Equivalent Class Partitioning: An Example

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

March 14, 2023

Exercise: Design Test Cases

`Suppose that you are given a programming problem to determine whether a triangle is Equilateral (and valid) given 3 sides. We shall design/writing test cases before any code.`

Complete the following:

1. Design a function/method prototype/header for a method that tells us whether a triangle is Equilateral (and valid) given 3 sides.
2. Design test cases for the function/method. Express the test case as a tuple for the Software-Under-Testing (SUT, the method/function) `(test input, expected output)`
3. Perform *equivalence class partitioning*, and create all necessary test cases.

Equivalence Class Partitioning

Equivalence class partitioning is a black-box testing strategy, and is based on specification.

Equivalence Class is a concept in Set Theory (Discrete Mathematics). The method is partitioning test cases into multiple disjoint sets, each is an equivalence class.

The objectives are

- to have a sense of "complete" testing, and
- to avoid redundancy.

Identifying the equivalence classes is still a heuristic process.

Equivalence Class Partitioning: Heuristics

Identifying the equivalence classes is still a heuristic process.

1. If an input condition specifies a range of values, identify one valid equivalence class and two invalid equivalence classes.

A student receives "A" with scors $x \in [90, 100]$: valid class $\{x|90 \leq x \leq 100\}$, two invalid classes $\{x|x < 90\}$, $\{x|x > 100\}$

2. If an input condition specifies the number of values, identify one valid equivalence class and two invalid equivalence classes.

One or two instructors can be listed for a course section: valid class: x instructors listed where $x \in \{1, 2\}$; two invalid classes: no instructors listed $\{x|x = 0\}$, more than 2 instructors listed $\{x|x > 2 \wedge x \in \mathbb{Z}\}$.

Equivalence Class Partitioning: Heuristics

3. If an input condition specifies a set of input values, and there is reason to believe that the program handles each differently, identify a valid equivalence class for each and one invalid equivalence class.

Courses offered can be in-person, online, or hybrid: valid classes: {in-person class}, {online class}, {hybrid class}, invalid class: {undertermined}

4. If an input condition specifies a "must-be" situation ("must-be", "shall-be", "should-be"), identify one valid equivalence class and one invalid equivalence class.

The first character of a Java class name must be an uppercase letter: valid class: $\{x | x$ is a class name $\wedge\, x$ begins with an uppercase letter$\}$, invalid class:
$\{x | x$ is not a class name $\vee\, x$ does not begin with an uppercase letter$\}$

Equivalence Class Partitioning: Heuristics

5. If there is any reason to believe that the program does not handle elements in an equivalence class identically, split the equivalence class into smaller equivalence classes.

Exercise: Design Test Cases: Sample Solution I

`Suppose that you are given a programming problem to determine whether a triangle is Equilateral (and valid) given 3 sides. We shall design/writing test cases before any code.`

Complete the following:

1. Design a function/method prototype/header for a method that tells us whether a triangle is Equilateral (and valid) given 3 sides.
   `boolean isEquilateral(double x, double y, double z)`
2. Design test cases for the function/method. Express the test case as a tuple for the Software-Under-Testing (SUT, the method/function)
   `(test input, expected output)`
3. Perform *equivalence class partitioning*, and create all necessary test cases.

Exercise: Design Test Cases: Sample Solution II

```
boolean isEquilateral(double x, double y, double z)
```

- x: $\{x|0 \le x \le \infty\}$, $\{x|x < 0\}$, $\{x \text{ is } \infty\}$
- y: $\{y|0 \le y \le \infty\}$, $\{y|y < 0\}$, $\{y \text{ is } \infty\}$
- z: $\{z|0 \le z \le \infty\}$, $\{z|z < 0\}$, $\{z \text{ is } \infty\}$

Exercise: Design Test Cases: Sample Solution III

```
boolean isEquilateral(double x, double y, double z)
```

- $\mathbb{X}$: $\{\{x|0 \leq x < \infty\}, \{x|x < 0\}, \{x \text{ is } \infty\}\}$
- $\mathbb{Y}$: $\{\{y|0 \leq y < \infty\}, \{y|y < 0\}, \{y \text{ is } \infty\}\}$
- $\mathbb{Z}$: $\{\{z|0 \leq z < \infty\}, \{z|z < 0\}, \{z \text{ is } \infty\}\}$

Perform a Caretsion product of the sets

$\mathbb{X} \times \mathbb{Y} \times \mathbb{Z}$

Exercise: Design Test Cases: Sample Solution IV

```
boolean isEquilateral(double x, double y, double z)
```

- $\mathbb{X}$: $\{\{x|0 \le x < \infty\}, \{x|x < 0\}, \{x \text{ is } \infty\}\}$
- $\mathbb{Y}$: $\{\{y|0 \le y < \infty\}, \{y|y < 0\}, \{y \text{ is } \infty\}\}$
- $\mathbb{Z}$: $\{\{z|0 \le z < \infty\}, \{z|z < 0\}, \{z \text{ is } \infty\}\}$

Perform a Caretsion product of the sets

$\mathbb{X} \times \mathbb{Y} \times \mathbb{Z}$

Is there any reason to believe that the program does not handle elements in an equivalence class identically?

Exercise: Design Test Cases: Sample Solution V

Is there any reason to believe that the program does not handle elements in an equivalence class identically? How about

$\{\{x|0 \leq x < \infty\}, \{\{y|0 \leq y < \infty\}, \{\{z|0 \leq z < \infty\}\}$

Exercise: Design Test Cases: Sample Solution VI

Is there any reason to believe that the program does not handle elements in an equivalence class identically? How about

$\{\{x|0 \leq x < \infty\}, \{\{y|0 \leq y < \infty\}, \{\{z|0 \leq z < \infty\}\}$

- the class of (x, y, z) that makes a valid triangle
- the class of (x, y, z) that makes an invalid triangle

Exercise: Design Test Cases: Sample Solution VI

Is there any reason to believe that the program does not handle elements in an equivalence class identically? How about

$\{\{x|0 \leq x < \infty\}, \{\{y|0 \leq y < \infty\}, \{\{z|0 \leq z < \infty\}\}$

1. the class of (x, y, z) that makes valid triangles
   1.1 the class of (x, y, z) that makes Equilateral triangles
   1.2 the class of (x, y, z) that makes non-Equilateral triangles
2. the class of (x, y, z) that makes an invalid triangle
   2.1 Thinking about how the program may check whether if it is a valid triangle, we can consider a special case where there might be an overflow, e.g.,

```
if (x+y < z) return false;
```

Exercise: Design Test Cases: Sample Solution VII

To give concrete test cases, sample each class for test cases (at least 1 each), e.g., in the format of `((x, y, z), r)`

```
((-1, 1, 1), false)
((-1, -1, 1), false)
((-1, -1, Double.POSITIVE_INFINITY), false)
((-1, -1, Double.NEGATIVE_INFINITY), false)
...
((1, 1, 1), true)
((3, 4, 5), false)
((3, 3, 7), false)
((3, 7, 3), false)
((7, 3, 3), false)
((Double.MAX_VALUE, Double.MAX_VALUE,
                    Double.MAX_VALUE), true)
```