Episode 3 Codes for printing

game.clj

```
(ns undead.game)

(def faces [:h1 :h1 :h2 :h2 :h3 :h3 :h4 :h4 :h5 :h5
            :fg :fg :zo :zo :zo :gy])

(defn ->tile [face]
  {:face face})

(defn create-game []
  {:tiles (shuffle (map ->tile faces))
   :sand (repeat 30 :remaining)})

(defn- revealed-tiles [game]
  (->> game :tiles (filter :revealed?)))

(defn- can-reveal? [game]
  (> 2 (count (revealed-tiles game))))

(defn- match-revealed [tiles]
  (mapv (fn [tile]
          (if (:revealed? tile)
            (-> tile (assoc :matched? true) (dissoc :revealed?))
            tile)) tiles))

(defn- check-for-match [game]
  (let [revealed (revealed-tiles game)]
    (if (and (= 2 (count revealed))
             (= 1 (count (set (map :face revealed)))))
      (update-in game [:tiles] match-revealed)
      game)))

(defn reveal-tile [game index]
  (if (can-reveal? game)
    (-> game
        (assoc-in [:tiles index :revealed?] true)
        (check-for-match))
    game))
```

game_test.clj

```
(ns undead.game-test
  (:require [undead.game :refer :all]
            [expectations :refer :all]))
```

```
(defn- find-face-index [game face]
  (first (keep-indexed (fn [index tile]
                         (when (and (= face (:face tile))
                                    (not (:revealed? tile)))
                           index))
                       (:tiles game))))

(defn reveal-one [face game]
  (reveal-tile game (find-face-index game face)))

;; create-game

(expect {:h1 2 :h2 2 :h3 2 :h4 2 :h5 2
         :fg 2 :zo 3 :gy 1}
        (->> (create-game) :tiles (map :face) frequencies))

(expect #(< 10 %) (count (set (repeatedly 100 create-game))))

(expect {:remaining 30} (frequencies (:sand (create-game))))

;; reveal-tile

(expect 1 (->> (reveal-tile (create-game) 0)
               :tiles (filter :revealed?) count))

(expect #{{:face :h1 :revealed? true}
          {:face :h2 :revealed? true}}
        (->> (create-game)
             (reveal-one :h1)
             (reveal-one :h2)
             (reveal-one :h3)
             :tiles
             (filter :revealed?)
             (set)))

(expect [{:face :h1 :matched? true}
         {:face :h1 :matched? true}]
        (->> (create-game)
             (reveal-one :h1)
             (reveal-one :h1)
             :tiles
             (filter :matched?)))

(expect #{{:face :h3 :revealed? true}}
        (->> (create-game)
             (reveal-one :h1)
             (reveal-one :h1)
             (reveal-one :h3)
             :tiles
             (filter :revealed?)
             (set)))
```