Scenario:

A person wants to buy bus passes on their opus card or to check how many they have left.

Design Paradigm and Expected Output:

With this machine, the user can choose to check their card balance, to buy tickets or to cancel the latest transaction. When the user checks their card, they can see how many trips they have left or if their card is charged for the month or for the week. When the user wants to buy tickets, they have the option to buy individual trips, to get a monthly pass or a weekly pass. If they regret having done a transaction, they have the option to cancel it. However, they can only cancel the latest transaction.

To be able to do this, the user would have to write their card's ID. If they don't have an ID yet, then they would have to register by putting their name, last name and their status (student or a normal user) and the machine would provide them with an ID for their card. If a user forgets their card ID, they'll have to call someone who will get access to all the accounts by writing a passcode. They'll then be able to see what ID their name is registered under.

Hierarchies:

There are two hierarchies: card and ticket

Card: the user can have a student card (with an applied discount) or a normal card.

Ticket: the user can buy a monthly ticket, a weekly ticket or an individual trip.

Interfaces:

Rechargeable: this interface is implemented by the Monthly and the Weekly classes. It contains a method recharge() that allows the user to activate the monthly or the weekly passes. It needs to be an interface since the IndividualTrip class inherits from the same parent as Monthly and Weekly, but it's not rechargeable.

Runtime-Polymorphism methods:

The recharge() method in the Monthly and the Weekly classes that apply runtime-polymorphism since it's in the Rechargeable interface and it is written in different ways in those classes, the method is overridden.

TextIO:

The Accounts class uses textIO to write the accounts in a file and to read from the file and put it in the TreeSet of cards. It is also used in the methods of the UserInputManger class to register a card, to check the balance (by reading the information from the file), to buy a pass, to cancel a transaction (both updating the card's information in the file).

Comparable and Comparator:

The Card class implements Comparable. Since Cards is an object, the TreeSet in the Accounts class needs to know how to sort the Cards. This class also needs a Comparator to sort and display the accounts the way the person who checks the accounts wants it (by first name or by last name).

Deliverable 2:

All the methods in the Card class (and its children), in the Owner class, in the Ticket, (and its children) and the Rechargeable interface will be implemented for deliverable 2. The methods with scanners and TextIO will be done if time will permit it, or else they will be done for deliverable 3.

UML class diagram: