



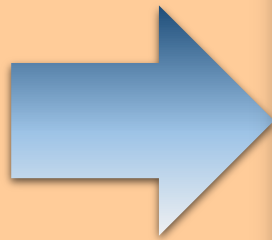
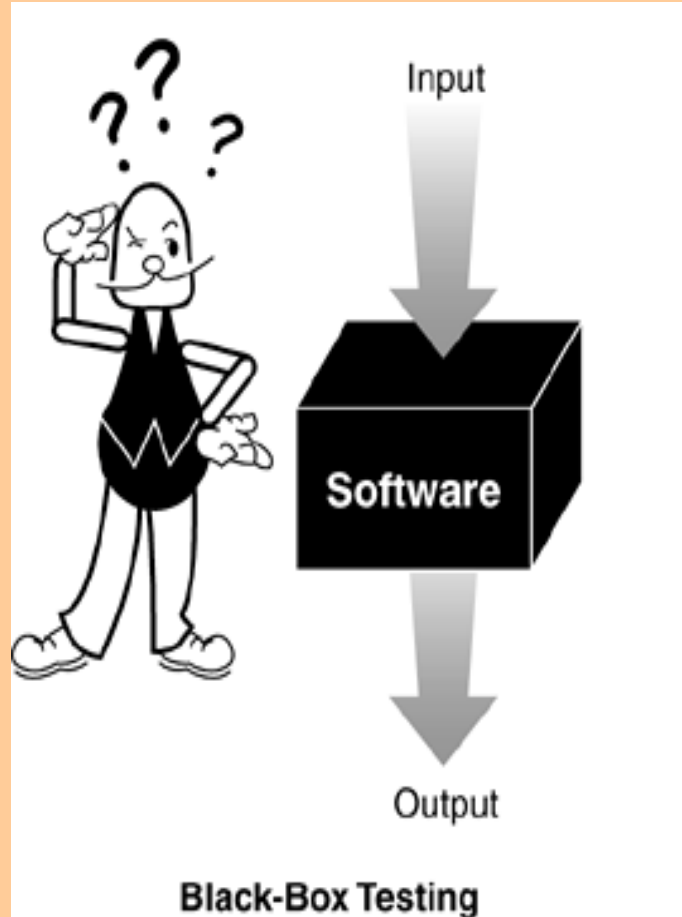
# BlackBox Testing

Testing & Implementasi

Elkin Rilvani

Pertemuan 5

[elkinrilvani@gmail.com](mailto:elkinrilvani@gmail.com)



Pengujian software terhadap spesifikasi perilaku eksternal tanpa mengetahui rincian pelaksanaan internal



Tujuan pengujian black box adalah untuk memverifikasi kebenaran perilaku software yang secara langsung mendukung aktivitas bisnis sehari-hari.

(Gerald D. Everettn, 2007)



## **Blackbox Testing are**

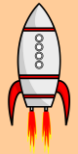
- Software Requirements
- Use Cases
- Only the executable program, and its data.



## **Blackbox testing cenderung menemukan jenis error**

- Missing functions
- Usability problems
- Performance problems
- Concurrency and timing errors
- Initialization and termination errors
- Etc.

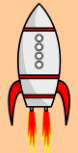
# Type Of Blackbox Testing



Equivalence partitioning



Boundary-value analysis



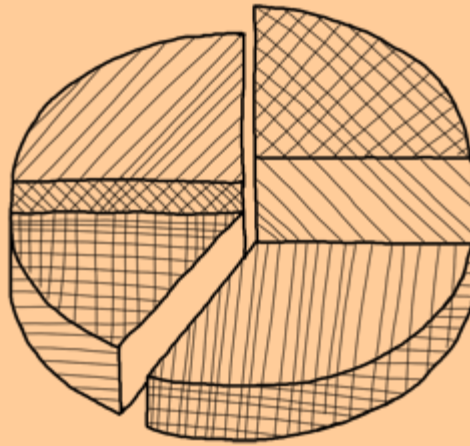
Cause-effect graphing



Error guessing

**(Glenford J. Myers, 2004)**

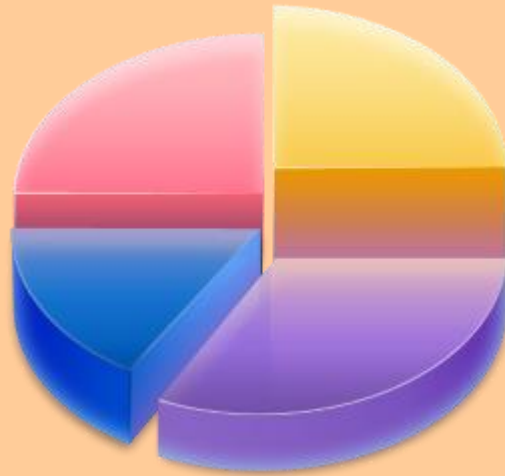
# Equivalence



# Partitioning

- Teknik pengujian black box di mana kasus uji dirancang untuk mengeksekusi perwakilan dari partisi ekuivalen (kesetaraan).
- Pada prinsipnya, uji kasus dirancang untuk mencakup setiap partisi minimal satu kali.
- (Rex Black & Jamie L. Mitchell, 2008)

# Equivalence



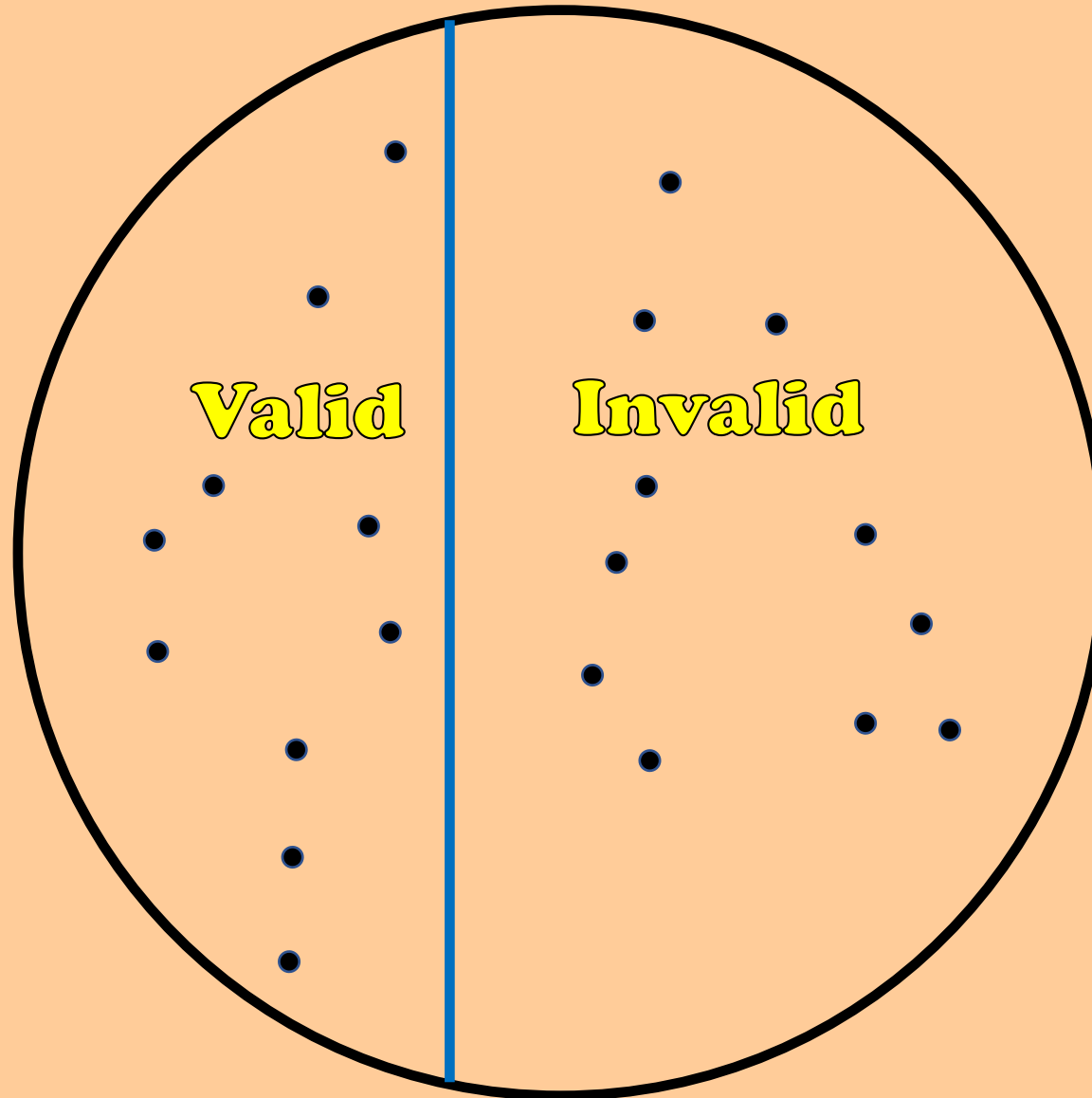
# Partitioning

## **Beberapa langkah pengujian Equivalence Partitioning**

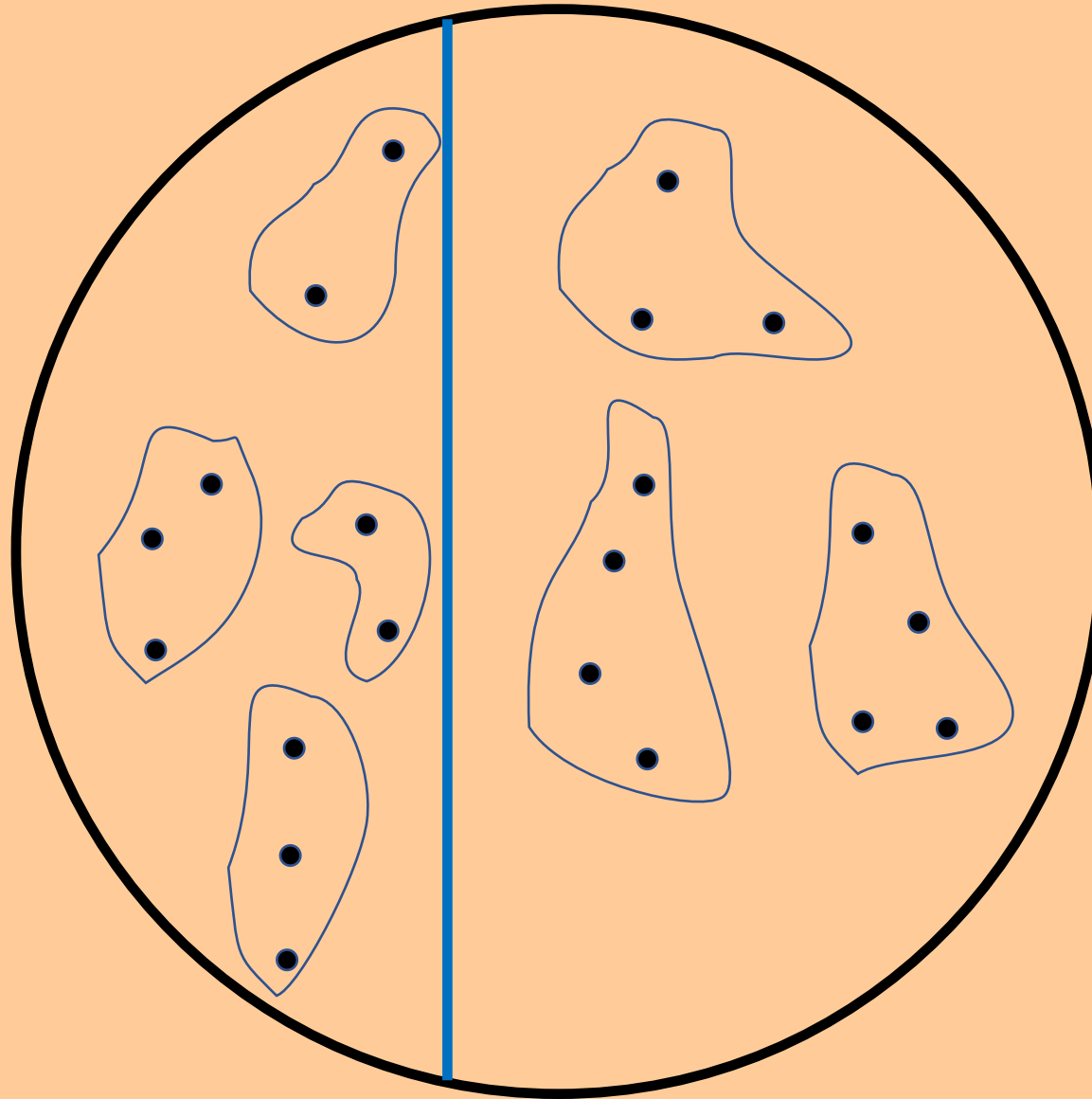
- Partisi kasus uji menjadi "kelas Equivalence"
- Setiap kelas Equivalence berisi seperangkat uji kasus "Equivalent"
- Dua uji kasus dianggap equivalent jika kita mengharapkan program memproses keduanya dengan cara yang sama (yaitu, ikuti jalur yang sama melalui kode).
- Jika mengharapkan program memproses dua kasus uji dengan cara yang sama, cukup uji salah satunya, sehingga mengurangi jumlah kasus uji yang harus jalankan.



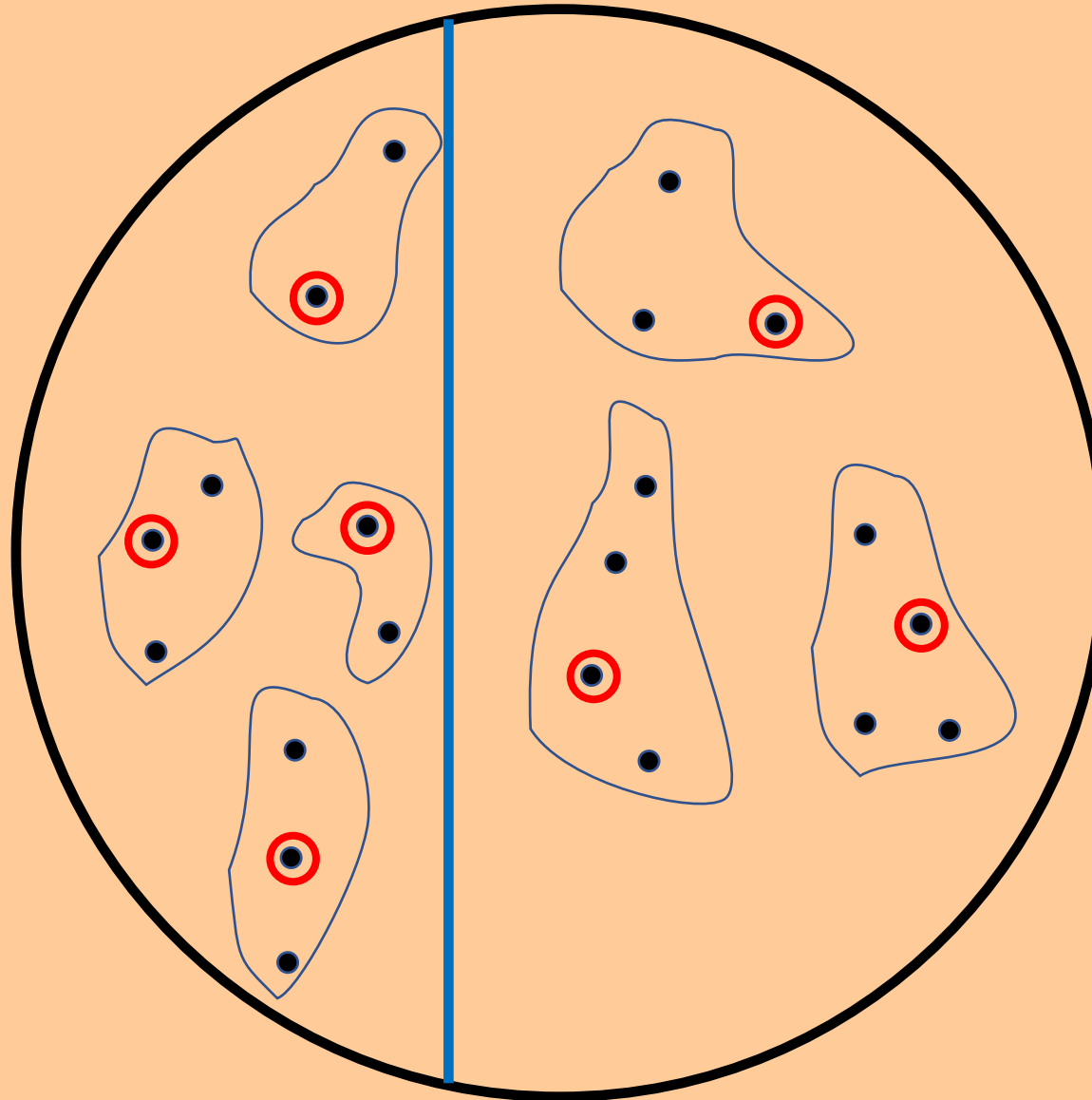
- First-level partitioning: Valid vs. Invalid test cases

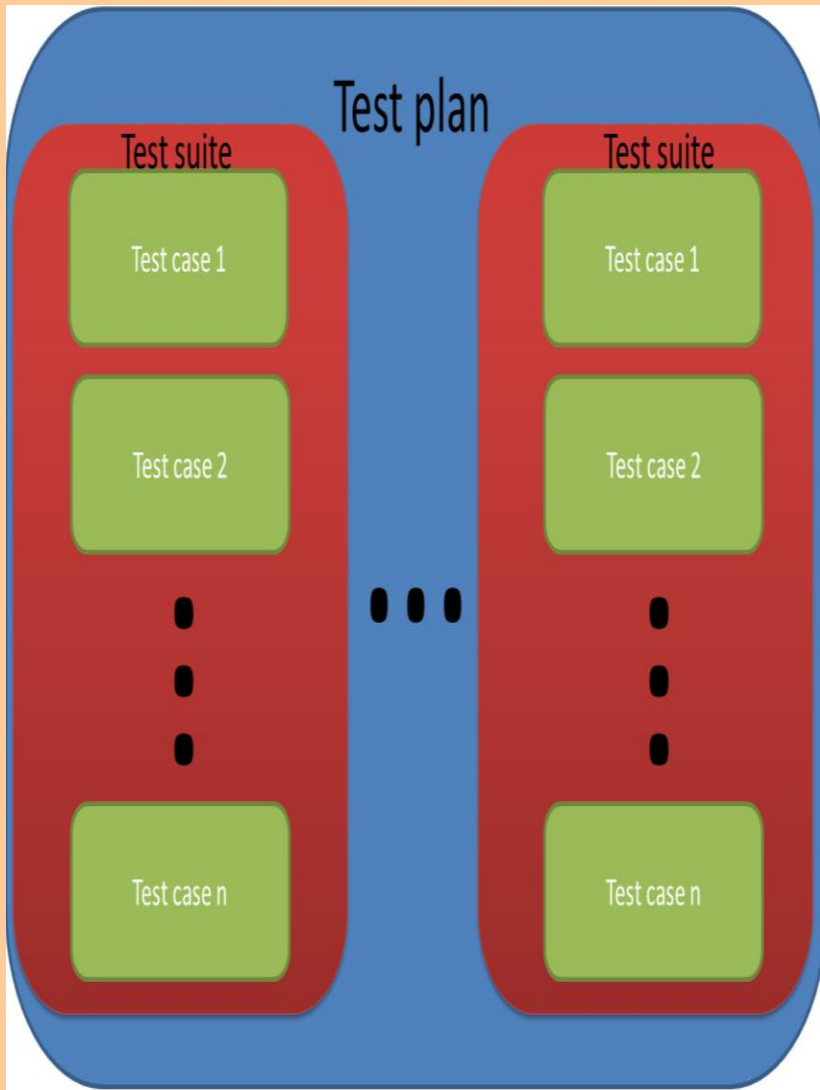


- Partition valid and invalid test cases into equivalence classes



- Buatlah sebuah test case untuk setidaknya satu nilai dari setiap kelas equivalence









- Saat merancang kasus uji, Anda dapat menggunakan definisi "Equivalence" yang berbeda, yang masing-masing akan memisahkan ruang uji dengan berbeda
- Contoh: `int Add (n1, n2, n3, ...)`
- Equivalence Definition 1: uji partisi dengan jumlah input (1, 2, 3, dll)
- Equivalence Definition 2: uji partisi dengan tanda nomor yang dikandungnya (positif, negatif, keduanya)
- Equivalence Definition 3: uji kasus partisi dengan besarnya operan (bilangan besar, angka kecil, keduanya)
- dll



- Pengujian Kasus pada website
- Contoh: string Fetch (URL)
- Equivalence Definition 1: uji kasus partisi dengan protokol URL ("http", "https", "ftp", "file", dll.)
- Equivalence Definition 2: uji kasus partisi berdasarkan jenis file yang diambil (HTML, GIF, JPEG, Plain Text, dll.)
- Equivalence Definition 3: uji partisi dengan panjang URL (sangat pendek, pendek, sedang, panjang, sangat panjang, dll)
- dll.



# Input case - examples

Input	Valid Equivalence Classes	Invalid Equivalence Classes
A integer N such that: $-99 \leq N \leq 99$		
Nomor telepon Area code : [200, 999] Prefix: (200, 999] Suffix: Ada 4 digit		

# THE NEXT STEP

Input	Valid Equivalence Classes	Invalid Equivalence Classes
A integer N such that: $-99 \leq N \leq 99$	$[-99, -10]$ $[-9, -1]$ 0 $[1, 9]$ $[10, 99]$	$< -99$ $> 99$ Malformed numbers {12-, 1-2-3, ...} Non-numeric strings {junk, 1E2, \$13} Empty value
Nomor telepon Area code : [200, 999] Prefix: (200, 999] Suffix: Ada 4 digit	555-5555 (555)555-5555 555-555-5555 $200 \leq \text{Area code} \leq 999$ $200 < \text{Prefix} \leq 999$	Format tidak valid 5555555, (555) (555) 5555, dll. Area code $< 200$ atau $> 999$ Area code dengan non-numerik karakter Mirip dengan Awalan dan Akhiran

# Boundary Value Analysis

- Boundary Value Analysis **(BVA)** merupakan desain testing uji kasus yang melengkapi equivalence partitioning.
- Dari pada memfokuskan hanya pada kondisi input, BVA juga menghasilkan kasus uji dari domain output.





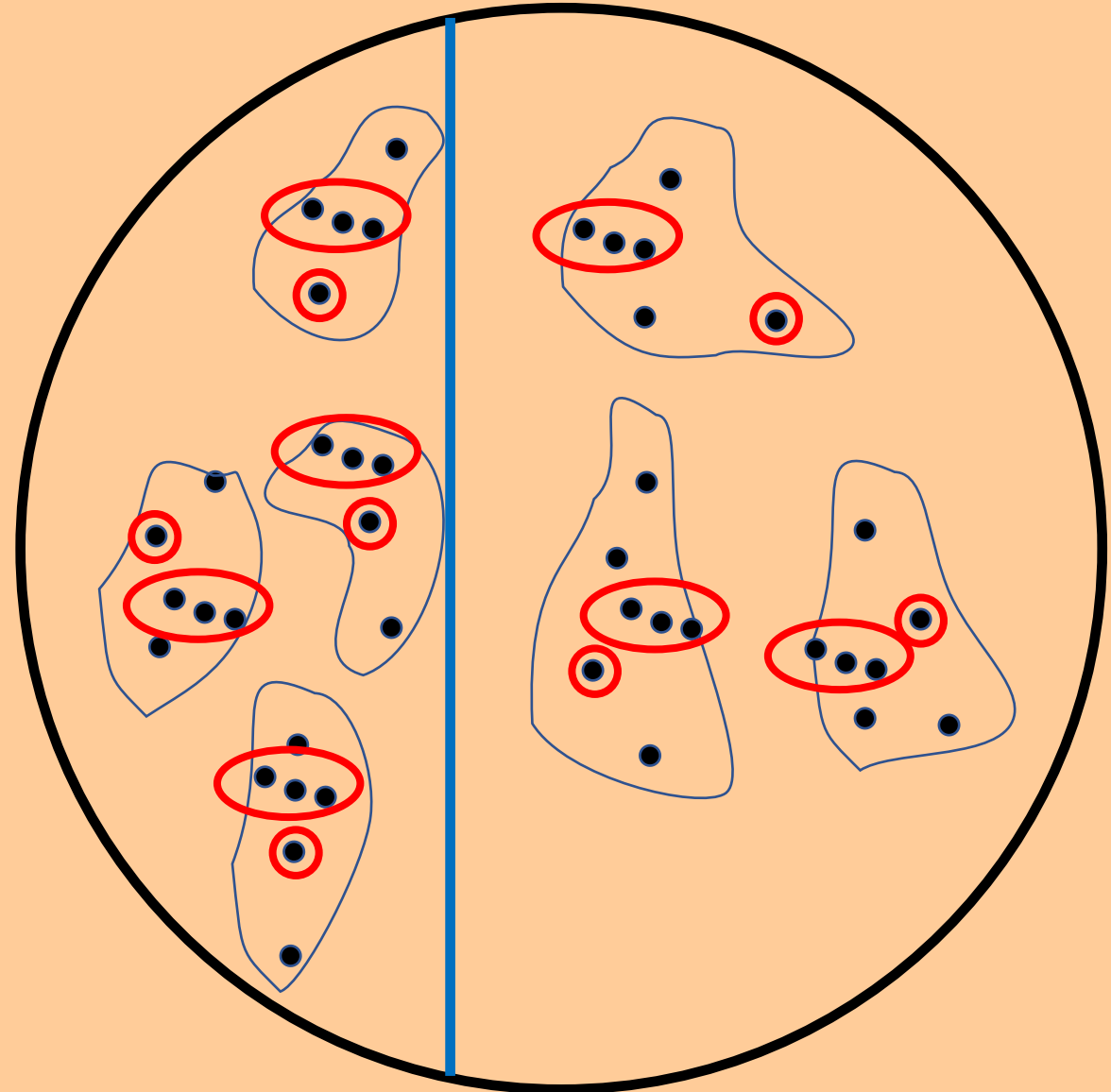


- Software yang digunakan untuk aplikasi perbankan menerima data dalam bentuk :
- Saat memilih nilai dari kelas Equivalent untuk diuji, gunakan nilai yang paling mungkin menyebabkan program gagal
- Kesalahan cenderung terjadi pada batas kelas Equivalent dan bukan pada "pusat"
- Jika  $(200 < \text{areaCode} \ \&\& \ \text{areaCode} < 999)$  {// kode area yang valid} Salah !
- Jika  $(200 \leq \text{areaCode} \ \&\& \ \text{areaCode} \leq 999)$  {// kode area yang valid}
- Pengujian kode area 200 dan 999 akan menangkap kesalahan ini, namun nilai pusat seperti 770 tidak akan

- Selain nilai pusat pengujian, kita juga harus menguji nilai batas
- Tepat di batas
- Sangat dekat dengan batas di kedua sisinya



- **BVA** Step
- Create test cases to test boundaries of equivalence classes



# BVA Case -examples

Input	Boundary Cases
A integer N such that: $-99 \leq N \leq 99$	?
Nomor telepon Area code : [200, 999] Prefix: (200, 999] Suffix: Ada 4 digit	?



# NEXT STEP

Input	Boundary Cases
A integer N such that: $-99 \leq N \leq 99$	-100, -99, -98 -10, -9 -1, 0, 1 9, 10 98, 99, 100
Nomor telepon Area code : [200, 999] Prefix: (200, 999) Suffix: Ada 4 digit	Area code: 199, 200, 201 Area code: 998, 999, 1000 Prefix: 200, 199, 198 Prefix: 998, 999, 1000 Suffix: 3 digits, 5 digits

# Conclusion

- Nilai numerik sering dimasukkan string yang kemudian dikonversi menjadi bilangan internal (integer to string)
- Konversi ini mengharuskan program untuk membedakan antara digit dan non-digit
- Kasus BVA yang perlu dipertimbangkan: Akankah program menerima / harus dikonfersi kedalam bentuk digital dengan bentuk dibawah

Char	/	0	1	2	3	4	5	6	7	8	9	:
ASCII	47	48	49	50	51	52	53	54	55	56	57	58

# Error Guessing

Teknik uji-kasus-desain yang bisa disebut menebak kesalahan.

## Latar Belakang Error Guessing

- Pengalaman dalam melakukan testing
- Intusisi (Istilah untuk kemampuan memahami sesuatu tanpa melalui penalaran rasional dan intelektualitas)

## Hal Yang Didapat

- Kesalahan tertentu (yang biasa terjadi/ tak terduga)
- Dokumentasi hasil kesalahan



# Basic Idea Tehnique

- Apakah program menerima "Input kosong" sebagai jawaban?
- Dua data memiliki inputan yang sama (bagaimana respon system).
- Memprediksi kemungkinan kesalahan input data

[illegible]