Understanding Abstractions

We start with a series of questions we would like to answer about two different lists of data.

The first list consists of the entries in the Boston Marathon. For each runner we record the name, gender, age, bib number, the starting time and the ending time of the run (in minutes only).

The second list is our familiar list of bank accounts that can be `Checking`, `Savings`, or `Credit` account.

We would like to write the methods for the first list that produce the following:

1. A. list of all male runners under 40 years old
2. B. count all male runners under 40 years old
3. C. Is there a male runner under 40 years old in the list?

1. A. list of all female runners who finished in under 180 minutes
2. B. count all female runners who finished in under 180 minutes
3. C. Is there a female runner who finished in under 180 minutes?

1. A. list of all runners over 40 who finished in under 180 minutes
2. B. count all runners over 40 who finished in under 180 minutes
3. C. Is there a runner over 40 who finished in under 180 minutes?

We would like to write the methods for the second list that produce the following:

1. A. list of all accounts with the balance under $200
2. B. count all accounts with the balance under $200
3. C. Is there an account with the balance under $200 in this list?

1. A. list of all accounts with the given customer name
2. B. count all accounts with the given customer name
3. C. Is there an account with the given customer name in this list?

1. A. list of all savings accounts
2. B. count all savings accounts
3. C. Is there a savings account in this list?

There is a lot of repretition in the above questions. Think first how could you solve all these problems easily in Scheme.

Hint: The solution to the problem 3. C. in the last set of questions would be simple:

```
;; is there a saving account in this list?
;; has-savings: [Listof Acct] -> boolean
(define (has-savings acct-list)
  (ormap savings? acct-list))
```

It would be a bit harder for the last question about the marathon runners. Assume we have the following data definition for the runner:

```
;; A Runner is
;;  (make-runner String Symbol Number Number Number Number)
(define-struct runner (name gender age bib start finish))
```

The solution would then be:

```
;; is there a runner in this list who is over 40
;; and who finished in under 180 minutes?
;; has-runner: [Listof Runner] -> boolean
(define (has-runner runner-list)
  (ormap (lambda (x)
           (and (> (runner-age x) 40)
                (< (- (runner-finish) (runner-start)) 180)))
         runner-list))
```

Now, how would we handle these abstractions in Java?

Hint: The book and numerous labs and lecture notes from the previous years can help you.