CS102 – Algorithms and Programming II
Programming Assignment 6
Spring 2025

ATTENTION:

- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention, where you replace the variables “Sec1”, “Surname” and “Name” with your actual section, surname and name:

  CS102_Sec1_Asgn6_Surname_Name.zip
- Upload the above zip file to Moodle by the deadline (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TAs and tutors during the lab. You have to make the preliminary submission before the lab day. After the TA checks your work in the lab, you will make your final submission. Even if your code does not change in between these two versions, you should make two submissions for each assignment.

GRADING WARNING:

- Please read the grading criteria provided on Moodle. The work must be done individually. Code sharing is strictly forbidden. We are using sophisticated tools to check the code similarities. The Honor Code specifies what you can and cannot do. Breaking the rules will result in disciplinary action.

Sorting Accounts

For this assignment, you will implement a console-based application that simulates a banking system with users and different types of accounts. Each user is stored with a 9-digit ID, name, and surname. Each user can have multiple accounts. Accounts store an amount of money in currency A, B, C, or D. Each of these currencies has a conversion rate stored in the bank. The conversion rate for each type of currency is randomly determined at the beginning of the application as a number between 0.2 and 5.0. Then, the user can set the conversion rate of each currency type.

Suppose an account has 10 credits in currency A. If the conversion rate of currency A is 0.5, then this account has 10 * 0.5 = 5 credits in common currency. In other words, multiplying the corresponding amount with the current conversion rate of the currency gives the amount in common currency. The accounts will store the amount in currency A, B, C, or D, but we will use the corresponding amount in common currency when finding the total amount a user has. To this end, you should include a getTotalAmountInCommonCurrency() method in your user class, which will find the sum of all the accounts of this user, converting each to the common currency. You may use an ArrayList to keep a user's accounts.

Include a method that generates the desired number of random users by selecting a name and surname from a list of names and surnames you prepare. For example, you can keep static arrays for names and surnames, each with at least 15 Strings in it; then, choose a random name and a

surname while creating a new user. The user should have an ID formed by concatenating 9 randomly selected digits (such as 892345900 or 002345738. This option should also create several accounts (for example, at least 2 and at most 10) for that user, each with a randomly determined type and amount in a range you determine. This method will help us have a set of users ready for the remaining operations. You can store the users in a user array of size 100000 and use a separate variable to keep track of the number of users inserted into this array.

You should have a sort option to order the users based on their IDs or total amount of credits in terms of the common currency. You will implement a hybrid sort that combines quick sort and insertion sort. Based on a user-determined integer *limit*, the hybrid sort uses the following algorithm to sort the portion of the array between indexes [from, to):

Sort (int from, int to):

- If the number of elements in between from and to is less than the limit:
    - Use quick sort partition to separate the elements less than the pivot and greater than the pivot. This will create two partitions [from, pivot) and [pivot+1, to). Use a recursive call to sort these two halves (the pivot is already at the correct position).
- Else (meaning the number of elements is less than the limit):
    - Use insertion sort to order this portion of the array.

In short, the algorithm creates the partitions using the quick sort analogy and uses insertion sort to sort the partitions that have less than limit elements in them. Since each partition is formed with the pivot in the correct position, when we sort the non-pivot elements in each partition, the whole array will be sorted. You may choose the pivot as an element in a specific index (such as the first or the last element of the current range) or use the median of the elements in the current partition. We would like to compare the average running time using different limit sizes. To this end, you may use the following code to measure time in milliseconds:

```
long start = System.currentTimeMillis();
// sort the array
long finish = System.currentTimeMillis();
long timeElapsed = finish - start;
```

For a sufficiently large number of users, sort the same arrangement of users using different limits and observe how running time changes. Sorting an already sorted array may result in a different performance. To this end, you may keep the original unsorted array as a copy and obtain a deep copy of this original array to sort your elements. Include an option to revert the user array to the original unsorted version so that you can use it to sort with different settings. Reverting the array means that you will copy the users in the original array to the current user array so that the order of the users is unsorted. After the sort is completed, print the elapsed time. Include an option to display the current user array, displaying each user with their ID, name, surname, and total amount of money in common currency. For example, after we sort the users, the displayed users will be in ascending order.

Also, include a method to display the account information of a user given the user's ID. This method should throw an exception if the given user ID does not correspond to any of the existing users. This exception should be handled with a try-catch block in the method that calls the method that throws the exception. If the user exists, this method should display the user's name, surname, and the total amount of credits in common currency, as well as each account of the user with its type and amount.

We should be able to modify the conversion rate of each currency. After changing the conversion rates, sorting the users based on the total amount may result in a different order. The following is a sample interaction with the program. Note that you do not need to have the same style of output as long as you support the desired functionality. Also, the sample includes a small number of users for the sake of simplicity, and therefore, the running time of the sorts is small. You are expected to test your implementation with a large number of users generated.

```
Welcome to the bank!
Current conversion rates:

A: 1.3473896
B: 1.245906
C: 2.0495324
D: 2.8282416

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 1
Generate how many?: 10

Generating 10 random users

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 2

112130784 Oliver Williams Total Amount: 2499.3008
698809670 Leo Evans Total Amount: 4738.786
439357777 Olivia Davies Total Amount: 4664.2188
245567968 Mateo Wright Total Amount: 942.602
```

```
562068378 Mia Wilson Total Amount: 3075.7256
160499504 Ezra Green Total Amount: 1419.5283
837843461 Levi Wilson Total Amount: 761.5398
509544648 Ellie Thompson Total Amount: 4286.8335
467208005 Elijah Taylor Total Amount: 1548.5774
007595279 Emma Wright Total Amount: 3270.618

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 5

Choose sort type:
1. By ID
2. By Total Amount
Option: 1

Enter sort limit: 3
Sort duration: 0

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 2

007595279 Emma Wright Total Amount: 3270.618
112130784 Oliver Williams Total Amount: 2499.3008
160499504 Ezra Green Total Amount: 1419.5283
245567968 Mateo Wright Total Amount: 942.602
439357777 Olivia Davies Total Amount: 4664.2188
467208005 Elijah Taylor Total Amount: 1548.5774
509544648 Ellie Thompson Total Amount: 4286.8335
562068378 Mia Wilson Total Amount: 3075.7256
698809670 Leo Evans Total Amount: 4738.786
837843461 Levi Wilson Total Amount: 761.5398

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
```

```
6. Reset to the original unsorted array
0. Exit
Option: 6

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 2

112130784 Oliver Williams Total Amount: 2499.3008
698809670 Leo Evans Total Amount: 4738.786
439357777 Olivia Davies Total Amount: 4664.2188
245567968 Mateo Wright Total Amount: 942.602
562068378 Mia Wilson Total Amount: 3075.7256
160499504 Ezra Green Total Amount: 1419.5283
837843461 Levi Wilson Total Amount: 761.5398
509544648 Ellie Thompson Total Amount: 4286.8335
467208005 Elijah Taylor Total Amount: 1548.5774
007595279 Emma Wright Total Amount: 3270.618

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 5

Choose sort type:
1. By ID
2. By Total Amount
Option: 2

Enter sort limit: 6
Sort duration: 0

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 2
```

```
837843461 Levi Wilson Total Amount: 761.5398
245567968 Mateo Wright Total Amount: 942.602
160499504 Ezra Green Total Amount: 1419.5283
467208005 Elijah Taylor Total Amount: 1548.5774
112130784 Oliver Williams Total Amount: 2499.3008
562068378 Mia Wilson Total Amount: 3075.7256
007595279 Emma Wright Total Amount: 3270.618
509544648 Ellie Thompson Total Amount: 4286.8335
439357777 Olivia Davies Total Amount: 4664.2188
698809670 Leo Evans Total Amount: 4738.786

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 3

Enter User ID: 55
Cannot find the user!

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 3

Enter User ID: 467208005

467208005 Elijah Taylor Total Amount: 1548.5774
Accounts:
1. Type: C Amount: 124.0142 Common: 254.17111
2. Type: B Amount: 148.0509 Common: 184.4575
3. Type: C Amount: 299.00137 Common: 612.813
4. Type: A Amount: 368.9621 Common: 497.13568

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
```

```
Option: 4

Set A: 11
Set B: 2
Set C: 3
Set D: 4

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 5

Choose sort type:
1. By ID
2. By Total Amount
Option: 2

Enter sort limit: 7
Sort duration: 0

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
6. Reset to the original unsorted array
0. Exit
Option: 2

837843461 Levi Wilson Total Amount: 1102.4358
245567968 Mateo Wright Total Amount: 1466.4962
160499504 Ezra Green Total Amount: 2278.7087
467208005 Elijah Taylor Total Amount: 5623.7314
509544648 Ellie Thompson Total Amount: 7462.8423
112130784 Oliver Williams Total Amount: 7969.7905
562068378 Mia Wilson Total Amount: 8589.844
439357777 Olivia Davies Total Amount: 13577.893
007595279 Emma Wright Total Amount: 13662.077
698809670 Leo Evans Total Amount: 14696.373

What do you want to do?

1. Generate random users
2. List users
3. Show user with ID
4. Set conversion rates
5. Sort users
```

```
6. Reset to the original unsorted array
0. Exit
Option: 0

Bye!
```

Test your application with different scenarios. How the sorting time change with different limits for the hybrid sort? How it changes when the array is already sorted?

Preliminary Submission: You will submit an early version of your solution before the final submission. This version should at least include the following:

- The sorting algorithm should be completed. You should have the required user logic done so that you can test your sort algorithm at least by user ID.

You will have time to complete your solution after you submit your preliminary solution. You can consult the TAs and tutors during the lab. Do not forget to make your final submission at the end. Even if you finish the assignment in the preliminary submission, you should submit for the final submission on Moodle.

Not completing the preliminary submission on time results in 50% reduction of this assignment's final grade.