CSE1729 - Introduction to Programming | November 14, 2017

Problem Set 9

1. Define a SCHEME function named `make-checking` which takes one parameter, a beginning balance and represents a checking account object. This checking account object should store the beginning balance along with any and all transactions that have been posted to the account. You may also wish to store the current balance separately. The checking account object should expose functions for depositing funds, writing a check, returning an account statement and returning the current balance. When returning an account statement, the object should evaluate to a string containing the beginning balance, each transaction in the order it was posted to the account, and the current balance as shown below. You must also use the tokens as named in the example below to retrieve your functions from the dispatcher for your object.

   Note: You will want to make use of the `number->string` function that produces a string containing the decimal representation of the number. Also, the `string-append` function can take several strings as parameters and produces one string which is the concatenation of all the strings passed as parameters.

```
>(define checking (make-checking 100))
>((checking 'write-check) 10)
>((checking 'write-check) 10)
>((checking 'deposit) 100)
>((checking 'write-check) 10)
>(display ((checking 'print-statement)))
>((checking 'balance))
beginning balance: 100
transaction: check amount: -10
transaction: check amount: -10
transaction: deposit amount: 100
transaction: check amount: -10
balance: 170
170
```

2. Define a SCHEME function named `make-clock` which takes two parameters which represent the current time in hours and minutes. Your clock object should store the current time in minutes only. Your clock object should expose three functions for working with the clock object:

   - `'tick` - advances the time of the clock object by one minute
   - `'time` - returns a string with the current 12-hour time indicating whether the current time is AM or PM
   - `'military` - returns a string representing the current time in military (24-hour) time

```
>(define get-time (clock 'time))
>(define get-mil (clock 'military))
>(display (get-time))
>((clock 'tick))
>(display (get-time))
>((clock 'tick))
>(display (get-time))
>(display (get-mil))
9:00 PM
9:01 PM
9:02 PM
21:02
```

Note: If the time is displayed in military time, your object should print any leading zeros for the hour. Both time formats should print leading zeroes for the minutes. For example:

```
>(define clock (make-clock 2 0))
>(define get-time (clock 'time))
>(define get-mil (clock 'military))
>(display (get-time))
>(get-mil)
>((clock 'tick))
>(display (get-time))
>(display (get-mil))
2:00 AM
02:00
2:01 AM
02:01
```

3. Define a SCHEME function, named `make-book`, which takes two string parameters (author and title) and implements an object that represents a book. It should expose the following functions:

   - `get-author` - which returns a string representing the author's name
   - `get-title` - which returns a string representing the title of the book

4. Design and implement a SCHEME object that represents a library. Your library should be able to store book objects. You should be able to create a library by evaluating the `make-library` function which takes no parameters. Therefore, the library starts with no books. Your library object should expose the following functions:

   - `add` - which takes one parameter, a book object, and stores it inside the library
   - `find-author` - which takes one string as a parameter, the author's name, and returns a list of author/title pairs (as a pair of strings) for all book objects in the library which were written by that author
   - `find-title` - which takes one string as a parameter, the title of the book(s) you wish to find, and returns a list of author/title pairs for all book objects in the library which share that title

   You should be able to test your code with the following test data:

```
(define mylib (make-library))
((mylib 'add) (make-book "Harry Potter and the Philosopher's Stone" "Rowling"))
((mylib 'add) (make-book "Harry Potter and the Chamber of Secrets" "Rowling"))
((mylib 'add) (make-book "Harry Potter and the Prisoner of Azkaban" "Rowling"))
((mylib 'add) (make-book "Harry Potter and the Goblet of Fire" "Rowling"))
((mylib 'add) (make-book "Harry Potter and the Order of the Phoenix" "Rowling"))
((mylib 'add) (make-book "Harry Potter and the Half-Blood Prince" "Rowling"))
((mylib 'add) (make-book "Harry Potter and the Deathly Hallows" "Rowling"))
((mylib 'add) (make-book "The Hunger Games" "Suzanne Collins"))
((mylib 'add) (make-book "Catching Fire" "Suzanne Collins"))
((mylib 'add) (make-book "Mockingjay" "Suzanne Collins"))
((mylib 'add) (make-book "The Magician" "Michael Scott"))
((mylib 'add) (make-book "The Magician" "W. Somerset Maugham"))

((mylib 'find-title) "The Magician")
((mylib 'find-author) "Suzanne Collins")
((mylib 'find-author) "Rowling")
```

5. Define a SCHEME function, named `make-track`, which takes one string parameter (title) and implements an object that represents a music track. The music track should have three *attributes* (title, artist and album) even though the function that creates it only takes one as a parameter. These attributes can be set initially to a default string value (e.g. "unknown"). Your `music-track` should expose the following *accessor* and *mutator* functions:

   - `get-title` - which returns a string representing the title of the music track
   - `get-artist` - which returns a string representing the artist's name
   - `get-album` - which returns a string representing the album in which the track was released
   - `set-title` - which changes the string representing the title of the music track
   - `set-artist` - which changes the string representing the artist's name
   - `set-album` - which changes the string representing the album in which the track was released

6. Design and implement a SCHEME object that represents a music library. Your music library should be able to store music track objects. You should be able to create a library by evaluating the `make-music-library` function which takes no parameters. Therefore, the music library starts with no tracks. Your music library object should expose the following functions:

   - `add` - which takes one parameter, a music track object, and stores it inside the library
   - `find-by-artist` - which takes one string as a parameter, the artist's name, and returns a list of lists containing the track title, artist, and album for all music track objects in the library which were recorded by the artist
   - `find-by-title` - which takes one string as a parameter, the title of the music track(s) you wish to find, and returns a list of lists containing the track title, artist, and album for all music track objects in the library which share that title
   - `find-by-album` - which takes one string as a parameter, the title of an album, and returns a list of lists containing the track title, artist, and album for all music track objects in the library which were released as part of that album

Submitting Solutions

Once you have finished:

1. Save your work to a file using the appropriate filename, problemSet9.rkt.
2. Submit your solution file for grading via the MIMIR site.