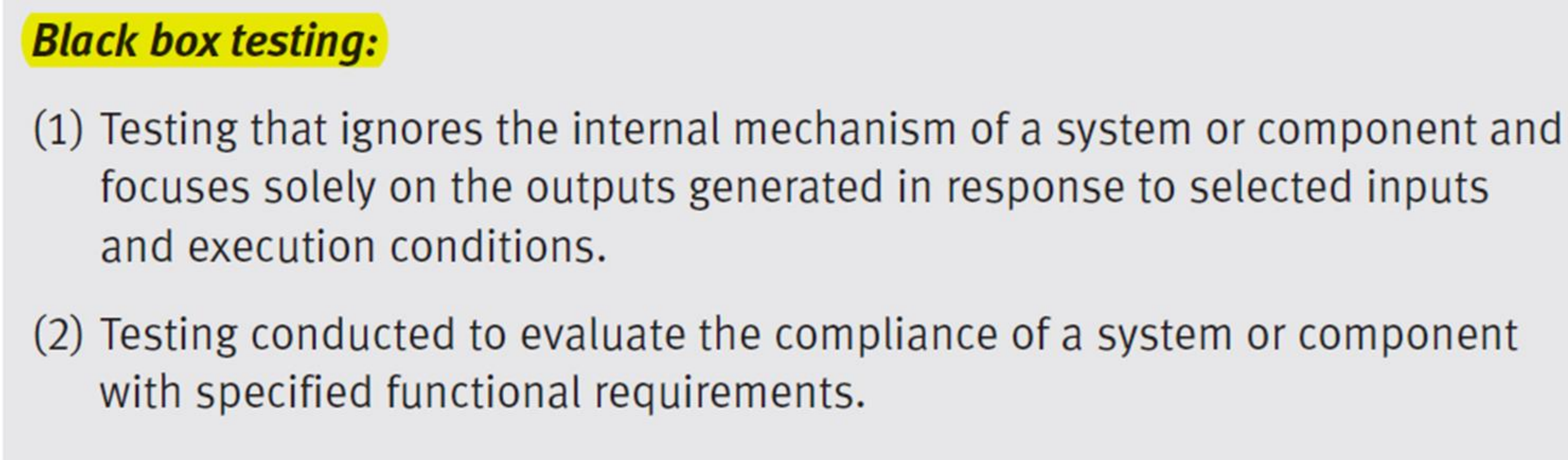
# Black-Box Testing Methods

Software QA, Birzeit University - Dr. Samer Zein

# Black-Box Testing: Intro

- Think of its techniques as a complement to White-Box testing techniques, not a substitute.



**Black box testing:**

(1) Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.

(2) Testing conducted to evaluate the compliance of a system or component with specified functional requirements.

# Black-Box Testing: Equivalence Classes Technique

- The focus here is to generate a set of test cases.
- The set has to be chosen carefully.
- Equivalence classes techniques aims at:
  - ▫ **A) Increasing the efficiency of testing**
  - ▫ **B) Minimizing the number of test cases**.
- That is, improved choice of test cases by efficient use of equivalence class partitioning

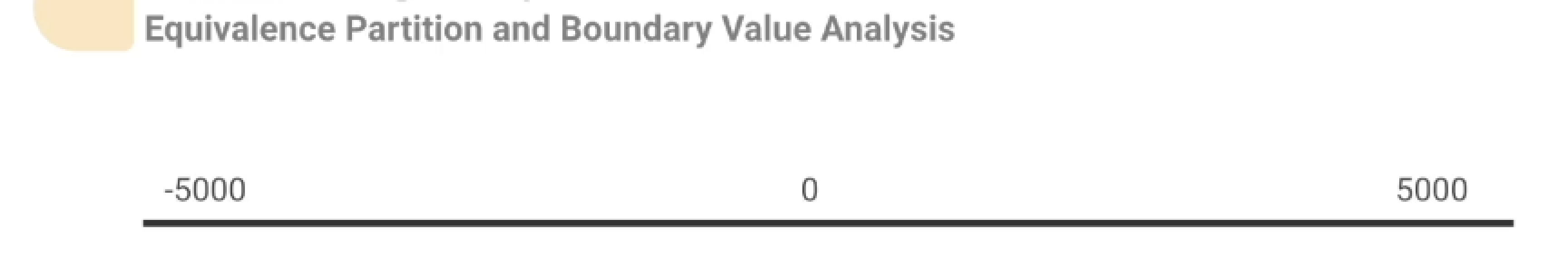
# Black-Box Testing: Equivalence Classes Technique..2

- What is an Equivalent Class (EC)?
- **It is a set of input variable values that are processed identically to produce the same output.**
- We also have:
  - **Valid EC**
  - **Invalid EC**
- Always remember that your test suit should contain both.

# Black-Box Testing: Equivalence Classes Technique..3

- The basic rule is ***that there should be at least one test case for each valid and invalid EC.***
- Thus, minimizing the number of test cases.
- It is your job to define the EC and their boundaries.
- **EC technique is based on requirements' specs, not code**.
- EC technique is much efficient from automated and random techniques, **Why?**

# Example:



Equivalence Partition and Boundary Value Analysis

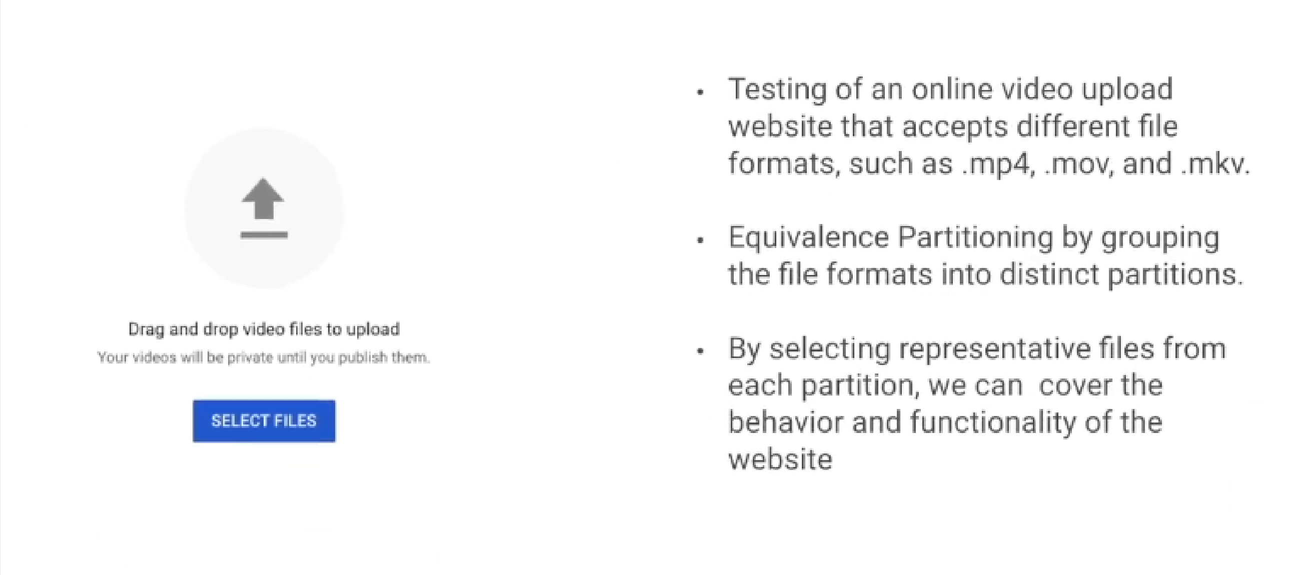
TC 1: -100

TC 2: 0

TC 3: 100

Focus on behaviour and response of the system

# Example 2:



Drag and drop video files to upload

Your videos will be private until you publish them.

- Testing of an online video upload website that accepts different file formats, such as .mp4, .mov, and .mkv.
- Equivalence Partitioning by grouping the file formats into distinct partitions.
- By selecting representative files from each partition, we can  cover the behavior and functionality of the website

Create a TC for each file type

# The General Process of Identifying EC

- A) If an input condition specifies a range of values (e.g., “the item count can be from 1 to 999”) then:

  - **identify one valid equivalence class (1<item count<999) and**
  - **two invalid equivalence classes (item count<1 and item count>999).**

# The General Process of Identifying EC ..2

- B) If an input condition specifies the number of values

(e.g., “one through six owners can be listed for the automobile”),

- **identify one valid equivalence class**
- **and two invalid equivalence classes (no owners and more than six owners).**

# The General Process of Identifying EC ...3

- C) If an input condition specifies a set of input values, **and there is reason to believe that the program handles each differently**

  eg.("type of vehicle must be BUS, TRUCK, TAXICAB, PASSENGER, or MOTORCYCLE"),

  - **identify a valid equivalence class for each**
  - **and one invalid equivalence class ("TRAILER," for example).**

# The General Process of Identifying EC…4

- D) If an input condition specifies a “must-be” situation,

  such as “first character of the identifier must be a letter,”

  - **identify one valid equivalence class (it is a letter)**
  - **and one invalid equivalence class (it is not a letter).**

# Important Note Regarding Invalid EC

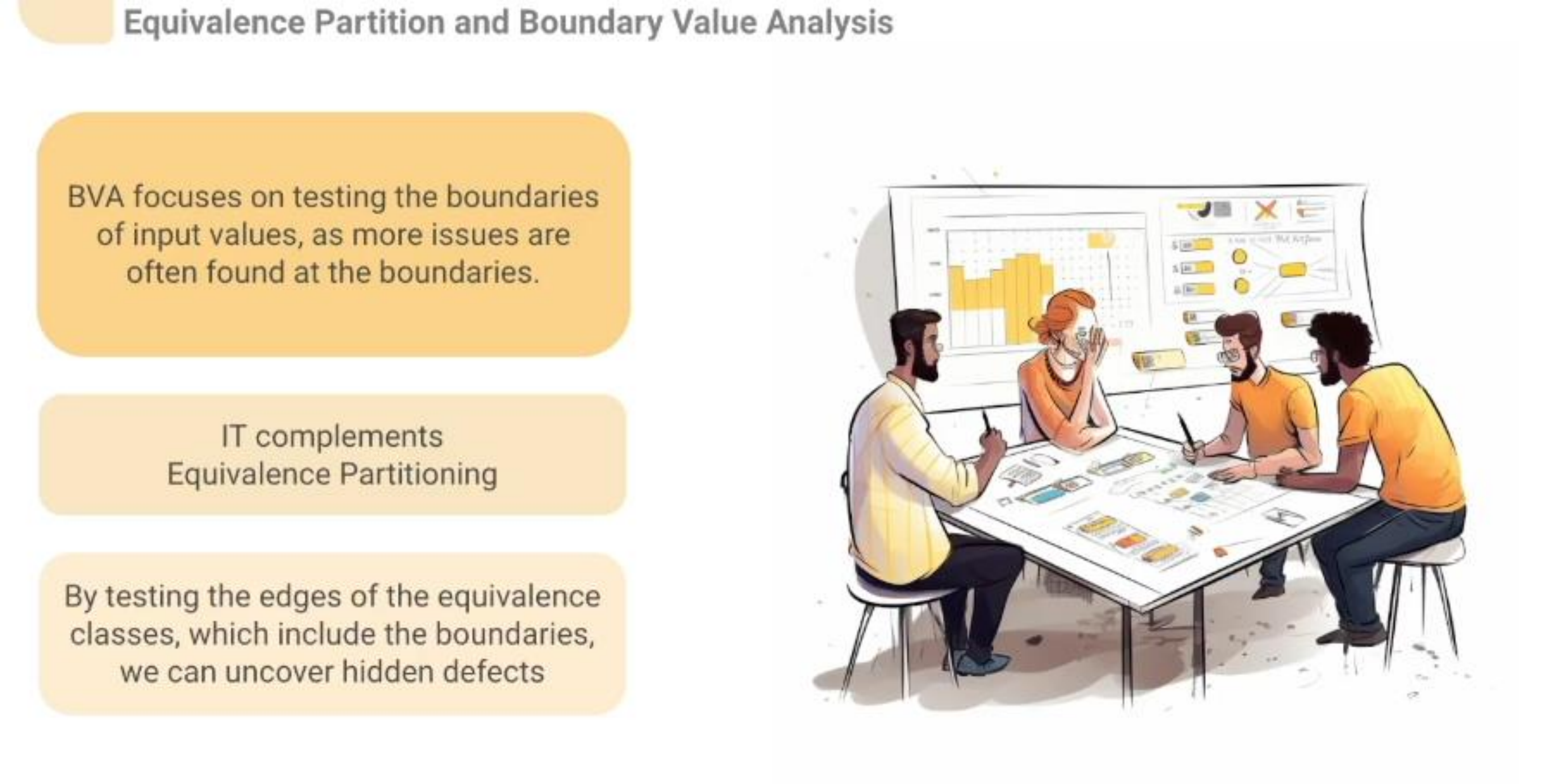
- **if the specification states "enter book type (HARDCOVER, SOFTCOVER, or LOOSE) and amount (1–999),"**
-

the test case, **(XYZ, 0),** expressing two error conditions (invalid book type and amount)

- will probably not exercise the check for the amount, since the program may say "XYZ IS UNKNOWN BOOK TYPE" and not bother to examine the remainder of the input.

# EC’s Boundary Values

- It is additional criterion that you should be aware of during assigning test cases for each EC.
- That is, **pay great attention to defining test cases to test the boundaries of each EC**
- More specifically, we need three test cases:
  - □ **Middle range**
  - □ **Lower bound**
  - □ **Upper Bound**



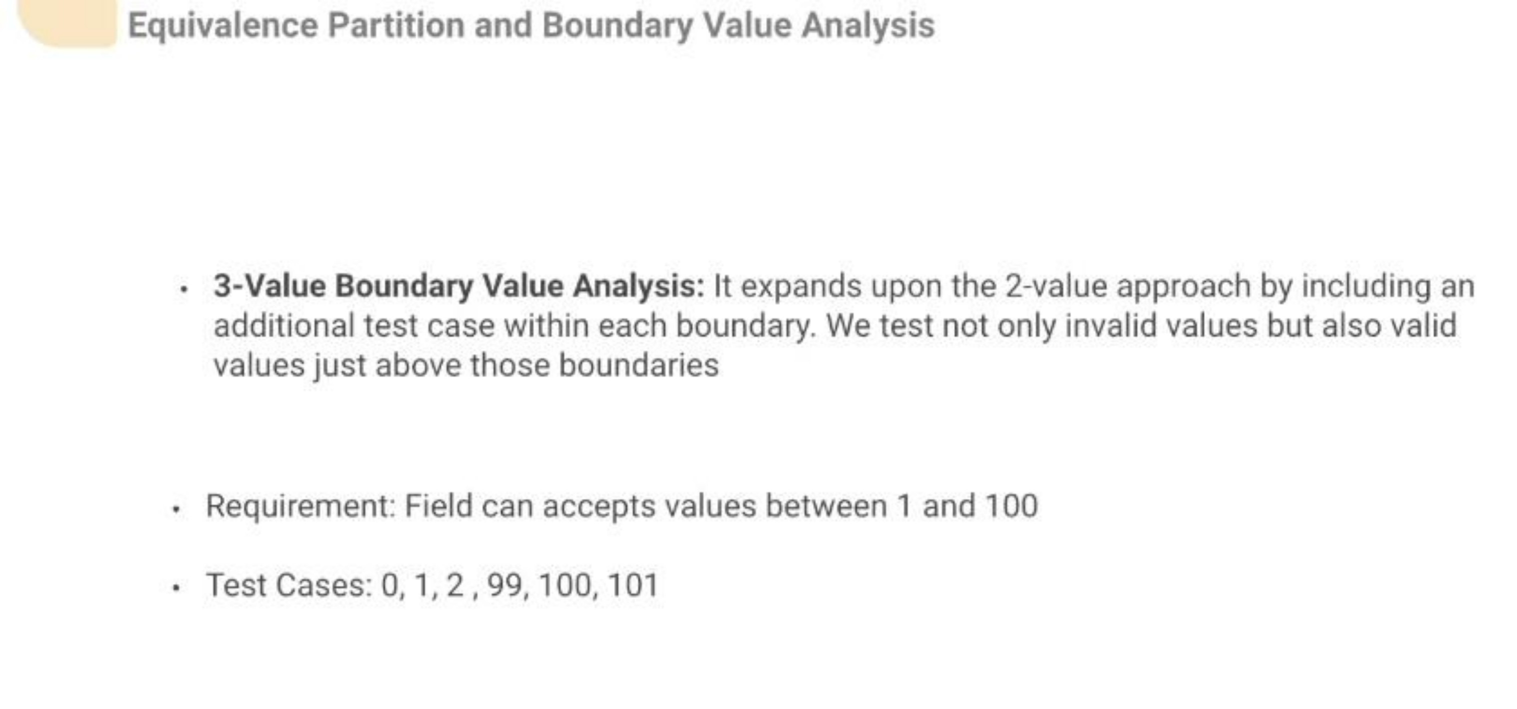
## Equivalence Partition and Boundary Value Analysis

BVA focuses on testing the boundaries of input values, as more issues are often found at the boundaries.

IT complements Equivalence Partitioning

By testing the edges of the equivalence classes, which include the boundaries, we can uncover hidden defects

# Example:

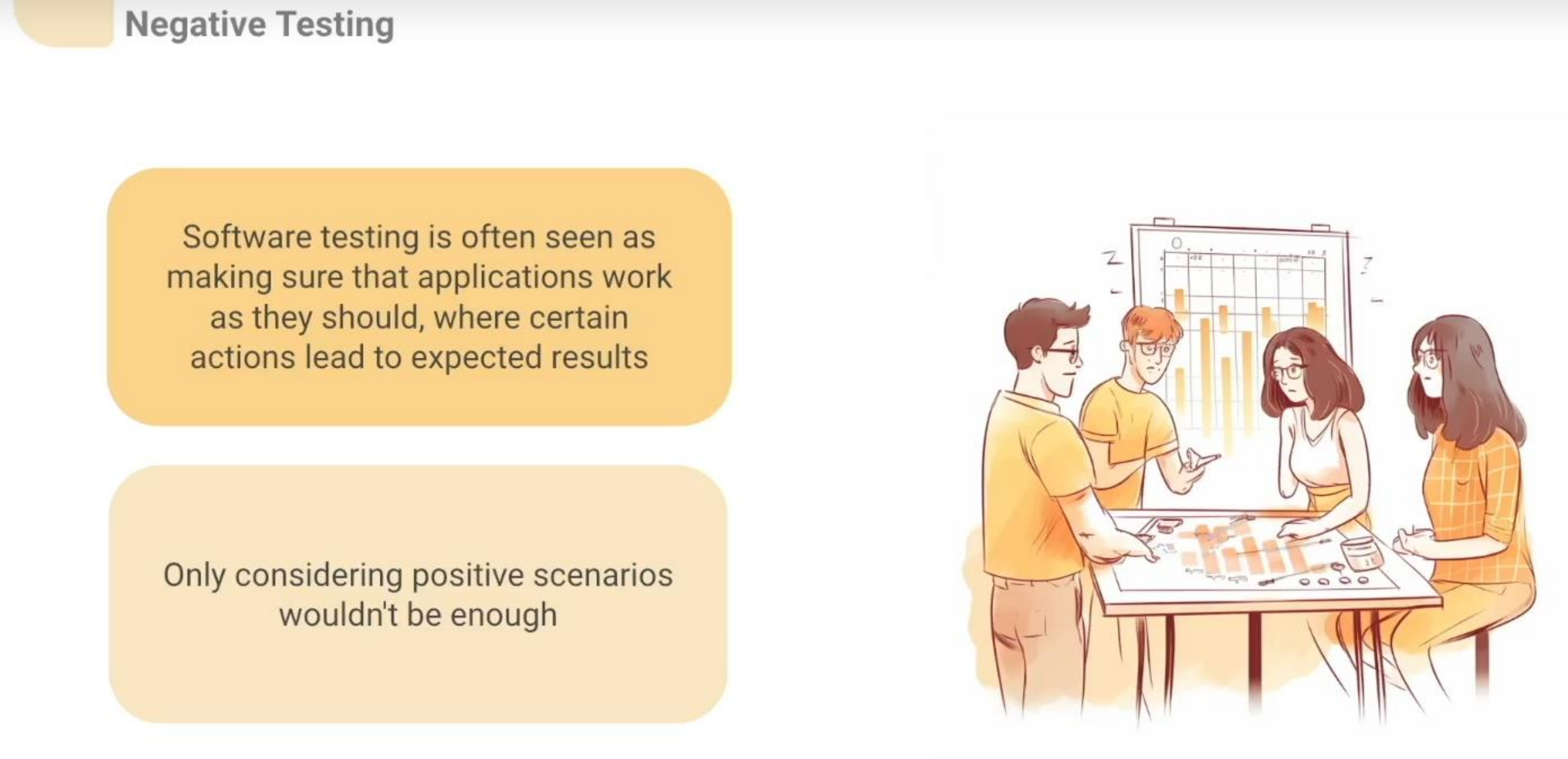


Equivalence Partition and Boundary Value Analysis

- **3-Value Boundary Value Analysis:** It expands upon the 2-value approach by including an additional test case within each boundary. We test not only invalid values but also valid values just above those boundaries

- Requirement: Field can accepts values between 1 and 100

- Test Cases: 0, 1, 2 , 99, 100, 101



## Negative Testing

Software testing is often seen as making sure that applications work as they should, where certain actions lead to expected results

Only considering positive scenarios wouldn't be enough

# Negative Testing

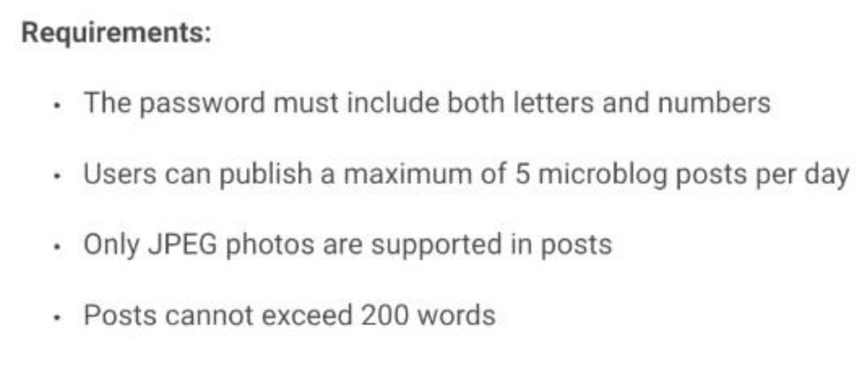
Negative testing ensures that an application can gracefully handle invalid or unexpected inputs, preventing crashes and incorrect behavior

The purpose of negative testing is to safeguard applications from unforeseen issues

# Example: Registration Form

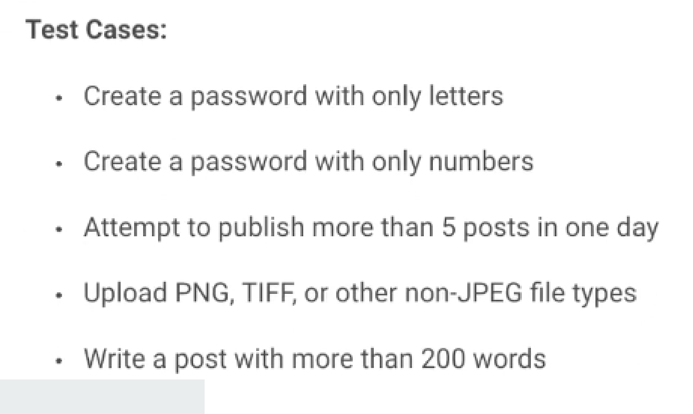
Some applications and web pages have fields that are marked as mandatory. To test the behavior of such fields, we can create tests that leave the required fields empty and analyse the application's response

# Another Example Based on requirements



**Requirements:**

- The password must include both letters and numbers
- Users can publish a maximum of 5 microblog posts per day
- Only JPEG photos are supported in posts
- Posts cannot exceed 200 words

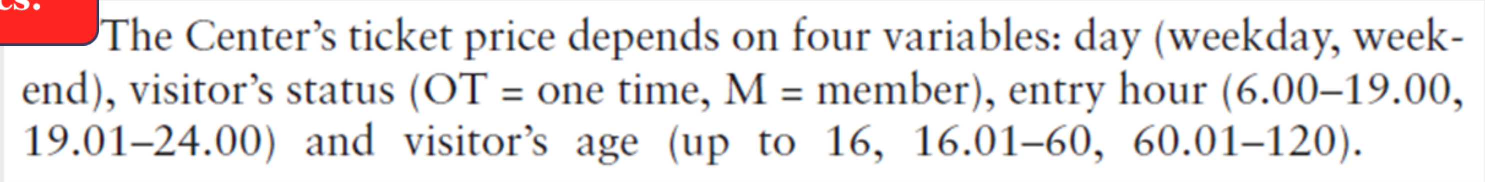


**Test Cases:**

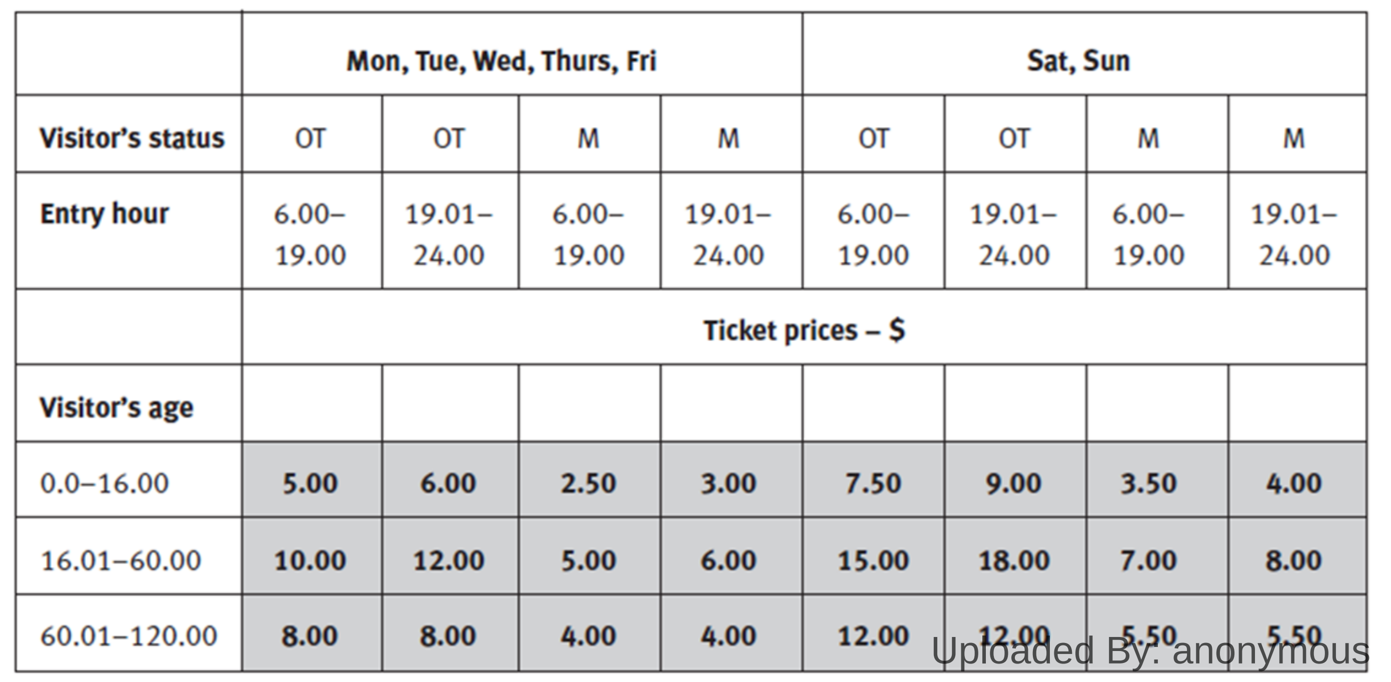
- Create a password with only letters
- Create a password with only numbers
- Attempt to publish more than 5 posts in one day
- Upload PNG, TIFF, or other non-JPEG file types
- Write a post with more than 200 words

# In Class Activity: EC Testing technique

**Requirements Specs:**



The Center's ticket price depends on four variables: day (weekday, weekend), visitor's status (OT = one time, M = member), entry hour (6.00–19.00, 19.01–24.00) and visitor's age (up to 16, 16.01–60, 60.01–120).



| | Mon, Tue, Wed, Thurs, Fri | | | | Sat, Sun | | | |
|---|---|---|---|---|---|---|---|---|
| **Visitor's status** | OT | OT | M | M | OT | OT | M | M |
| **Entry hour** | 6.00–19.00 | 19.01–24.00 | 6.00–19.00 | 19.01–24.00 | 6.00–19.00 | 19.01–24.00 | 6.00–19.00 | 19.01–24.00 |
| | **Ticket prices – $** | | | | | | | |
| **Visitor's age** | | | | | | | | |
| 0.0–16.00 | **5.00** | **6.00** | **2.50** | **3.00** | **7.50** | **9.00** | **3.50** | **4.00** |
| 16.01–60.00 | **10.00** | **12.00** | **5.00** | **6.00** | **15.00** | **18.00** | **7.00** | **8.00** |
| 60.01–120.00 | **8.00** | **8.00** | **4.00** | **4.00** | **12.00** | **12.00** | **5.50** | **5.50** |

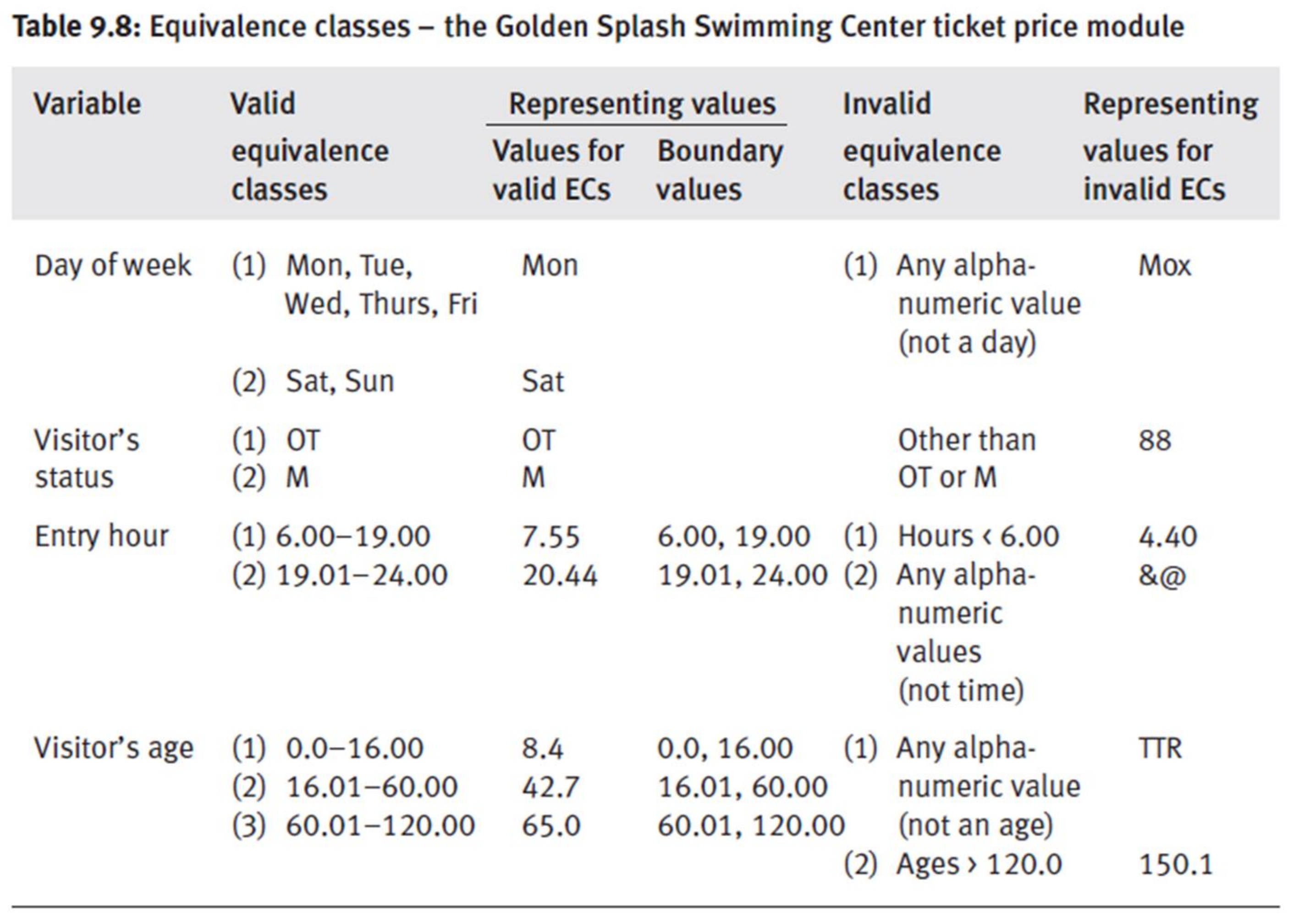
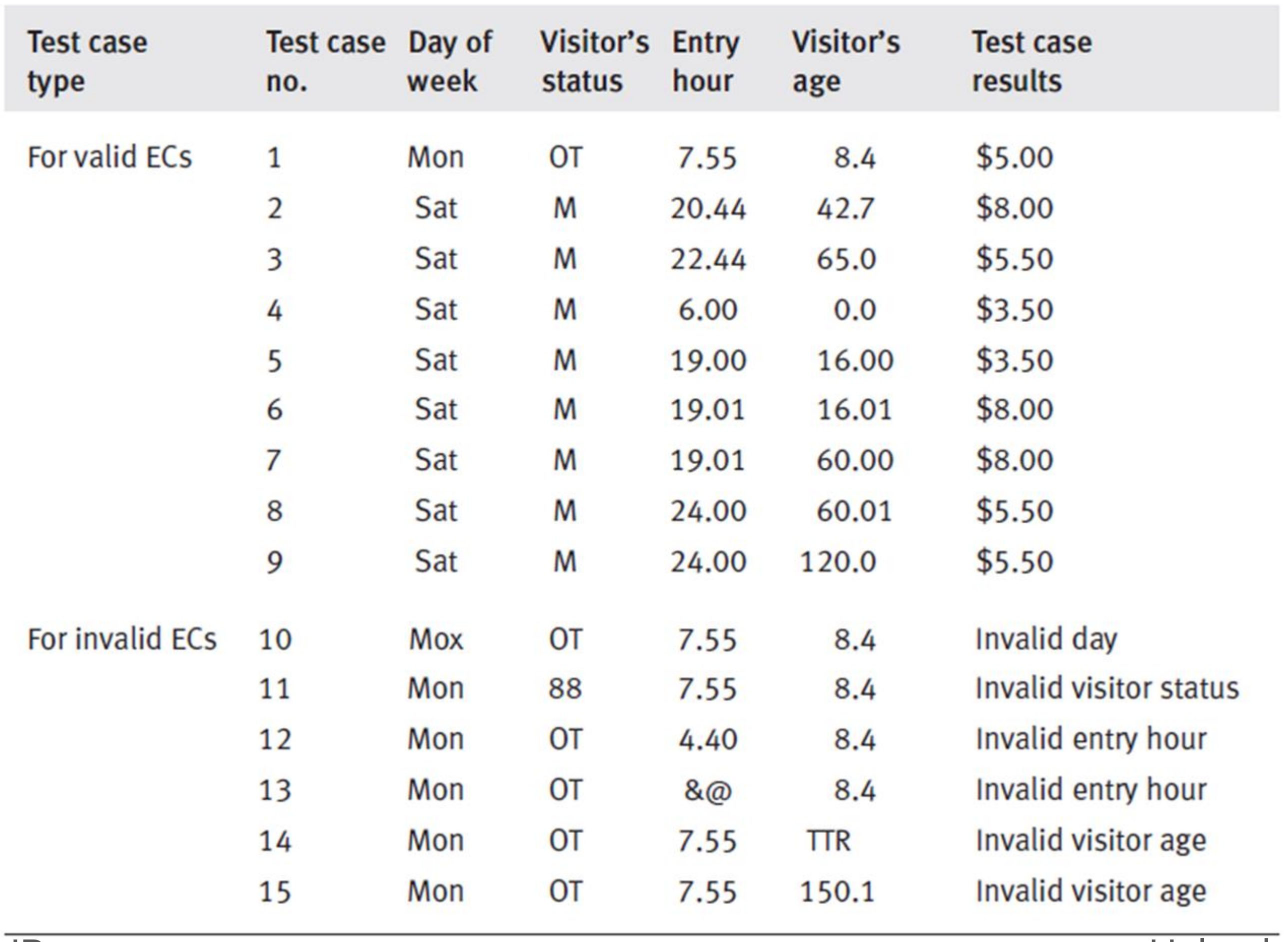


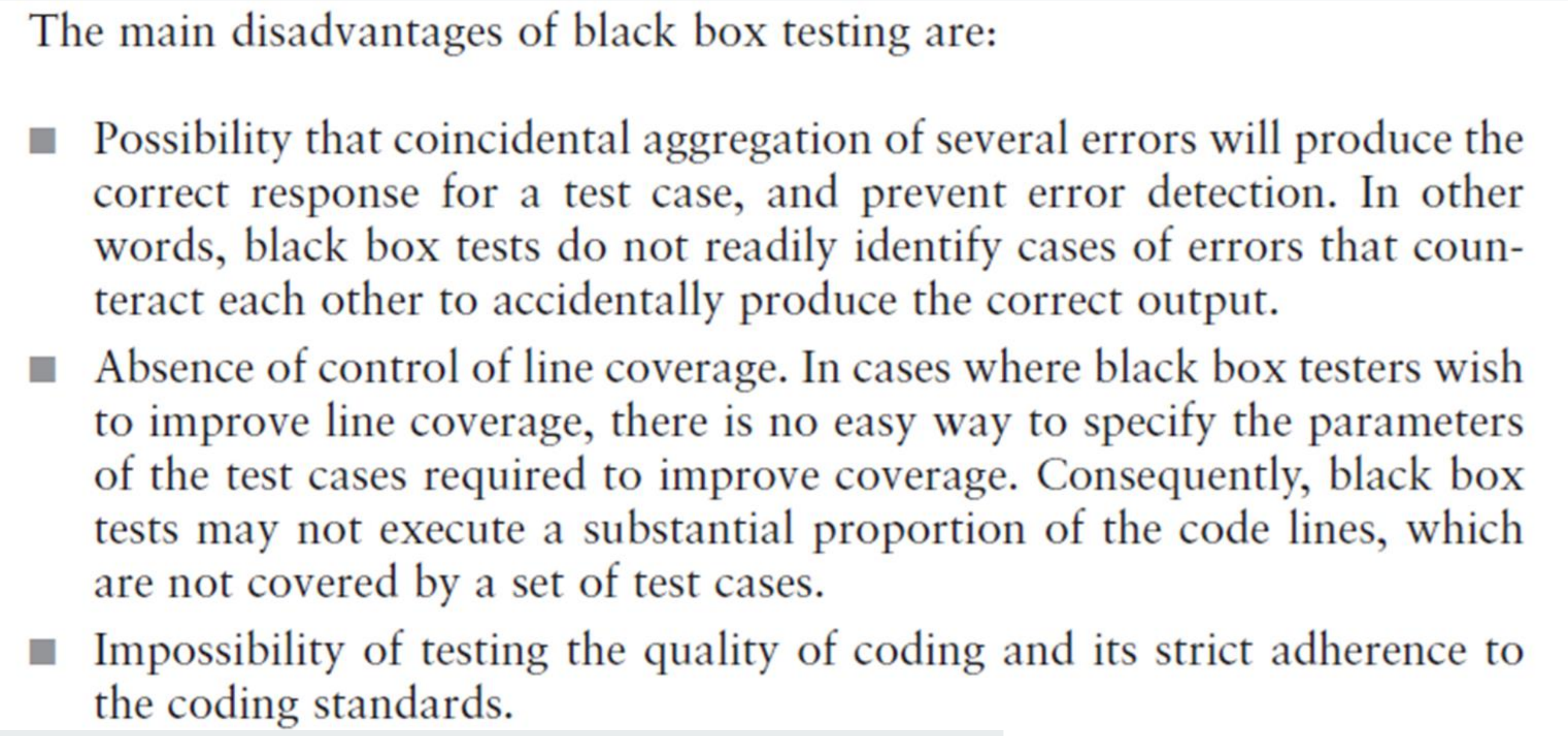
**Table 9.8:** Equivalence classes – the Golden Splash Swimming Center ticket price module

| Variable | Valid equivalence classes | Representing values | | Invalid equivalence classes | Representing values for invalid ECs |
|---|---|---|---|---|---|
| | | Values for valid ECs | Boundary values | | |
| Day of week | (1) Mon, Tue, Wed, Thurs, Fri | Mon | | (1) Any alpha-numeric value (not a day) | Mox |
| | (2) Sat, Sun | Sat | | | |
| Visitor's status | (1) OT<br>(2) M | OT<br>M | | Other than OT or M | 88 |
| Entry hour | (1) 6.00–19.00 | 7.55 | 6.00, 19.00 | (1) Hours < 6.00 | 4.40 |
| | (2) 19.01–24.00 | 20.44 | 19.01, 24.00 | (2) Any alpha-numeric values (not time) | &@ |
| Visitor's age | (1) 0.0–16.00 | 8.4 | 0.0, 16.00 | (1) Any alpha-numeric value (not an age) | TTR |
| | (2) 16.01–60.00 | 42.7 | 16.01, 60.00 | | |
| | (3) 60.01–120.00 | 65.0 | 60.01, 120.00 | | |
| | | | | (2) Ages > 120.0 | 150.1 |

Table 9.9: Test cases – the Golden Splash Swimming Center ticket price module



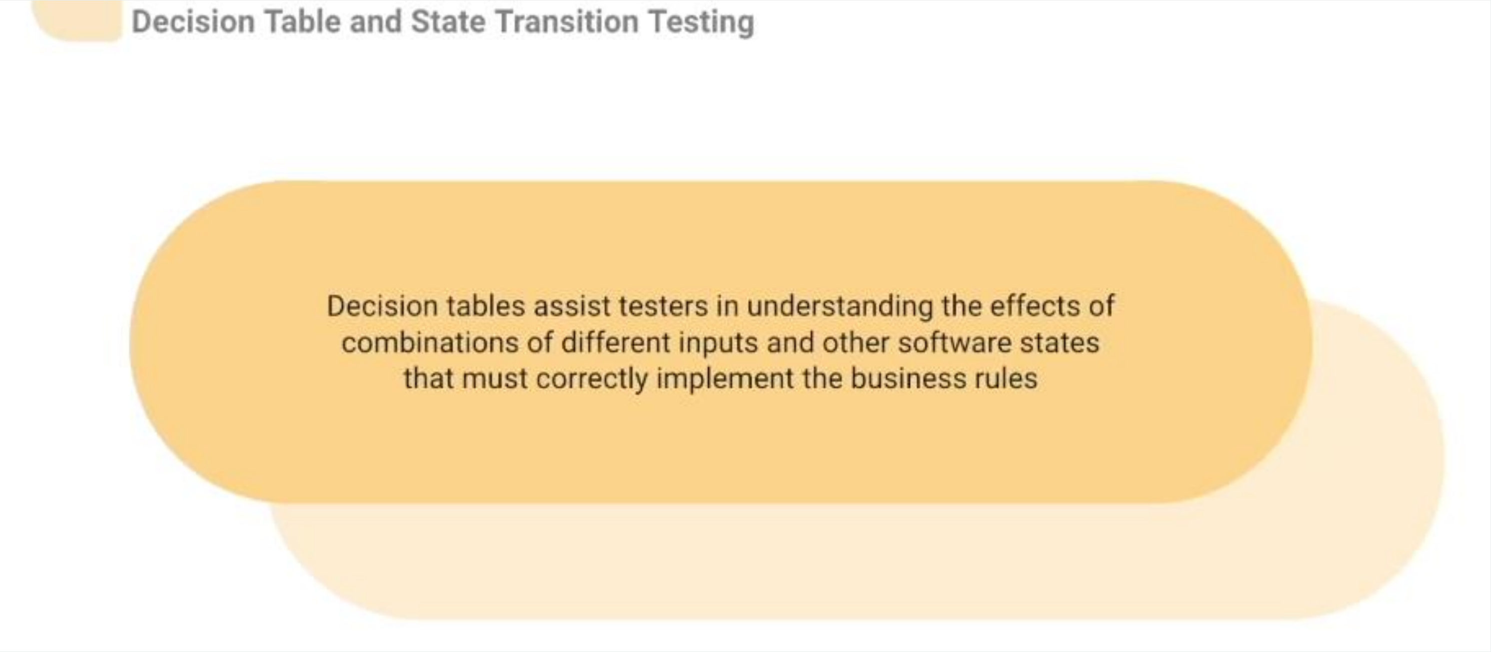
| Test case type | Test case no. | Day of week | Visitor's status | Entry hour | Visitor's age | Test case results |
|---|---|---|---|---|---|---|
| For valid ECs | 1 | Mon | OT | 7.55 | 8.4 | $5.00 |
| | 2 | Sat | M | 20.44 | 42.7 | $8.00 |
| | 3 | Sat | M | 22.44 | 65.0 | $5.50 |
| | 4 | Sat | M | 6.00 | 0.0 | $3.50 |
| | 5 | Sat | M | 19.00 | 16.00 | $3.50 |
| | 6 | Sat | M | 19.01 | 16.01 | $8.00 |
| | 7 | Sat | M | 19.01 | 60.00 | $8.00 |
| | 8 | Sat | M | 24.00 | 60.01 | $5.50 |
| | 9 | Sat | M | 24.00 | 120.0 | $5.50 |
| For invalid ECs | 10 | Mox | OT | 7.55 | 8.4 | Invalid day |
| | 11 | Mon | 88 | 7.55 | 8.4 | Invalid visitor status |
| | 12 | Mon | OT | 4.40 | 8.4 | Invalid entry hour |
| | 13 | Mon | OT | &@ | 8.4 | Invalid entry hour |
| | 14 | Mon | OT | 7.55 | TTR | Invalid visitor age |
| | 15 | Mon | OT | 7.55 | 150.1 | Invalid visitor age |



The main disadvantages of black box testing are:

- Possibility that coincidental aggregation of several errors will produce the correct response for a test case, and prevent error detection. In other words, black box tests do not readily identify cases of errors that counteract each other to accidentally produce the correct output.
- Absence of control of line coverage. In cases where black box testers wish to improve line coverage, there is no easy way to specify the parameters of the test cases required to improve coverage. Consequently, black box tests may not execute a substantial proportion of the code lines, which are not covered by a set of test cases.
- Impossibility of testing the quality of coding and its strict adherence to the coding standards.

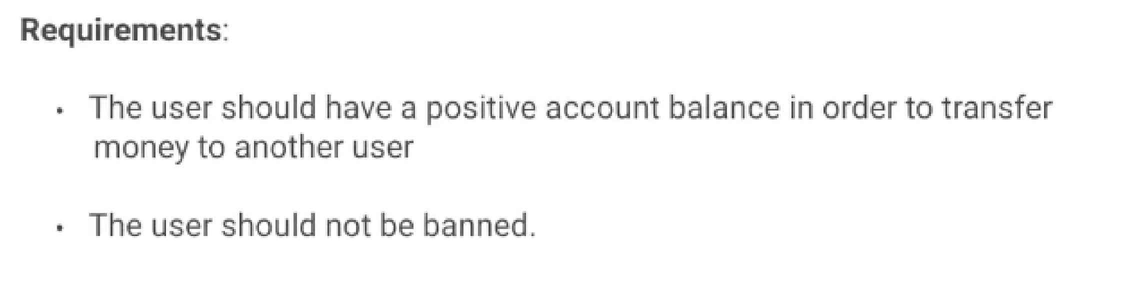
# Decision Table Testing



Decision Table and State Transition Testing

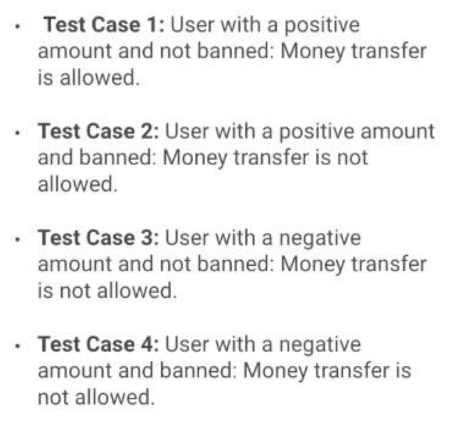
Decision tables assist testers in understanding the effects of combinations of different inputs and other software states that must correctly implement the business rules

# Example:

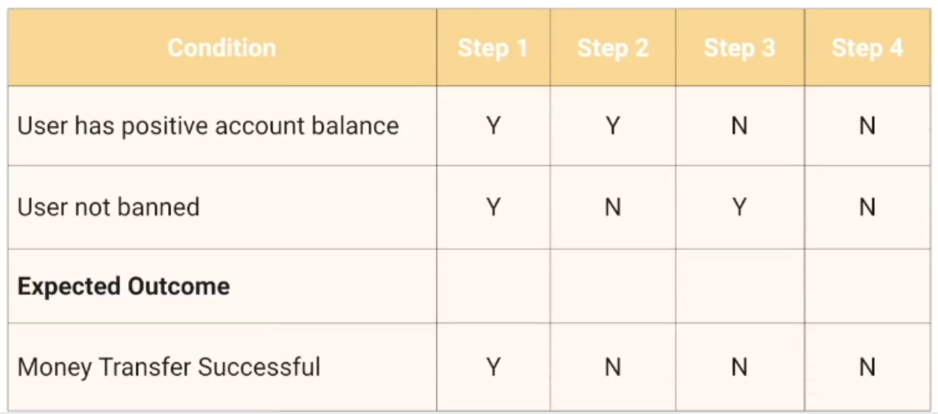


**Requirements**:

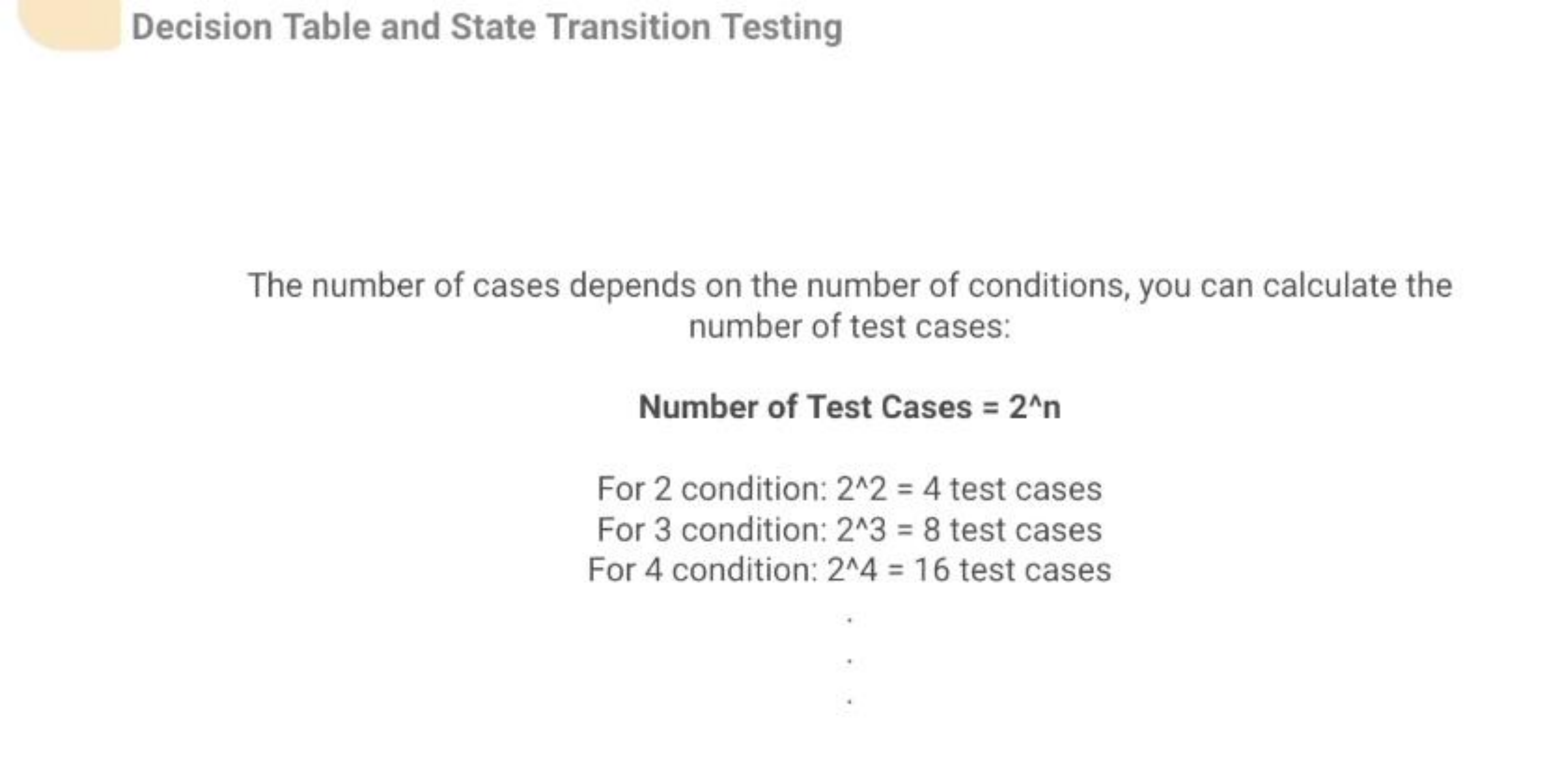
- The user should have a positive account balance in order to transfer money to another user
- The user should not be banned.



- **Test Case 1:** User with a positive amount and not banned: Money transfer is allowed.
- **Test Case 2:** User with a positive amount and banned: Money transfer is not allowed.
- **Test Case 3:** User with a negative amount and not banned: Money transfer is not allowed.
- **Test Case 4:** User with a negative amount and banned: Money transfer is not allowed.



| Condition | Step 1 | Step 2 | Step 3 | Step 4 |
|---|---|---|---|---|
| User has positive account balance | Y | Y | N | N |
| User not banned | Y | N | Y | N |
| **Expected Outcome** | | | | |
| Money Transfer Successful | Y | N | N | N |

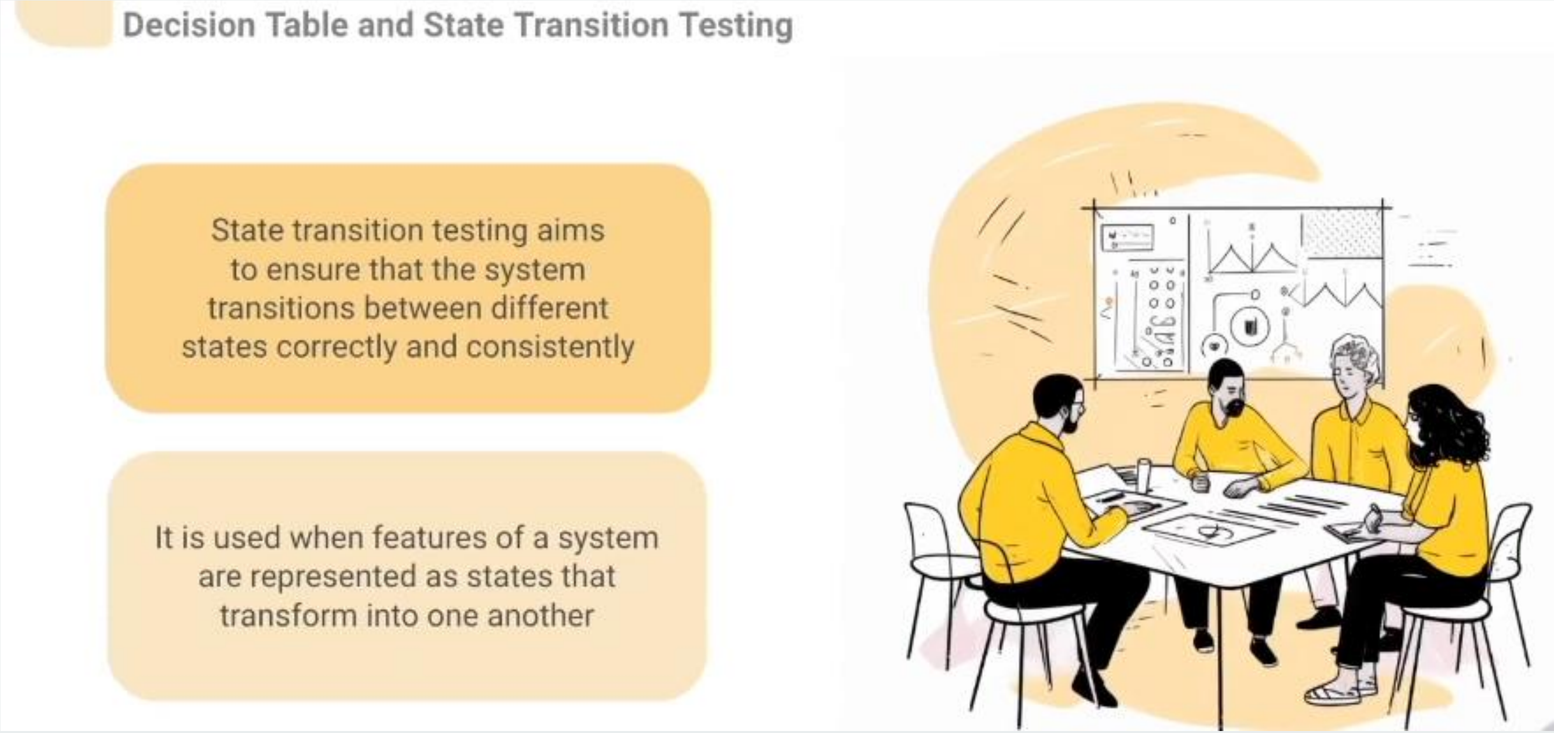


## Decision Table and State Transition Testing

The number of cases depends on the number of conditions, you can calculate the number of test cases:

**Number of Test Cases = 2^n**

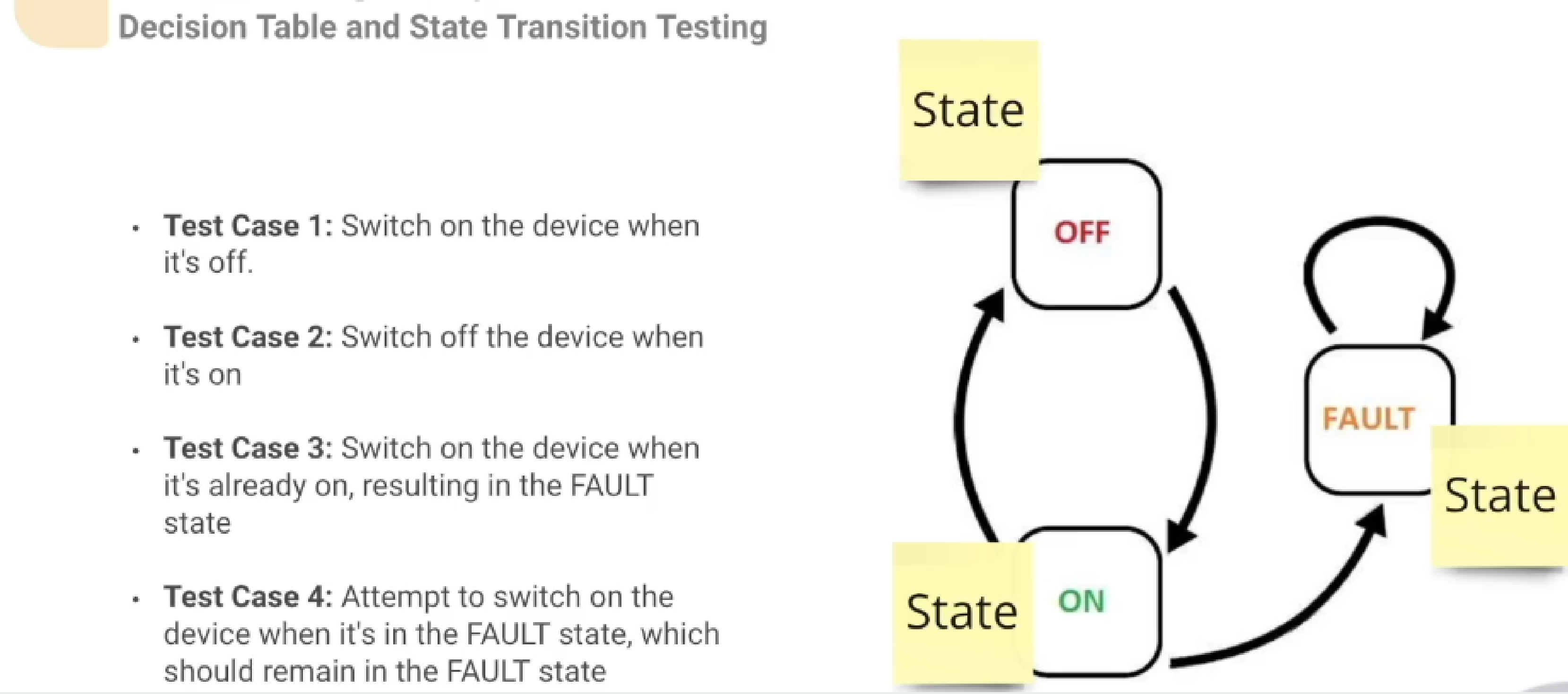
For 2 condition: 2^2 = 4 test cases
For 3 condition: 2^3 = 8 test cases
For 4 condition: 2^4 = 16 test cases
.
.
.



## Decision Table and State Transition Testing

State transition testing aims to ensure that the system transitions between different states correctly and consistently

It is used when features of a system are represented as states that transform into one another

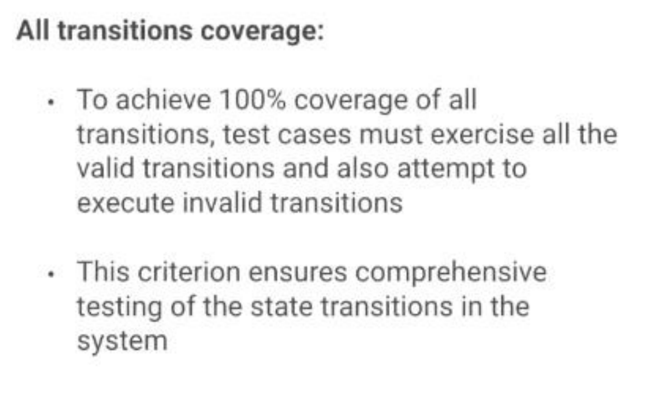


## Decision Table and State Transition Testing

- **Test Case 1:** Switch on the device when it's off.
- **Test Case 2:** Switch off the device when it's on
- **Test Case 3:** Switch on the device when it's already on, resulting in the FAULT state
- **Test Case 4:** Attempt to switch on the device when it's in the FAULT state, which should remain in the FAULT state

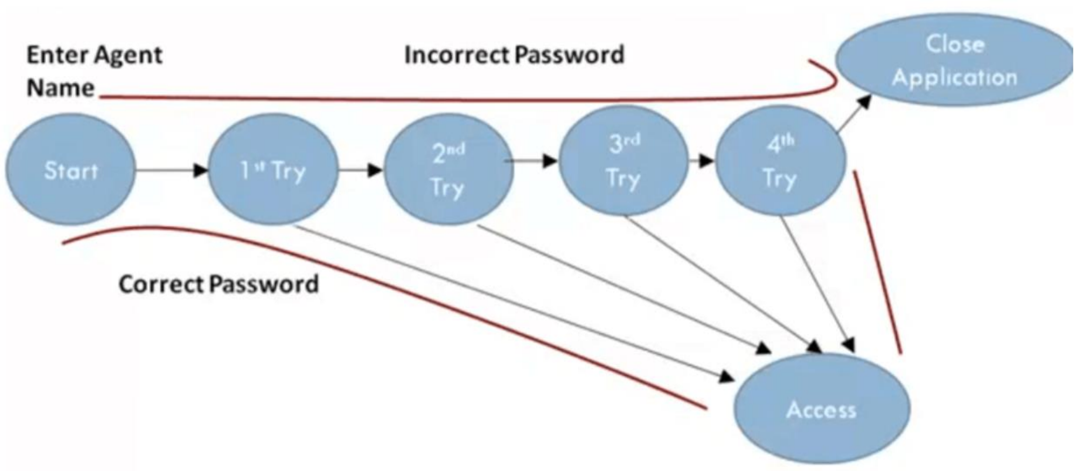
State
OFF
FAULT
State
State
ON

# Example 2



**All transitions coverage:**

- To achieve 100% coverage of all transitions, test cases must exercise all the valid transitions and also attempt to execute invalid transitions
- This criterion ensures comprehensive testing of the state transitions in the system


Enter Agent Name
Incorrect Password
Close Application
Start
1st Try
2nd Try
3rd Try
4th Try
Correct Password
Access