Exercise 1

- Apply the equivalence partitioning black box technique to obtain the test cases for a software module that classifies individuals according to the following inputs:
  - Code: string of 3 digits not starting with "00"
  - Control character: A character in 'a'..'z' or character '-'
  - Person Type: "hired" or "interim"
- The expected output values are:
  - "S1", if the code represents an even number and person is "hired".
  - "S2", if the code represents an even number and person is "interim".
  - "S3", if the code represents an odd number and person is "hired".
  - "S4", if the code represents an odd number and person is "interim".
  - "S5", if the control character is "-".
  - "S6", otherwise.
- Note: in case of conflict, S5 has more priority.

! LOOK EXPECTED OUTPUT → REDUCED CLASSES

Equivalence Partitioning

- **Identification Heuristics** of equivalence classes:
  - a) If a **range of values** is specified for the input data, a valid class and two invalid classes will be created ! Check the type of value
  - b) If a **finite value** is specified a valid class and two invalid classes will be created
  - c) If a condition of type «it must be» or a **boolean** condition is specified (e.g. " the first character must be a character from the alphabet», a valid class «it is an alphabet character» and another invalid class «it is not an alphabet character» will be created → EXACT VALUE
  - d) If a **set of valid accepted values** and the program treats them in a different way, then a valid class for each value and another invalid class must be created
  - e) If it is suspected that some concrete elements of a class are handled in a different way, the class must be partitioned into **reduced classes**

Diff approach:

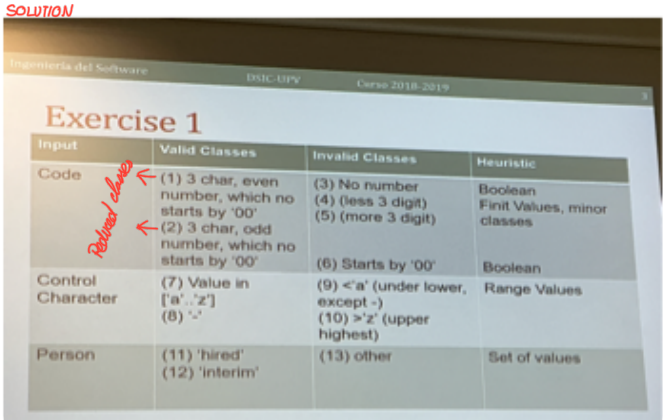
Input classes	Valid cases	Invalid cases	Heuristic
Code	Code.ParseInt ∈ [10 ... 999]	Code. ParseInt() ⇒ Error Code. ParseInt() < 10 Code. ParseInt() > 999	range → 1 valid, 2 invalid ~~boolean~~
Control_char	Control_char ∈ ['a'...'z'] Control_char = '-'	char < 'a' (except from '-') char > 'z'	range ~~set of accepted values~~
Person	"hired" "interim"	other	Set of values

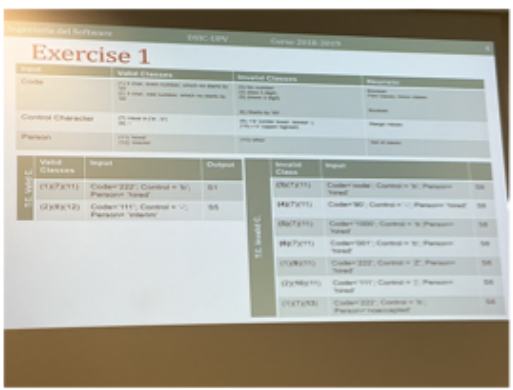
! Cover as many valid cases in 1 test

Valid classes	Input	Output

Invalid Classes	Input	Output

SOLUTION





Exercise 4

Apply the equivalence partitioning black box technique to obtain the test cases for a method which generates a report according to the following inputs:

- Student Name: Which contains at least name and one surname
- Group: three characters, the first is a letters A,C or D, and the following two are two digits from 01 until 15
- Theory mark (T): positive number (10 maximum)
- Laboratory mark(L): positive number (10 maximum)
- Deliverable mark (D) : positive number (8 maximum)

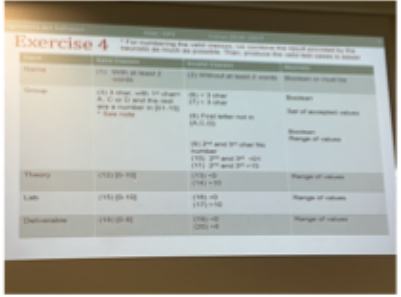
In the report, the method adds the final course mark by means of the equation: 0.6*T+0.4*P+0.1*D

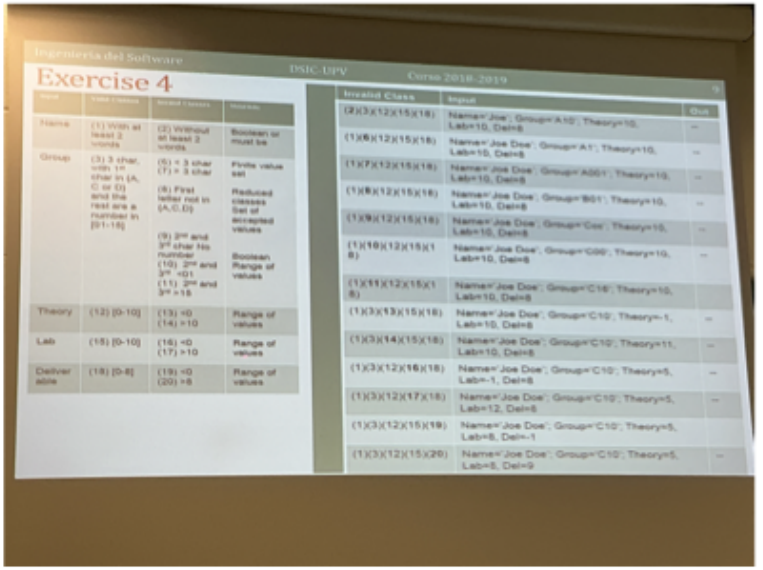
Provide the equivalence partitioning table with the input variable, valid classes, invalid classes and heuristic applied. Also, provide two tables the resulting with test cases, one for valid classes and another for the invalid ones.

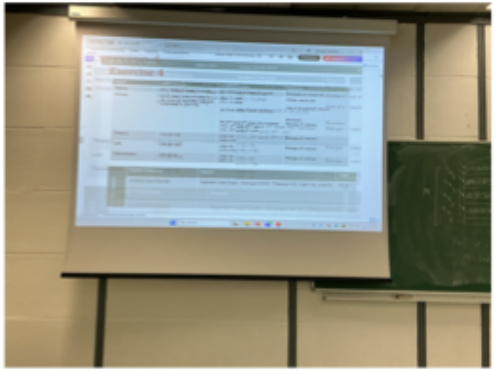
Input classes	Valid classes	Invalid classes	Heuristic
Student name	(1) Has at least a name and 1 surname ✓	(2) It only has less than 2 names ✗ OK, but we could test more	Boolean
Group	(3) First letter must be A, C or D * SET condition and Ends with number in [01, ..., 15]	(4) The first letter is not A, C or D (5) The last two characters don't belong to [01, ..., 15]	~~Exact value~~ / Boolean * As the code behaves in the same way for all
Theory_mark	(6) Value in [0, ..., 10]	(7) Value < 0 (8) Value > 10	Range
Laboratory_mark	(9) Value in [0, ..., 10]	(10) Value < 0 (11) Value > 10	Range
Delivery_mark	(12) Value in [0, ..., 8]	(13) Value < 0 (14) Value > 8	Range

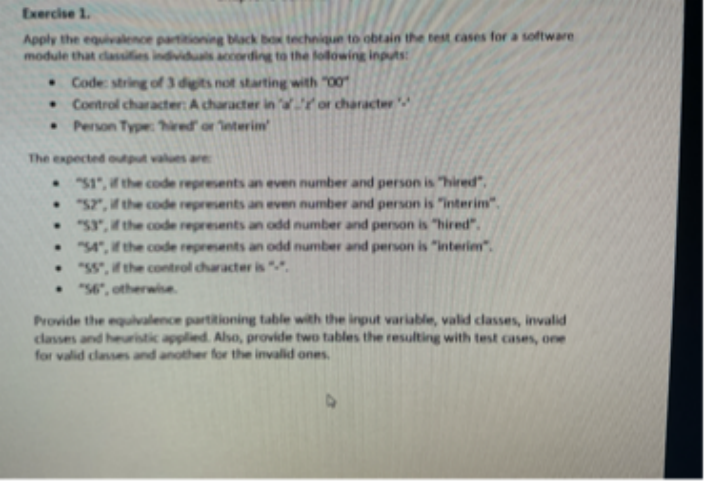
Valid cases	Test	Output

Invalid cases	Test	Output


Exercise 4


Ingeniería del Software
DSIC UPV
Curso 2018-2019
Exercise 4
Invalid Class
Input
Out
(2)(3)(12)(15)(18)
(1)(6)(12)(15)(18)
(1)(7)(12)(15)(18)
(1)(8)(12)(15)(18)
(1)(9)(12)(15)(18)
(1)(10)(12)(15)(18)
(1)(11)(12)(15)(18)
(1)(3)(13)(15)(18)
(1)(3)(14)(15)(18)
(1)(3)(12)(16)(18)
(1)(3)(12)(17)(18)
(1)(3)(12)(15)(19)
(1)(3)(12)(15)(20)





Exercise 1.

Apply the equivalence partitioning black box technique to obtain the test cases for a software module that classifies individuals according to the following inputs:

- Code: string of 3 digits not starting with "00"
- Control character: A character in 'a'..'z' or character '-'
- Person Type: 'hired' or 'interim'

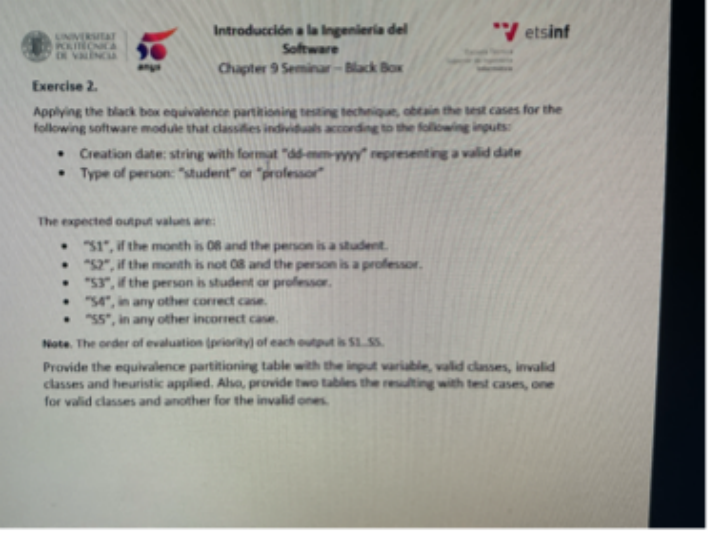
The expected output values are:

- "S1", if the code represents an even number and person is "hired".
- "S2", if the code represents an even number and person is "interim".
- "S3", if the code represents an odd number and person is "hired".
- "S4", if the code represents an odd number and person is "interim".
- "S5", if the control character is "-".
- "S6", otherwise.

Provide the equivalence partitioning table with the input variable, valid classes, invalid classes and heuristic applied. Also, provide two tables the resulting with test cases, one for valid classes and another for the invalid ones.

Input variable	Valid classes	Invalid classes	Heuristic
Code	(1) string of 3 digits, not starting with 00 and representing an even nº (2) string of 3 digits, not starting with 00 and representing an odd nº	(3) String has less than 3 digits (4) String has more than 3 digits (5) String starts with 00 (6) Not a number	Boolean Reduced classes Boolean
Control character	(7) char belonging to ['a', ..., 'z'] (8) char = '-'	(9) char < 'a' (except from '-') (10) char > 'z'	Range of values
Person Type	(11) "hired" (12) "interm"	(13) type ∉ [hired, interm]	Set of valid accepted values

Invalid classes	Input	Output
(3) (7) (11)	Code = "abc" Control character = 'a' Person = 'hired'	S6
(4) (7) (11)	Code = "10" Control character = 'a' Person = 'hired'	S6
(5) (8) (12)	Code = "1000" Control character = '-' Person = 'interm'	S6
(6) (8) (12)	Code = '001' Control character = '-' Person = interm	S6
(1) (9) (12	Code = '100' Control character = '.' Person = interm	S6
(1) (10) (11)	Person = '100' Control character = 'Z'	S6
(2) (8) (13)	Person = '101' Control char = '-' Person = "halal	S6



Introducción a la Ingeniería del Software
Chapter 9 Seminar – Black Box

Exercise 2.

Applying the black box equivalence partitioning testing technique, obtain the test cases for the following software module that classifies individuals according to the following inputs:

- Creation date: string with format "dd-mm-yyyy" representing a valid date
- Type of person: "student" or "professor"

The expected output values are:

- "S1", if the month is 08 and the person is a student.
- "S2", if the month is not 08 and the person is a professor.
- "S3", if the person is student or professor.
- "S4", in any other correct case.
- "S5", in any other incorrect case.

Note. The order of evaluation (priority) of each output is S1..S5.

Provide the equivalence partitioning table with the input variable, valid classes, invalid classes and heuristic applied. Also, provide two tables the resulting with test cases, one for valid classes and another for the invalid ones.

Input	Valid classes	Invalid classes	Heuristic
Creation_date	(1) Has a format dd-mm-yyyy representing a valid date where mm ≠ 08 (2) Has a format dd-08-yyyy representing a valid date	(3) Has a different format (4) It does not represent a valid date	Reduced classes Boolean
Type of person	(5) "student" (6) "profesor"	(7) type of person ∉ [student, profesor]	Set of accepted values

Valid classes	Input	Output
(1) (6)	date = 09-11-2005, type = profesor	S2
(2) (5)	date = 29-08-2014, type = student	S1

clases	Input	Output
(3) (5)	date = 01-enero-1999, type = student	S5
(4) (6)	date = 42-23-0000, type = profesor	S5
(1) (7)	date = 29-11-2005, type = hola	S5