Implementation process started right after completing design, according to waterfall model. Work has been divided and one part of group started with model classes, when the other part was designing GUI and proceeded with its implementation. We found this approach more efficient due to parallel system development. Moreover, too much people at one task was problematic, because of merge conflicts in git and coding ideas missmatch.

During this process, the design was constantly changing as to adjust to new ideas or requirements which came out

Now I will present few interesting code examples. First one is Member List. This class is responsible for primary storing all members signed up to VIPASSANA, as an ArrayList.

Lists classes, beside storing objects, provide basic operations on lists, such like adding and deleting objects, searching through lists with desired criteria or generating new, specialized lists. Here we can see three methods. getListOfMembersWhoHasntPaid() searches list, and checks if member paid. If not, object is added to a new list, which will be returned and passed further inside the system. getListOfEmails() generates email list of all members. It is used when administrator wants to send newsletter. <change picture>

Next important part of implementation is Controller. This is our connection between model and view. One of the most extensively used type of method is that one, which returns DefaultTableModel. Hence, we are using tables to present data in GUI, table model is needed. When table is about to be created, view calls one of the static methods in controller which returns appropriate model for table. Firstly, we define column names. Secondly, we get desired data from model using VIAManager, and we are parsing list into two-dimensional array, where one row represents one table entry. Thirdly, model is created. However, we want to have more control over our table. That’s why we extend DefaultTableModel with anonymous inner class, overriding getColumnClass() and isCellEditable() methods. In getColumnClass() type of each column is defined. For example, column with index 3 is type Boolean, which will result displaying it as a column of checkboxes. In isCellEditable() we enable editing only for column with checkboxes.

Last class which will be presented is FileManager. Here all saving and reading from files is handled. We have two types of reading data, binary and text files. The purpose of text files reading is to load initial data into system. It was much faster to write methods for this and prepare text files, then insert data manually. That approach was very helpful during testing, when we already had data to work with. In this case, information about member object are written in text file in specific format. We read each line, separate it, and store information into variables. When all data is retrieved, Member object is created and added to list.

Second part of FileManager is binary files reading and storing. Since all our classes in model implement Serializable interface, we can store them using Java Object Serialization and save whole lists as an object into file.

readEventFile() method is an example of reading binary file using ObjectsInputStream. Firstly, we create ObjectInputStream instance using FileInputStream. Secondly, we read whole EventList object. Due to using IDs in our system, ID in Event class must be restored.

FileManager is also responsible of reading and generating newsletter files. Here we can see how state of newsletter files is being restored into the system. Newsletters are stored as ArrayList of text files, and all newsletters files names are stored in file allNewsletters as we generate name of newsletter file based on current date. File with names is being read and in loop we create files, until we reach end of file.

Last part of implementation process was testing. After finishing one part of system, all members were manually testing this part, searching for bugs. In a result, before moving to next part, majority of issues ware fixed. Moreover, we had our system tested by professional manual tester with ISTQB certificate, who also found some bugs.

And that is all from implementation part. Now Miśka will talk about process.