Class 5: Designing Specifications

6.102 — Software Construction
Spring 2024

Warmup

Exercise:  yellkey.com/**stand**

Nanoquiz:  yellkey.com/**else**

```
function split(s: string, sep: string): Array<string>
// requires: sep.length = 1
// effects: returns array `v` such that ???
```

Open `warmup.ts` and critique each suggestion for part of the postcondition

Also open `clicker.mit.edu/6.102` for some clicker questions…

Warmup

Exercise: yellkey.com/**stand**

Nanoquiz: yellkey.com/**else**

```
/**
 * Splits a string into parts separated by a separator character
 * @param s    string to split
 * @param sep  separator to split on; requires sep.length = 1
 * @returns    a k-element array `v` such that
 *               text = v[0] + sep + v[1] + ... + sep + v[k-1]
 *
 */
function split(s: string, sep: string): Array<string>
```

Which of these input/output pairs is allowed by the spec above? (can pick more than one)

A. `split("ab--cd-ef", "-" ) → ["a", "b", "c", "d", "e", "f"]`
B. `split("ab--cd-ef", "-" ) → ["ab", "", "cd", "ef"]`
C. `split("ab--cd-ef", "-" ) → ["ab", "cd", "ef"]`
D. `split("ab--cd-ef", "-" ) → ["ab--cd", "ef"]`
E. none of the above

Warmup

Exercise: yellkey.com/**stand**

Nanoquiz: yellkey.com/**else**

```
/**
 * Splits a string into parts separated by a separator character
 * @param s    string to split
 * @param sep  separator to split on; requires sep.length = 1
 * @returns    a k-element array `v` such that
 *               text = v[0] + sep + v[1] + ... + sep + v[k-1]
 *             and no elements of `v` contain `sep`
 */
function split(s: string, sep: string): Array<string>
```

Which of these input/output pairs is allowed by the spec above? (can pick more than one)

A. `split("ab--cd-ef", "-" ) → ["a", "b", "c", "d", "e", "f"]`
B. `split("ab--cd-ef", "-" ) → ["ab", "", "cd", "ef"]`
C. `split("ab--cd-ef", "-" ) → ["ab", "cd", "ef"]`
D. `split("ab--cd-ef", "-" ) → ["ab--cd", "ef"]`
E. none of the above

Nanoquiz

- This quiz is just for you and your own brain:
  - closed-book, closed-notes
  - nothing else on your screen
- Lower your laptop screen when you're done


yellkey.com/**else**

Where we use specs

Criteria for comparing specs

```
/**
 * @param a != 0
 * @param b
 * @param c where b^2 - 4ac >= 0
 * @returns x such that ax^2 + bx + c = 0
 */
function quadraticRoot(a: number, b: number, c: number): number
```

This spec is: *(pick all that apply)*

deterministic
underdetermined
nondeterministic

operational
declarative

```
/**
 * @param a != 0
 * @param b
 * @param c where b^2 - 4ac >= 0
 * @returns x such that ax^2 + bx + c = 0
 */
```

```
/**
 * @param a != 0
 * @param b
 * @param c where b^2 - 4ac >= 0
 * @returns largest x such that ax^2 + bx + c = 0
 */
function quadraticRoot(a: number, b: number, c: number): number
```

This new spec is: *(pick all that apply)*

deterministic
underdetermined
nondeterministic

operational
declarative

```
/**
 * Solve a quadratic equation using Newton's method:
 *   - start by setting x = 1
 *   - compute correction = -(ax^2 + bx + c)/(2ax + b)
 *   - update x = x + correction
 *   - iterate on correction and x until |correction| < 0.000001
 * @param a != 0
 * @param b
 * @param c where b^2 - 4ac >= 0
 * @returns x as computed above
 */
function quadraticRoot(a: number, b: number, c: number): number
```

This new spec is: *(pick all that apply)*

deterministic
underdetermined
nondeterministic

operational
declarative

find-first

```
/**
 * Find the first occurrence of x in sorted array a.
 *
 * @param x value to find
 * @param a array sorted in increasing order,
 *              a[0] <= a[1] <= ... <= a[n-1]
 * @returns lowest i such that a[i]==x, or -1 if no such i
 */
function find(x: number, a: Array<number>): number
```

find-first

```
/**
 * Find the first occurrence of x in sorted array a.
 *
 * @param x value to find
 * @param a array sorted in increasing order,
 *              a[0] <= a[1] <= ... <= a[n-1]
 * @returns lowest i such that a[i]==x, or -1 if no such i
 */
function find(x: number, a: Array<number>): number
```

find-any

```
/**
 * Find any occurrence of x in sorted array a.
 *
 * ... (same as above)
 * @returns some i such that a[i]==x, or -1 if no such i
 */
function find(x: number, a: Array<number>): number
```

find-first

```
/**
 * Find the first occurrence of x in sorted array a.
 * @param x value to find
 * @param a array sorted in increasing order
 * @returns lowest i such that a[i]==x, or -1 if no such i
 */
```

find-any

```
/**
 * Find any occurrence of x in sorted array a.
 * ... (same as above)
 * @returns some i such that a[i]==x, or -1 if no such i
 */
```

find-once

```
/**
 * Find x that occurs exactly once in sorted array a.
 * @param x value to find; required to appear exactly once in a
 * @param a array sorted in increasing order
 * @returns the index i such that a[i] == x
 */
```

find-first

```
/**
 * Find the first occurrence of x in sorted array a.
 *
 * @param x value to find
 * @param a array sorted in increasing order
 * @returns lowest i such that a[i]==x, or -1 if no such i
 */
```

find-any-unsorted

```
/**
 * Find any occurrence of x in any array a.
 *
 * @param x value to find
 * @param a array (in any order)
 * @returns some i such that a[i]==x, or -1 if no such i
 */
```

find-first

```
/**
 * Find the first occurrence of x in sorted array a.
 *
 * @param x value to find
 * @param a array sorted in increasing order
 * @returns lowest i such that a[i]==x, or -1 if no such i
 */
```

find-any-throw

```
/**
 * Find any occurrence of x in sorted array a.
 *
 * ... (same as above)
 * @returns some i such that a[i]==x
 * @throws NotFoundError if no such i
 */
```

Exercise:  yellkey.com/**stand**

Writing specs

Open `specs.ts`

For each method:

`getIgnoreCase()`
`increment()`

- fill in *requires* and *effects* (precondition and postcondition)
  - so that **both** pseudocode implementations satisfy the spec
  - and spec is reasonable for a client too (if perhaps not ideal)