in a text file called **"Records.txt"** by using replacement selection sort algorithm. In replacement selection sort, there are m registers that are kept in main memory, and these m registers are used for sorting the records. That's, you are going to create an array object and its length is going to be m. And size of the array is going to be dynamic (You are not allowed to use "**ArrayList"** for registers.). Pseudocode of replacement selection sort is:

Step1: The m registers are filled with records from the input to be sorted.
Step2: All registers are put into the "on" state.
Step3: Select the register which has the smallest of all "on" registers.
Step4: Transfer the contents of the selected register to the output (call it as key Y).
Step5: Replace the contents of the selected register by the next input record.

If new record key > Y
    Go to step 3
If new record key = =Y
    Go to step 4
*If new record key < Y
    Go to step 6

Step6: Turn the selected key register "off".

If all registers are now "off"
    We have completed a sorted substring (run).
    We start a new substring group and go to step 2.
Else
    Go to step 3

Notice: The pseudocode is taken from the course's web site, but the line with * is corrected.

In this assignment, number of registers is going to be determined by user. So, first you are going to ask user to enter register number. Then, you are going to create an array object dynamically allocated and its length will be the number entered by user. Here is an example of ***dynamic array allocation***:

```
public class replacement
{
        private static int[] register;

        public static void main(String[] args)
        {
            //regNo is going to the number of registers entered by user
            register = new int[regNo];
        }
}
```

After creating an array dynamically, you are going to read first m records from "Records.txt". Then, you are going to follow steps given in the pseudocode. In this assignment, you are going to sort the records in increasing order (as it is in the example given in course's web page), thus, in each iteration of the program, you are going to choose the register which has the smallest record among all registers, and you are going to store that record. You can store the sorted records until now in an "**ArrayList**" or however you like. In the beginning of the program, all registers are turned on. But, when the program enters into step 6, you are going to turn of the register after replacing the smallest record among registers with the next record waiting in the list (as it is in the example given in course's web page). Turning a register off means that register is not going to be used for sorting anymore until turning it on again. You can implement turning on and off processes as you wish in this assignment. For example, you can use another dynamic array like the register array to store condition of the register such as turned off or turned on. When all registers were turned off, it means program completed a run successfully, and it is going to pass to next run by turning all registers on. At the end of the program, you are going to print the records sorted in each run on the screen.

In course's web site, there is an example of sorting numbers given by using replacement selection sort. In that example, register number is 3. Firstly, first 3 records (numbers) are read into 3 registers. Then, steps given in the pseudocode are followed. When program enters into step 6, the register is turned off. In the example, * symbol represents that register was turned off. When all registers are turned off (*0*1*6 in the example), it means program completed a run successfully, and we are passing to the next run by turning all registers on. In the example, - symbol represents that the register is empty, which means there are no records left to be sorted. For this example, at the end of the program, you are going to print the records sorted in each run on the screen such as:

Run 1: 2 5 7 8 9
Run 2: 0 1 3 4 6

Record File (Records.txt) Structure

The record file (Records.txt) given to you is going to be like the given example record file. If you look at "Records.txt" file, the records are separated with a coma (,) and all records are stored in a single line such as "5,2,9,7,0,8,1,6,3,4". In this file, each record is a **non-zero integer**, so there will be no negative or floating numbers in this file. Besides, this file won't be empty, that's, there will be at least one record in "Records.txt". You don't need to test your program for these criteria.

Implementation

You will have only one java class, **"Replacement.java"** which will make these operations in order:

- ask user to enter the number of registers
- create an array object dynamically allocated and its length will be the number entered by user
- read records from the text file "Records.txt"
- follow the steps given in the pseudocode
- print the records sorted in each run on the screen

Notifications

- There is a good example of this assignment in course's web page. The implementation of this assignment will be same with that given example.
- You don't need to check any validation rules such as during testing process, "Records.txt" file will be put in the same directory with your program.
- During testing your program, the register number entered by user won't be more than the number of records in Records.txt file, so at the beginning of the program, registers won't be empty. Moreover, the register number entered by user won't be less than 1 (zero or any negative number), which means 1 ≤ register number ≤ record number.
- For storage of sorted records, you are allowed to use ArrayList, but for registers, you are only allowed to use arrays dynamically allocated.
- For correctness of your program, you can use given example Records.txt file. If you enter 3 for register number as it is in the course's web site, you can check if your program works. But, this example won't be enough to see if your program is working completely. Testing your program with different register numbers and different record files will help you to see if your program has any exceptions.

Testing Details

A sample record file is posted to the website to help you to see the structure of record file and to test your program. This will not be the record file that will be used in grading your program. While the record file provided should be useful, you should also do testing on your own record file to ensure that your program works correctly. During your submission, you will only submit "Replacement.java" which will be implemented by you.

Submission Rules

You will submit only one file: **"Replacement.java"** java class. You will put only this java class into the winrar file will be *StudentID_Name_Surname.rar* (20491000_Emre_Celik.rar). (Don't use Turkish characters) Please don't send the project file of programs like Eclipse or Visual Studio. You will just put **"Replacement.java"** which is nested in "**src**" directory of these programs. You can upload the same file for several times before submission due date. Each time you upload a new file which has the same file name with your previously uploaded file, your previously uploaded file will be substituted with the recently uploaded file. The ones who do not obey the submission format will be penalized for 5 points. Late submissions will not be accepted.

Upload Page: http://www.cs.bilkent.edu.tr/~yenicag/