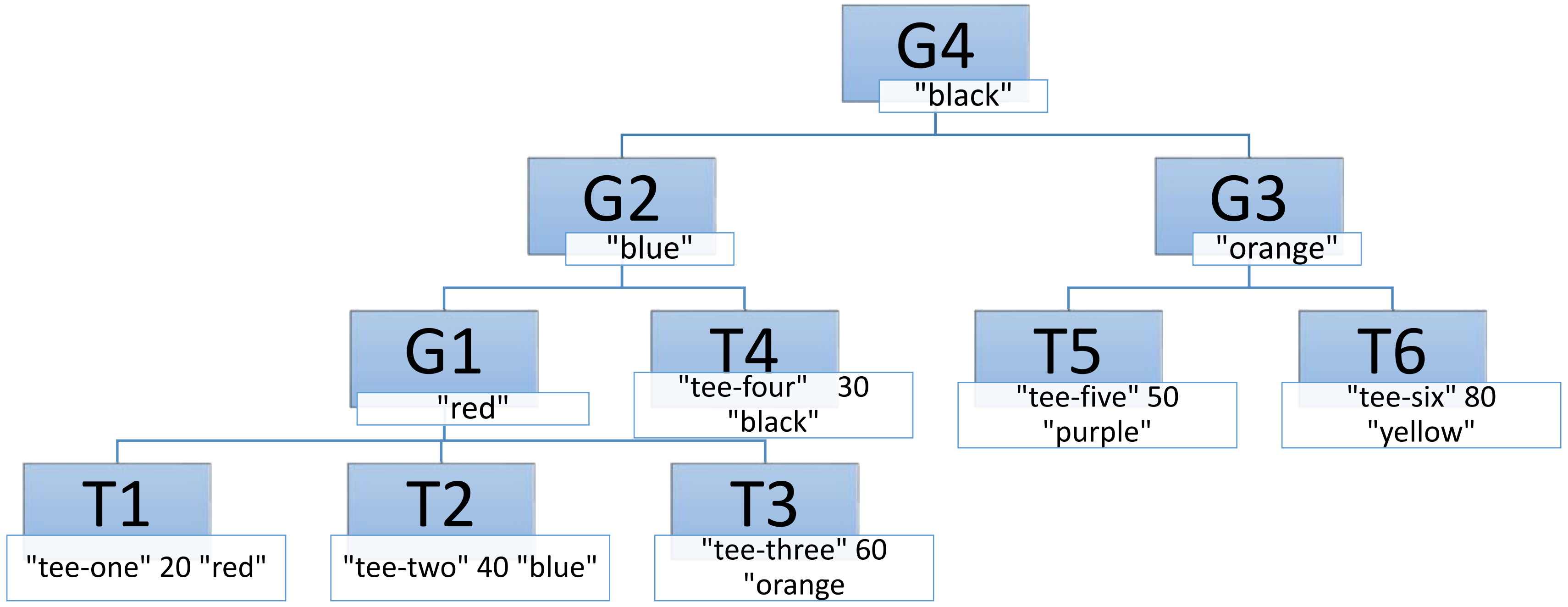
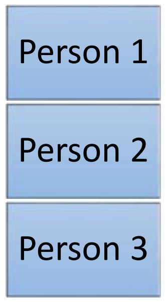
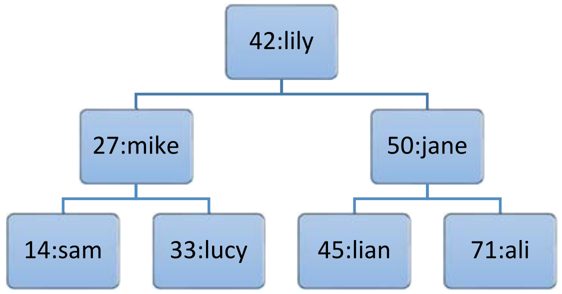
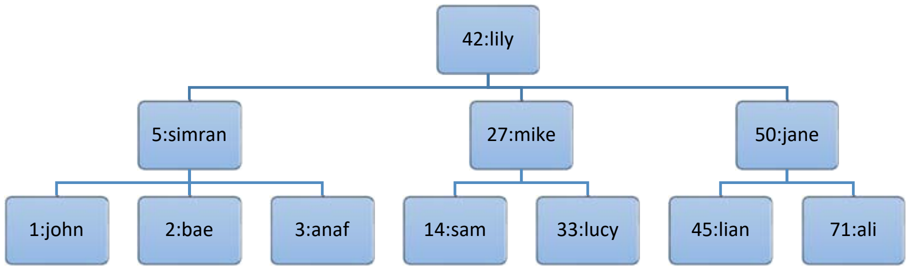

G4
"black"
G2
"blue"
G3
"orange"
G1
"red"
T4
"tee-four" 30 "black"
T5
"tee-five" 50 "purple"
T6
"tee-six" 80 "yellow"
T1
"tee-one" 20 "red"
T2
"tee-two" 40 "blue"
T3
"tee-three" 60 "orange







```
(define-struct person (name age))

;; Person is (make-person String Natural)

;; ListOfPerson is one of:
;; - empty
;; - (cons Person ListOfPerson)
```

```
(define-struct btree-node (k v l r))

;; BTree is one of:
;;  - false
;;  - (make-btree-node Integer String BTree BTree)
```

```
(define-struct ttree-node (k v l c r))

;; TTree is one of:
;;  - false
;;  - (make-ttree-node Integer String TTree TTree TTree)
```

```
(define-struct elt (name data subs))

;; Element is (make-elt String Integer ListOfElement)

;; ListOfElement is one of:
;;  - empty
;;  - (cons Element ListOfElement)
;; interp. A list of file system Elements
```

```
(define-struct terminal (label weight color))
(define-struct group (color subs))

;; Region is one of:
;;  - (make-terminal String Natural Color)
;;  - (make-group Color ListOfRegion)

;; ListOfRegion is one of:
;;  - empty
;;  - (cons Region ListOfRegion)
```

```
(define-struct terminal (label weight color))
(define-struct group (color subs))

;; Region is one of:
;;  - (make-terminal String Natural Color)
;;  - (make-group Color ListOfRegion)

;; ListOfRegion is one of:
;;  - empty
;;  - (cons Region ListOfRegion)
```

```
(define (fn-for-region r)
  (cond [(terminal? r)
         (... (terminal-label r)
              (terminal-weight r)
              (terminal-color r))]
        [(group? r)
         (... (terminal-color r)
              (fn-for-lor (group-subs r)))]))

(define (fn-for-lor lor)
  (cond [(empty? lor) (...)]
        [else
         (... (fn-for-region (first lor))
              (fn-for-lor (rest lor)))]))
```