Project 2 Documentation

Casey Burklow, Isaac Dudney

Data Structures

There are three data structures in the program:

1. `Person` - Class with four variables.
2. `Line` - A list of Persons in the line for the bathroom.
3. `Bathroom` - A list of Persons in the bathroom.

These three make up the majority of data manipulated in the program. There are other assorted state-based integers that help keep program logic clear and simple.

Structure

The code is split into a few state-managing data structures, utility functions, and the core logic.

The state management happens through four main variables and a `Person` struct. There's a time variable (`t`) to indicate when events should happen, a departed counter (`dep`) to ensure that the code ends when it needs to, and two lists for the `bathroom` and `line`. The person struct manages the things a person needs to know: their gender, the amount of time they spend in the bathroom, and the global time they must leave (calculated upon entering the bathroom).

The utility functions manage moving the people back and forth, essentially. There's `Arrive`, `UseFacilities`, and `Depart`. They do what they sound like: `Arrive` adds the people to the bathroom, `UseFacilities` removes those who are done using the bathroom, and `Depart` is a utility function to help `UseFacilities` do its job.

The core logic is very simple: add people to the line, use the facilities, add people to the bathroom, and increment the time by one.

Combined together, each piece of this structure fulfills every requirement in this project.

Psuedocode

```
### state
current_time = 0
departed = 0
line = []
bathroom = []

# class
Person(identifier):
        leave_time = 0
        in_time = random(3,7)
        identifier = identifier
        gender = random_weighted(60,0,1)

### functions

## utility
person_pop():
        set_leave_time(line.nextPerson)
        return(line.nextPerson.pop())
person_delete(index):
        set_leave_time(line[index])
        return(line.pop(index))

## main
# add to bathroom
Arrive():
        if bathroom is empty:
                bathroom.append(person_pop())

        for person in line:
                if person.gender == bathroom[0].gender and bathroom is not full:
                        bathroom.append(person_pop())
                else:
                        return
# remove from bathroom
```

```
36 UseFacilities():
37         if bathroom is empty:
38                 return
39         for person in bathroom:
40                 if person.leave_time == current_time
41                         departed += 1
42                         Depart(person)
43 # utility to remove from bathroom
44 Depart(Person):
45         bathroom.remove(Person)
46
47 ### main logic
48
49 # a groups of 5, b groups of 10, c 20 at once
50 input("Case␣a,␣b,␣c:␣")
51
52 case a:
53         if current_time = 0:
54                 line.add(5,Person)
55         else if current_time = 10:
56                 line.add(5,Person)
57         else if current_time = 20:
58                 line.add(5,Person)
59         else if current_time = 30:
60                 line.add(5,Person)
61
62         UseFacilities()
63         if bathroom is not full and line is not empty:
64                 Arrive()
65         current_time += 1
66 case b:
67         if current_time = 0:
68                 line.add(10,Person)
69         else if current_time = 10:
70                 line.add(10,Person)
71
```

```
72          UseFacilities()
73          if bathroom is not full and line is not empty:
74                  Arrive()
75          current_time += 1
76  case c:
77          if current_time = 0:
78                  line.add(20,Person)
79
80          UseFacilities()
81          if bathroom is not full and line is not empty:
82                  Arrive()
83          current_time += 1
```