Filip Segota

Feb.16.21

Donation Center GUI

**Problem Summary:**  
Develop a program to track donations and donors for a donation center, including GUI.

**Implementation Requirements:**  
> The program must be able to lookup donations of a specific donor (all and set period.)  
> Data must be kept between runnings of the program.

* We can’t go over estimated time

**System Design**Diagram

Description automatically generated

**Testing Plan/Report**

|  |  |  |
| --- | --- | --- |
| Test Case | Expected Output | Actual Output |
| **Change of name** Change of name of a donor without losing donations in the event of a name change. Donor.changePerson(New Person(“John”,[email@email.com](mailto:email@email.com))); Donor.getName();  Donor.getEmail(); | John  [email@email.com](mailto:email@email.com) | John  [email@email.com](mailto:email@email.com) |
| **No Donations in time period** The donations of a specific donor are requested but the donor has no donations during that time span Driver.getDonations(01, 01.01.20); | empty array of Donation[] | empty array of Donation[] |
| **No Donations for donor** The donations for a donor are requested but that donor has not yet  Driver.getDonations(01); | empty array of Donation[] | empty array of Donation[] |
| **Donation for new Donor and retrieval** A new donor is created, makes a donation, and the donation is retrieved.  newDonor(“Person”, True, new Person(“John Smith”, [email@email.com](mailto:email@email.com));  newDonation(10.00, monetary, getByID(01), “test donation);  getDonations(01); | Id: 01  Void  Array of Donation[] with one entry of 10.00 monetary. | Id: 01  Void  Array of Donation[] with one entry of 10.00 monetary. |
| **Many donations and retrieval** Many donations of varying amounts and types.  newDonation(random,random, getDonor(01), “random”); x 100 getDonations(01); | Array of Donation with one hundred donations of varying size and type. | Array of Donation with one hundred donations of varying size and type. |
| **Testing of classes** |  |  |
| **Person and Donor**  Donor donor1 = new Donor(“Family”, true, “Johnson”, “john.johnson@gmail.com”, false, false);  donor1.changeName(“Jackson”);  System.out.println(donor1.getName());  donor1.changeEmail("jackson@gmail.com");  System.out.println(donor1.getEmail());  System.out.println(donor1.isBusiness());  System.out.println(donor1.isForProfit());  System.out.println(donor1.getType());  System.out.println(donor1.getDonorID());  System.out.println(donor1.getActiveState());  donor1.changeActive(false);  System.out.println(donor1.getActiveState());  donor1.changeActive(true);  System.out.println(donor1.getIndex());  Donation donation1 = new Donation(500.00, “1”, donor1, java.time.LocalDate.of(2021, 1, 26), “description”);  donor1.addDonation(donation1);  System.out.println(donor1.getDonations());  System.out.println(donor1.getStartDate());  System.out.println(donor1.getLastDate());  Person person2 = new Person(“Peter”, “”, false, false);  System.out.println(donor1.getIndexFam());  donor1.addFamilyMember(person2);  System.out.println(donor1.getIndexFam());  person2.changeName(“Joe”);  person2.changeEmail(“joe@gmail.com”);  System.out.println(person2.getName());  System.out.println(person2.getEmail()); | “Jackson”  “jackson@gmail.com”  “false”  “false”  “Family”  “0”  “true”  “false”  “0”  array with one donation  “2021-01-26”  “2021-01-26”  “0”  “1”  “Joe”  “joe@gmail.com” | “Jackson”  “jackson@gmail.com”  “false”  “false”  “Family”  “0”  “true”  “false”  “0”  -//-  “2021-01-26”  “2021-01-26”  “0”  “1”  “Joe”  “joe@gmail.com” |
| **Donation**  System.out.println(donation1.toString());  donation1.setType(“2”);  System.out.println(donation1.getType());  donation1.setDate(java.time.LocalDate.of(2021, 1, 25));  System.out.println(donation1.getDate());  donation1.setDescription(“changed description”);  System.out.println(donation1.getDescription());  donation1.setValue(300.00);  System.out.println(donation1.getValue());  Donor donor2 = new Donor(“Family”, true, “Johnson”, “john.johnson@gmail.com”, false, false);  donation1.setDonor(donor2); | “1 500.0 description 2021-01-26”  “2”  “2021-01-25”  “changed description”  “300.00”  New donor set | “1 500.0 description 2021-01-26”  “2”  “2021-01-25”  “changed description”  “300.00”  New donor set |
| **Bad cases**  Incompatible type - passing the incompatible argument to the method (ex. Int instead of String, String instead of Donor,...)  Calling a method of the wrong class (ex. Calling a method of Donation class with Donor instance)  No suitable constructor found (ex. Not given enough argumetns to the constructor) | Example  person2.changeName(2);  donor1.getDescription();  Donor donor3 = new Donor(“Family”); |  |

**Analysis of Time Required**  
In my initial estimation of building a GUI, I estimated that it’s going to take me 15 hours to finish it

It took me 15 hours to do as much as I did and if I had just one more hour, I would bring all the components together nicely. I haven’t finished showing all the donations of the specific donor, and also showing all the donations there are.

**Identification of Outside Resources**An online chart software was used to design the UML diagram.

**Potential Security Risks**> Data may be able to be accessed outside of the program.  
> No authentication required to access donor data.  
> Donor data is stored using sequential ids which could easily and programmatically be enumerated by a malicious actor by feeding sequential ids in.

Donations or family can get out of bounds. That is potential security risk. That can be prevented by checking if those arrays are full before adding another member.

**Potential Ethical Issues/Ethical Report**> Storing user data unencrypted and relatively unprotected on the user’s hard disk.

**Future Improvements**> Encrypt user data before it is saved to hard drive.  
> Implement non sequential user ids.

> There is little input verification so more of that would be helpful.

* Finish the GUI(specific donor’s donations, and all the donations there are)

**Lessons Learned**> It’s very important to make sure methods are placed in a class that makes sense.  
> Beginning to learn how to properly divide the work between two people.

* Learned a lot about making a GUI, including components, listeners, and so on.
* Learned that to make a whole program with all its components, it requires a lot of time, and it’s best if each task is spread, so we can focus more on one thing until we finish it, and only then moving to next things.
* Learned about time management.
* Learned a lot about event-programming.