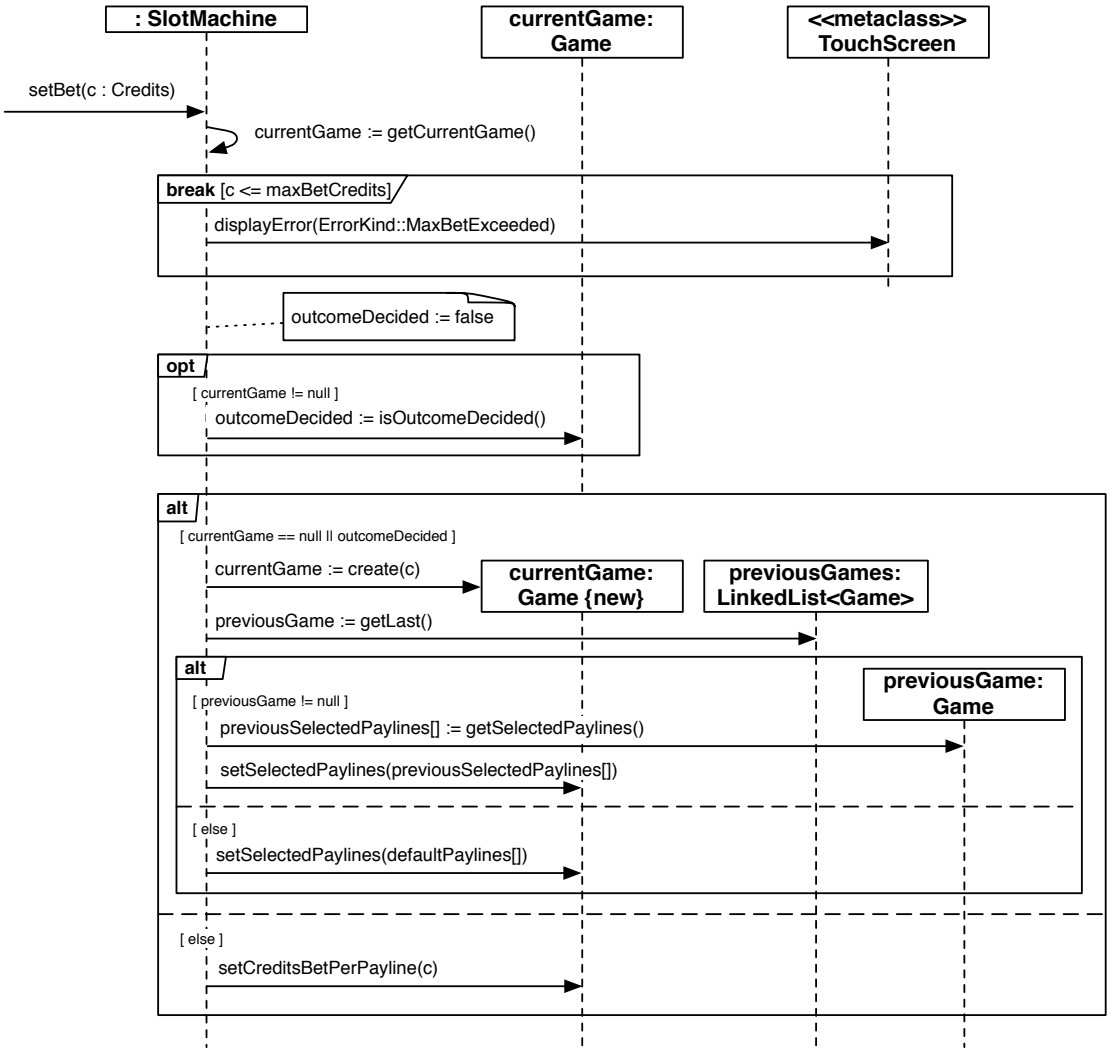
COMP-533 – Final: Video Slot Machine

Céline Bensoussan and Matthias Schöttle
celine.bensoussan@mail.mcgill.ca, mschoettle@cs.mcgill.ca

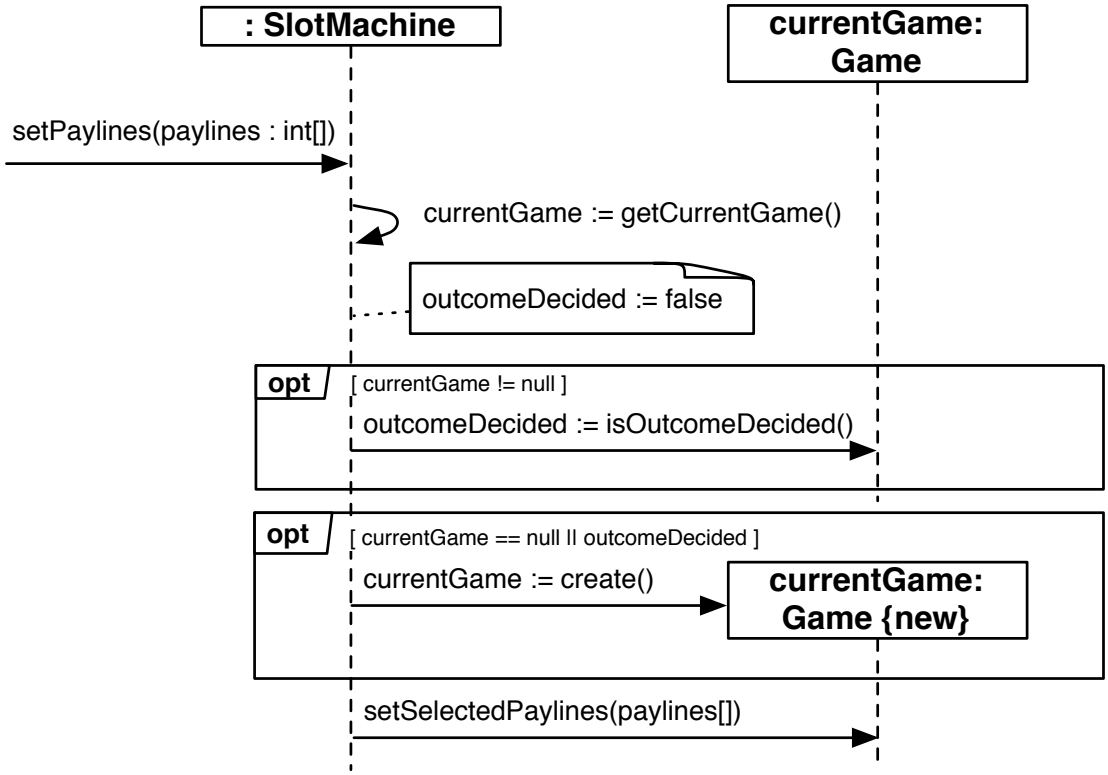
November 26, 2013

Interaction Model

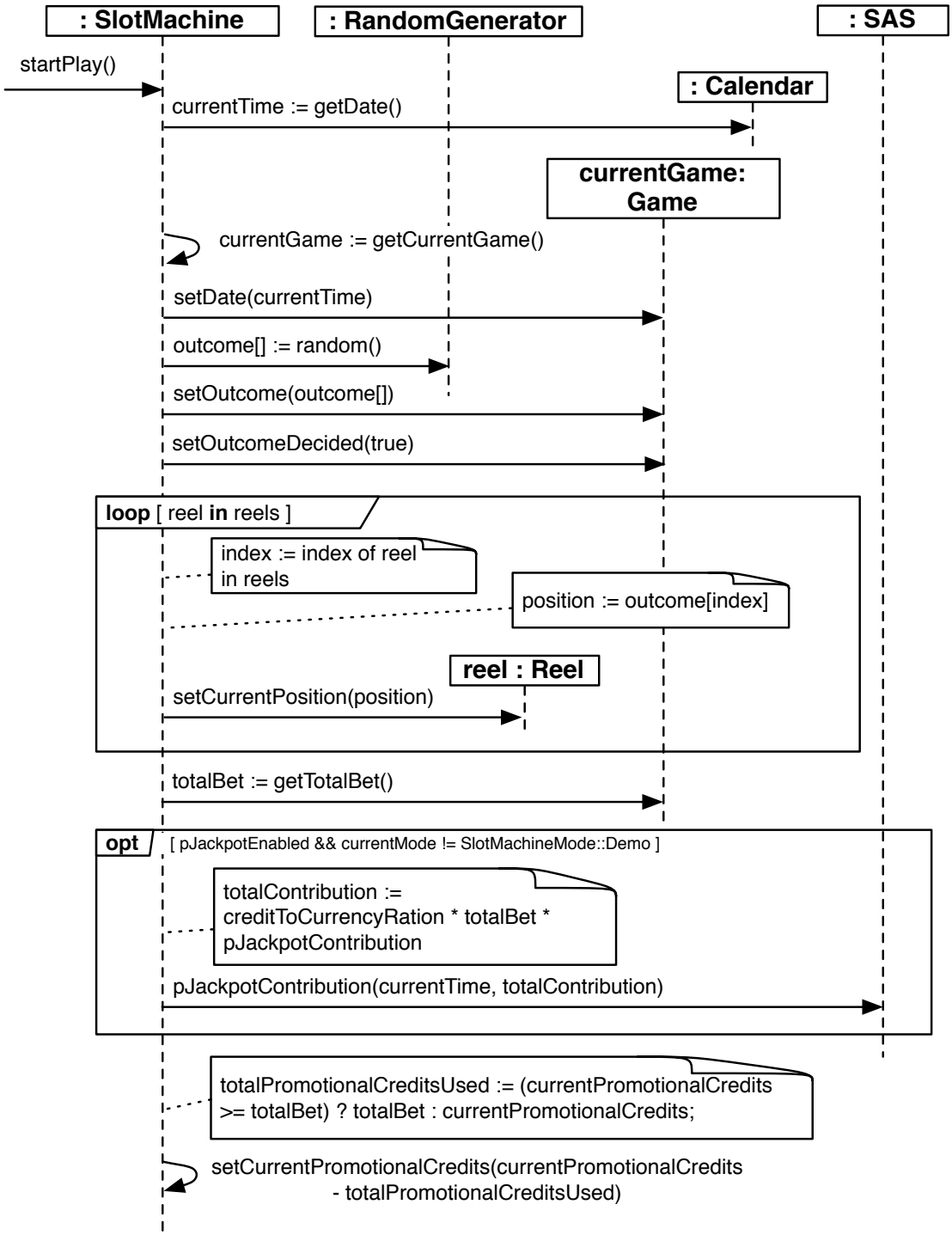
setBet Operation



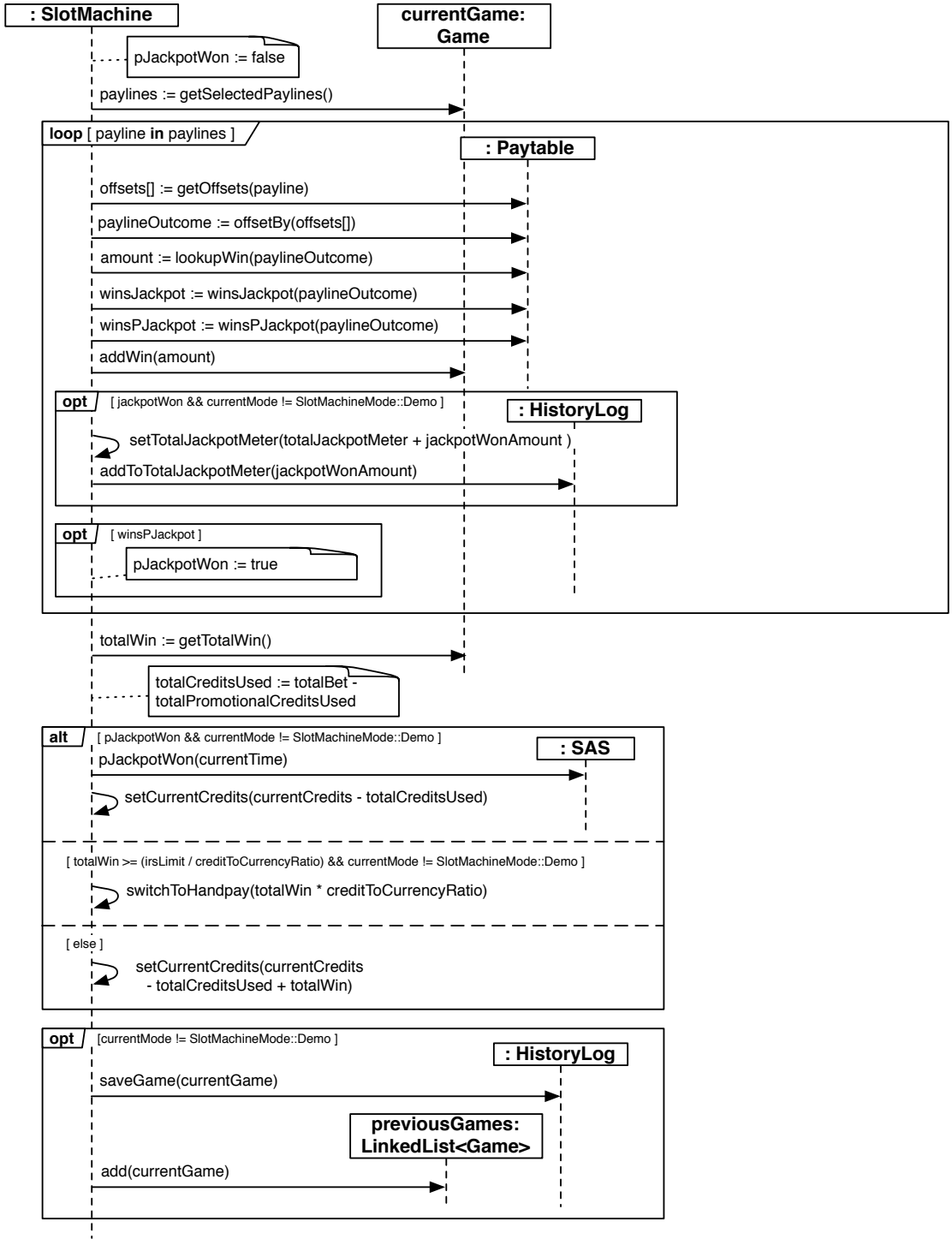
setPaylines Operation



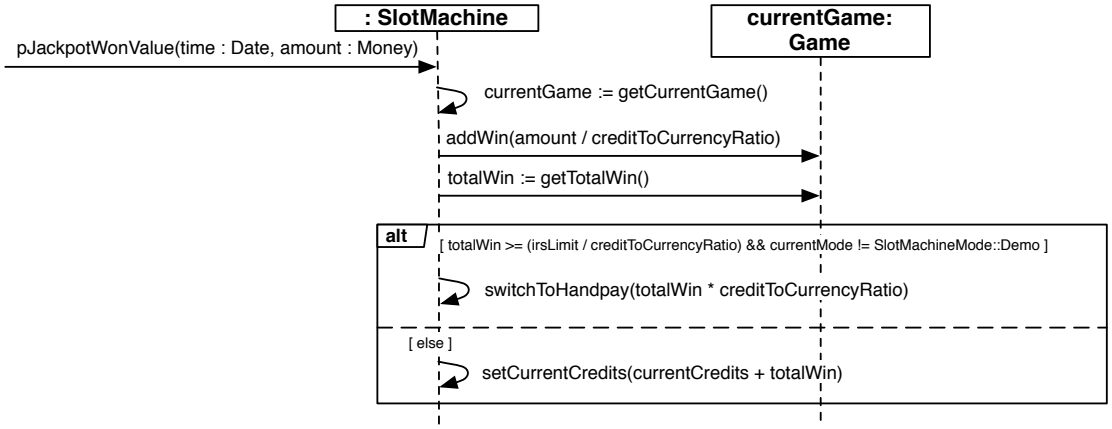
startPlay Operation (1/2)



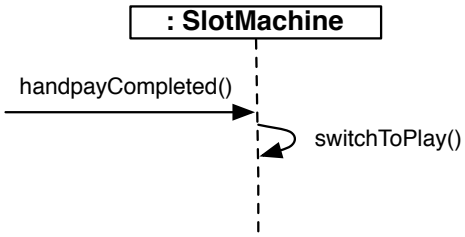
startPlay Operation (2/2)



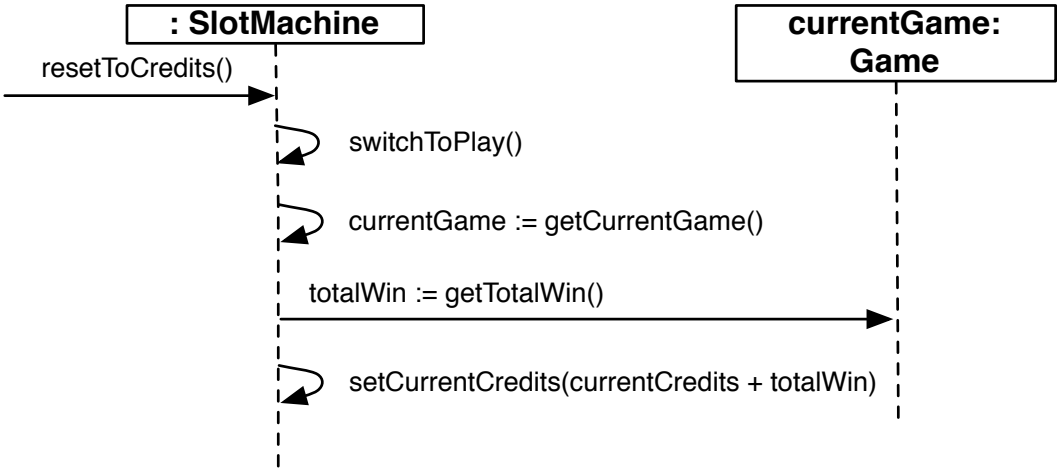
pJackpotWonValue Operation



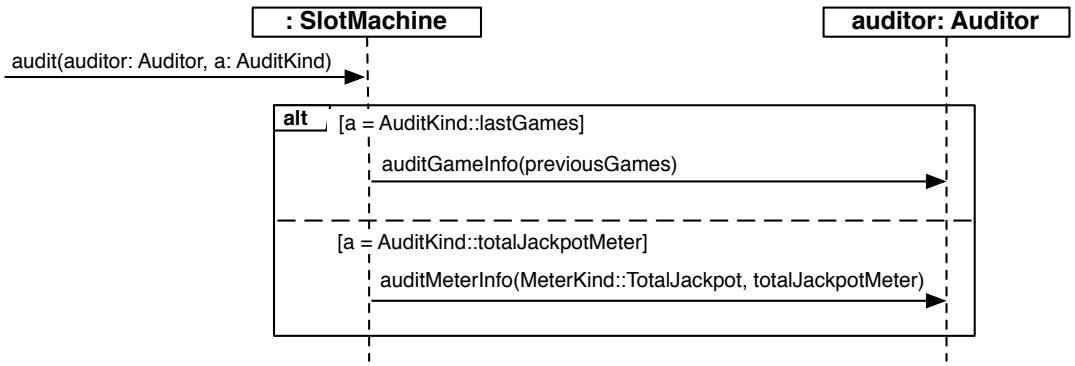
handpayCompleted Operation



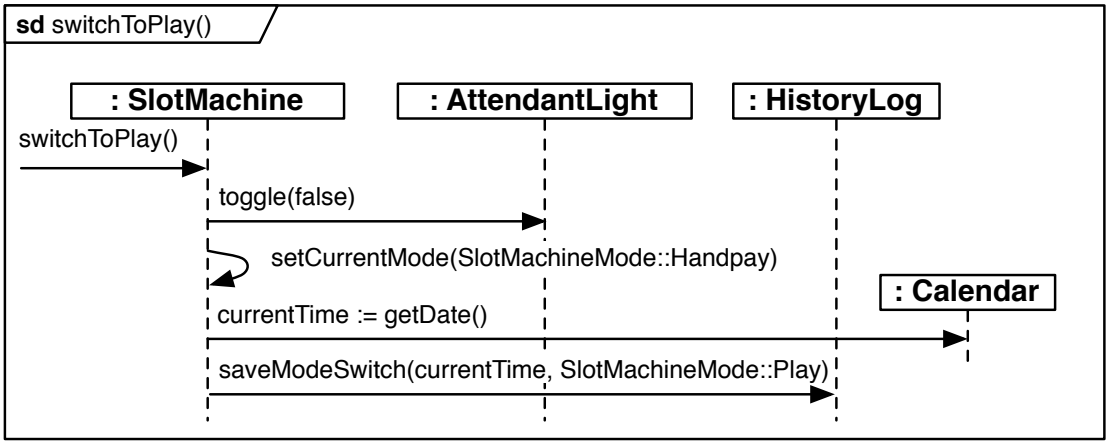
resetToCredits Operation



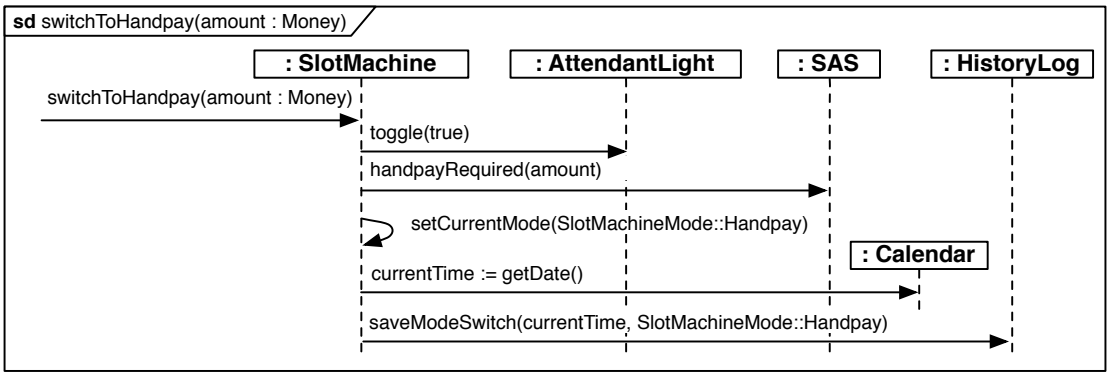
audit Operation



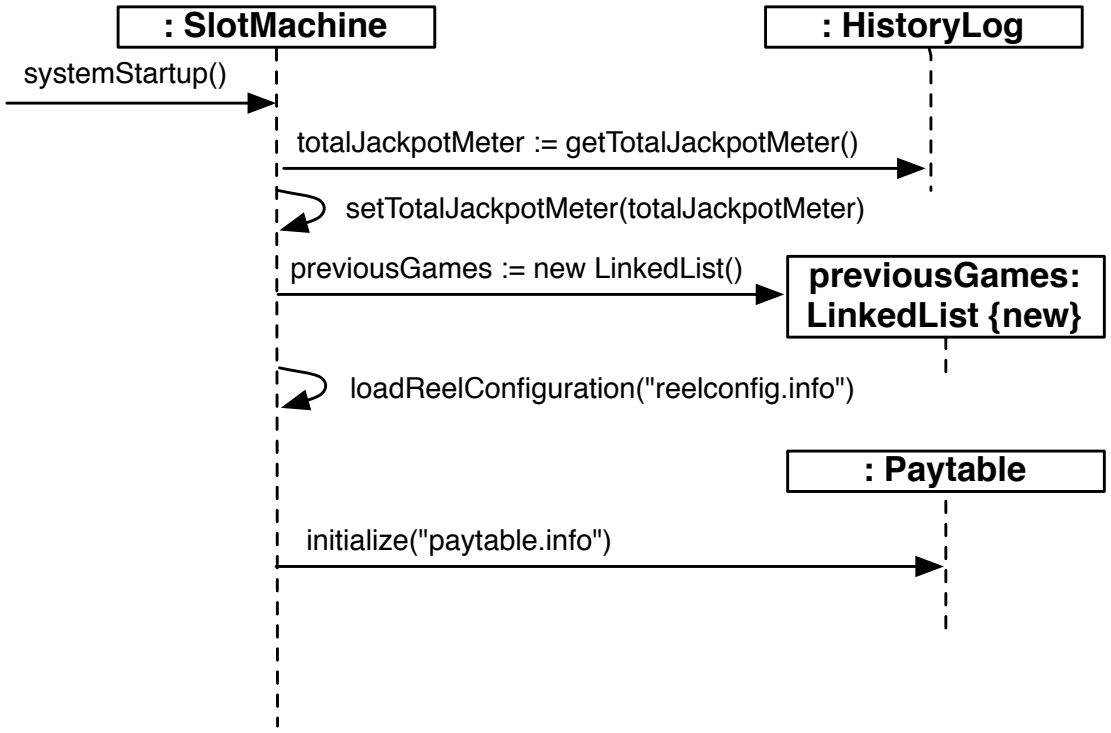
switchModeToPlay Operation



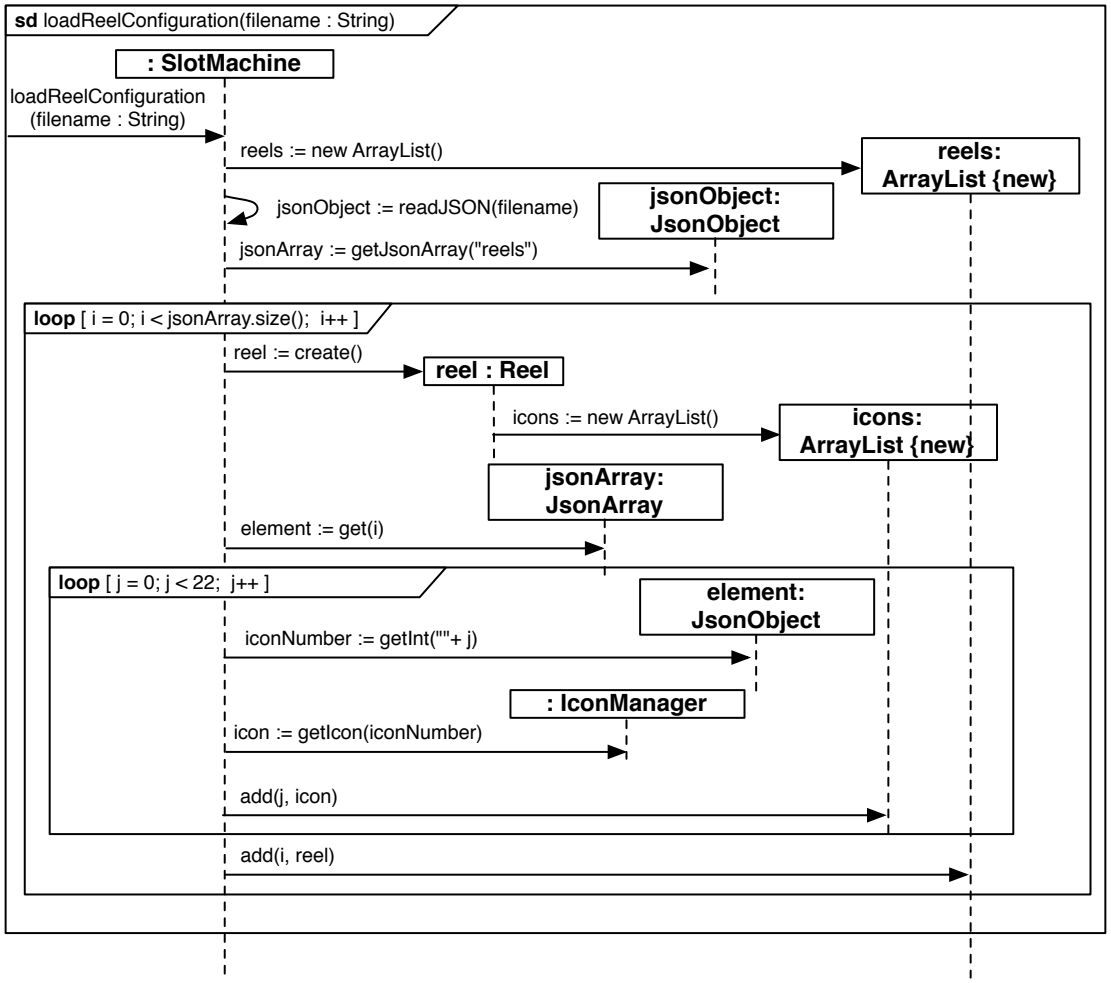
switchModeToHandpay Operation



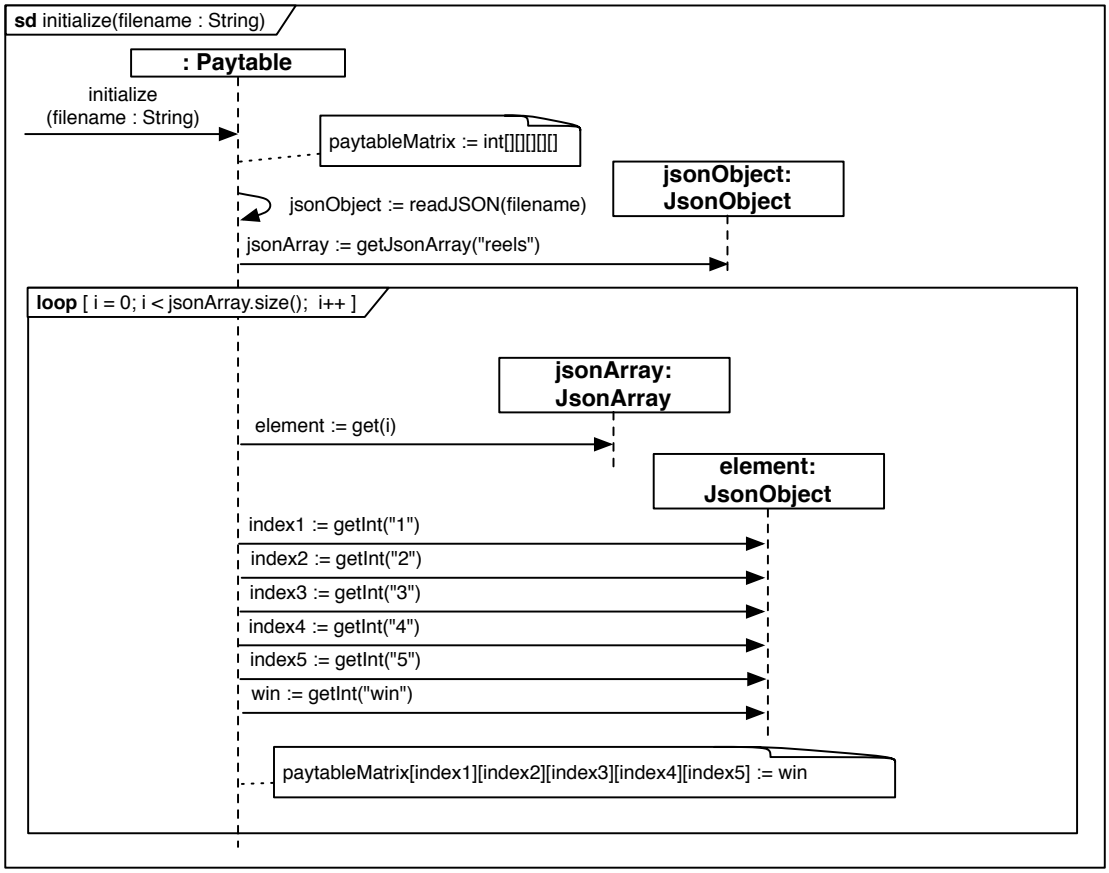
systemStartup Operation



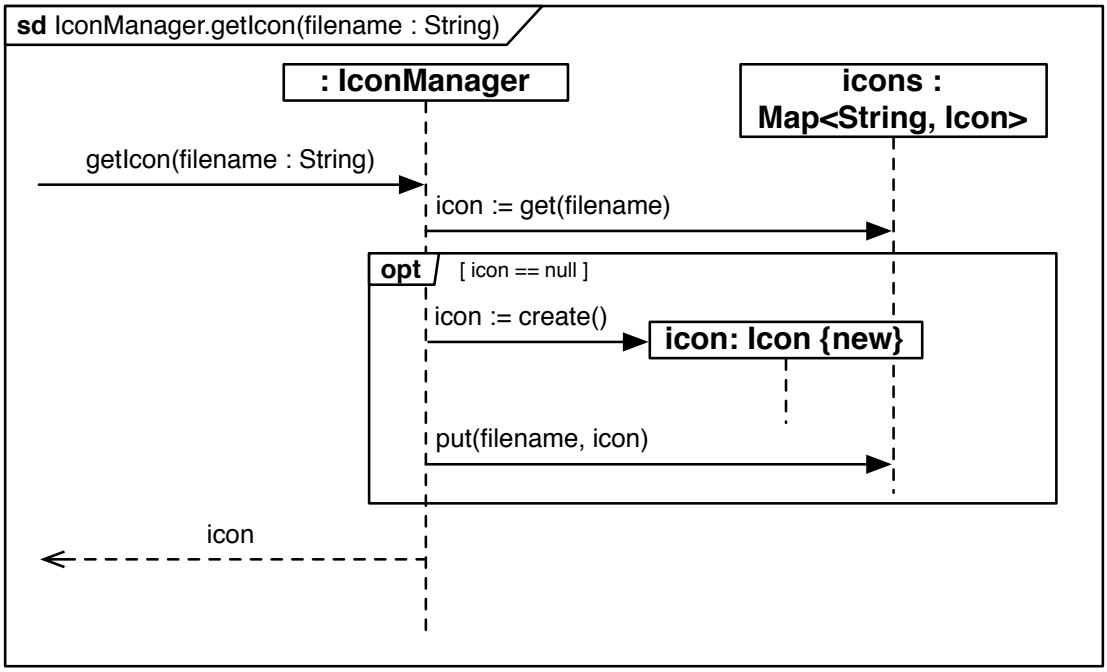
loadReelConfiguration Operation



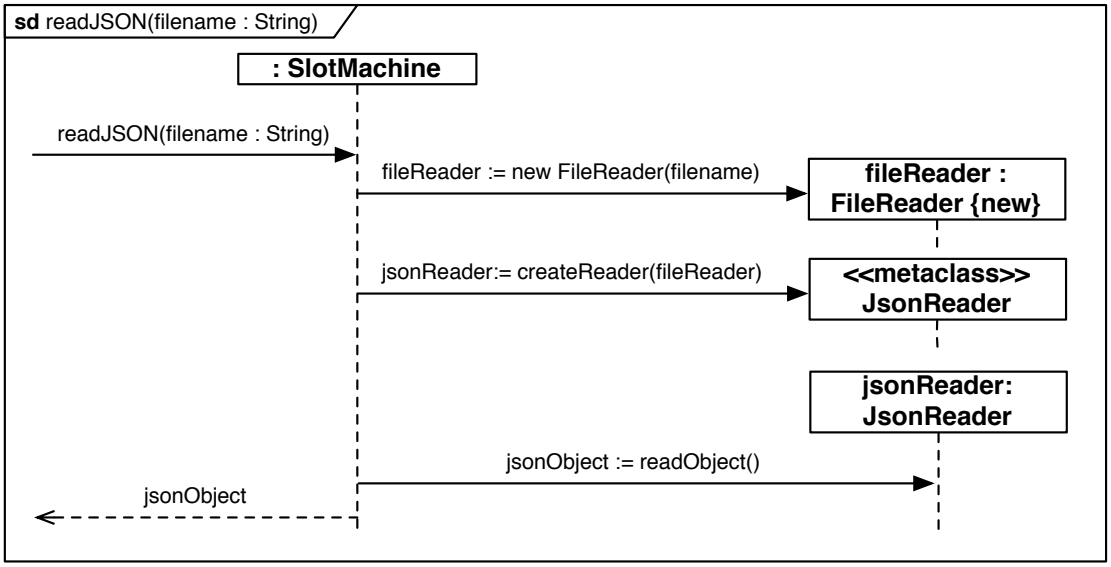
initialize Operation



getIcon Operation



readJSON Operation



Design Class Model

Please refer to the additional .ram file. Due to some limitations in TouchRAM, please take the following comments into consideration:

- **Datatypes:** The following datatypes are modelled as classes with the suffix "*Datatype*". Wherever possible/needed they are used as the return or parameter type. However, as they cannot be used for attributes, their base types are given then.
  - MoneyDatatype: Encodes money, i.e., a non-negative real number.
  - CreditsDatatype: Encodes credits, i.e., a non-negative natural number.
- **Enums:** The following enums are modelled as classes with the suffix "*Enum*". Their complete definition is given in figure 1. Wherever possible/needed they are used as the return or parameter type. However, as they cannot be used for attributes, their base types are given then.
- **Arrays:** Arrays are modelled as classes and prefixed with "*ArrayOf*" and followed by the type. Wherever possible/needed they are used as the return or parameter type. However, as they cannot be used for attributes, they are associated to the appropriate class. The multiplicities denote the size of the array. Whenever an array is used in the interaction model, it is suffixed with "[ ]".
- **HistoryLog:** This is the interface to the non-volatile memory. In our case a database is used, which means that SQL queries are executed when one of the defined operations are invoked.
- **Previous Games:** Previous games are held in a linked list (*LinkedList<Game>*) for the current player. This means that whenever a cashout occurs, the list is emptied. A linked list is used in order to be able to get the last game by calling *getLast()*.
- **File format:** The configuration files are in JSON format and are read using the JSON classes provided by Java 7 in the package *javax.json*.
- **Utility classes:** Some utility classes are not explicitly modelled, such as the collection for *previousGames*. Just the association is modelled. The used collection type can be seen in the interaction model. An exception is Map for the IconManager, which maps a unique integer to the icon.
- **Singleton/system-wide classes:** The following classes are system-wide and as such are singletons. However, for space reasons in the interaction model calling the getInstance method first to get the actual instance is omitted. In the design class model the static method *getInstance* is shown.
- **Paytable matrix:** The paytable matrix has an association to *ArrayOfInt*. However, it is actually a 5-dimensional integer array to map icon numbers to the amount that gets won with that icon configuration.

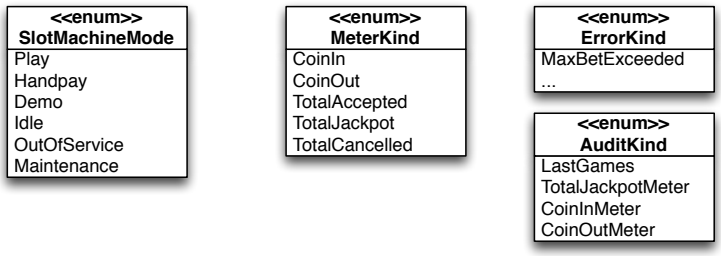


Figure 1: Enums.