CSCI 3055U, Assignment 2, Programming in Clojure
Calvin Lo #100514352

(4) Complete the following table:

| | Advantage | Disadvantage |
|---|---|---|
| `(defn render [data] ...)` | Fast in the first few expressions | Modify existing code when adding new code |
| multimethod | Very flexible<br>Support arbitrary dispatch<br>Can create ad hoc taxonomies | More expensive<br>Slow |
| protocol | Datatypes can implement multiple protocols<br>Provide only specification, not implementation<br>Existing datatypes can be extended<br>Protocol method are namespaced<br>faster | Doesn't allow complex stuff (only one type)<br>Doesn’t support arbitrary dispatching |

(5) What are some ways of handling inheritance?

Interfaces: provides specification, not implementation.
Protocols: provides specification, not implementation.
multimethod dispatch: know about java inheritance.
Collections: java inheritance hierarchy.
Prefer method: multiple inheritance.
Derive: inheritance in ad hoc types.
Record: base type and subtype .

(6) What does the following code do?

```
(defn g
  ([f & colls]
     (apply concat (apply map f colls))))
```

(defn g ([f & colls] will declare a function called g and take a string vector as arguments. (apply map f colls) will join the character with same position within each elements in the string vector and form a new sequence. (apply concat (apply map f colls)) will separate the sequence provided in the previous step character by character and form a new sequence.]

Eg. Input str["abc" "def"]
    (apply map f colls) will give ("ad" "be" "cf")
    (apply concat (apply map f colls)) will give (\a \b \c \d \e \f)