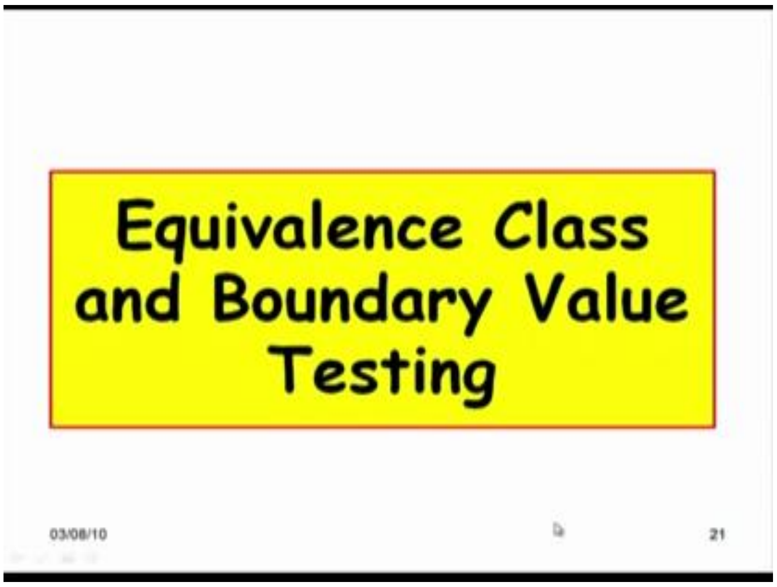
Software Testing

Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 06
Equivalence and Boundary Value Testing

Welcome to this session, we will discuss about the Equivalence Class Partition and the Boundary Value Test Cases.

(Refer Slide Time: 00:29)

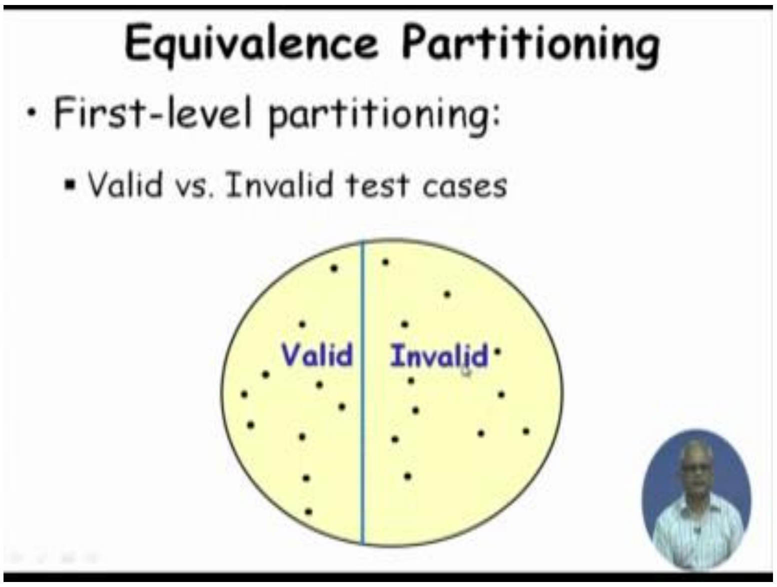


Equivalence Class and Boundary Value Testing

03/08/10 21

Saying that these are black box testing, two strategies for black box testing very popular strategies are Equivalence Class and Boundary Value Testing. After we complete this we will look at the other black box testing strategies.

(Refer Slide Time: 00:50)

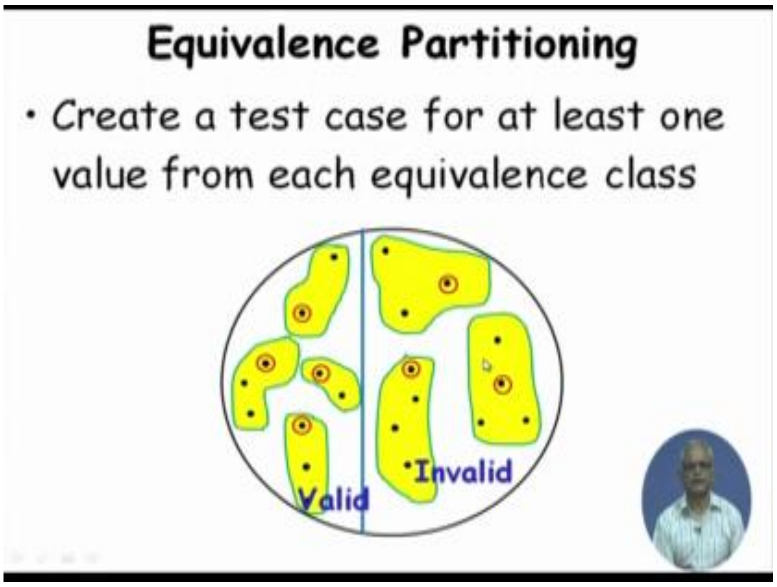


Equivalence Partitioning

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

We are saying that the main problem here in equivalence partitioning is to design the equivalence classes, and once we have designed the equivalence classes, it is straight forward to choose one value from each equivalence class, and also to consider if the equivalence classes have any boundary. We are saying that given any unit for which you have to design equivalence class, there are at least two - one are the invalid set of values and the other are the valid set of values, and each of them will have typically different sets of equivalence classes.

(Refer Slide Time: 01:43)

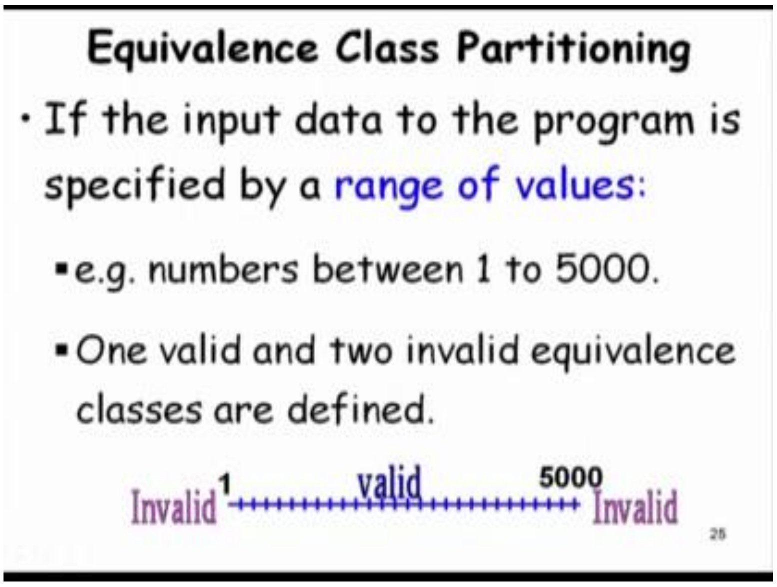


Equivalence Partitioning

- Create a test case for at least one value from each equivalence class

So, we need to identify each of this, the valid set of equivalence classes and the invalid set of equivalence classes. And for each of these, equivalence classes which have been identified the valid and the invalid set of equivalence classes we need to select one value and that will form our equivalence class test.

(Refer Slide Time: 02:09)



Equivalence Class Partitioning

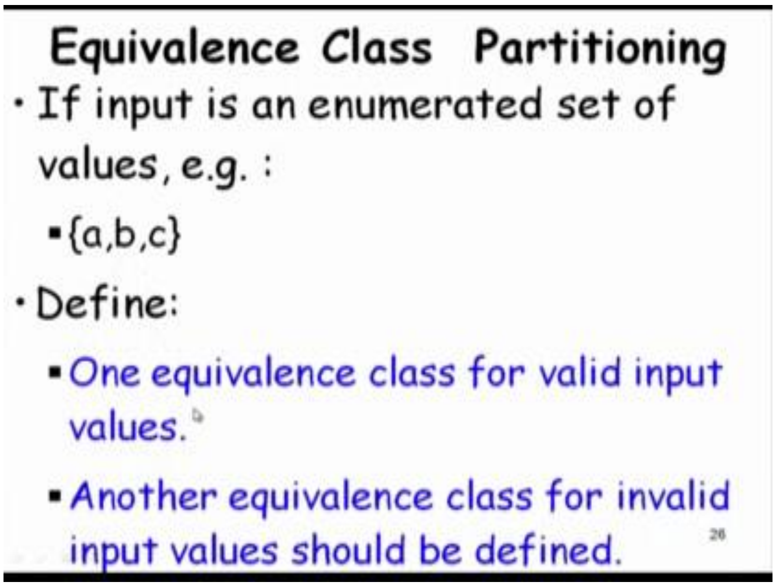
- If the input data to the program is specified by a range of values:
  - e.g. numbers between 1 to 5000.
  - One valid and two invalid equivalence classes are defined.

Invalid
1
valid
5000
Invalid
25

A very simple example to start with; and as I said earlier that designing test cases can be very challenging; I start with a very simple one and then we look at more challenging ones. So the simplest one, we just specify that a function takes a value between 1 and 5000 and displays the result. And let us assume that there is only one scenario here; let us say it is just takes a value between; 1 to 5000 and checks whether it is even or odd.

Then we will have 1 equivalence class which is valid, and there are two invalid set of equivalence class which is less than 1 and more than 5000. The valid will have two equivalence classes; one are the, whether the number is an odd number whether it is an even number.

(Refer Slide Time: 03:26)



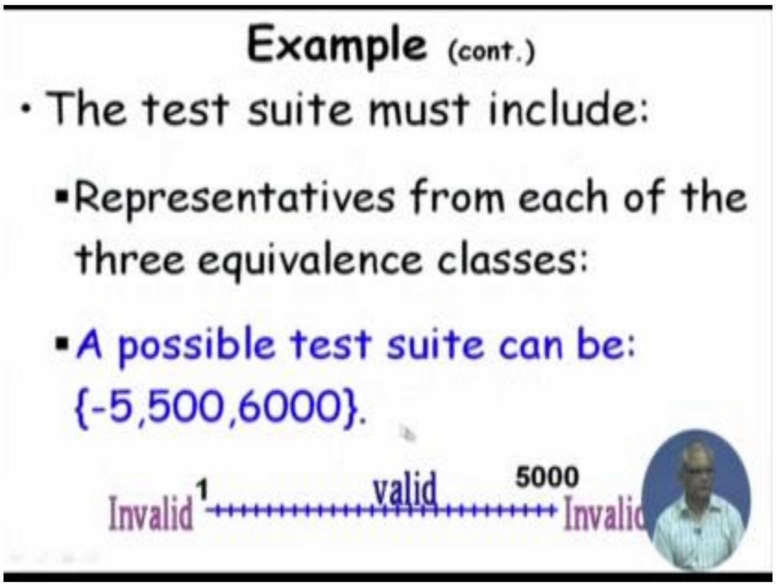
Equivalence Class Partitioning

- If input is an enumerated set of values, e.g. :
  - {a,b,c}
- Define:
  - One equivalence class for valid input values.
  - Another equivalence class for invalid input values should be defined.

26

We said that if the input value is enumerated set of equivalence classes then we have 1 valid equivalence class and 1 invalid equivalence class, which is either a, b, c or which is neither of a, b, c.

(Refer Slide Time: 03:51)



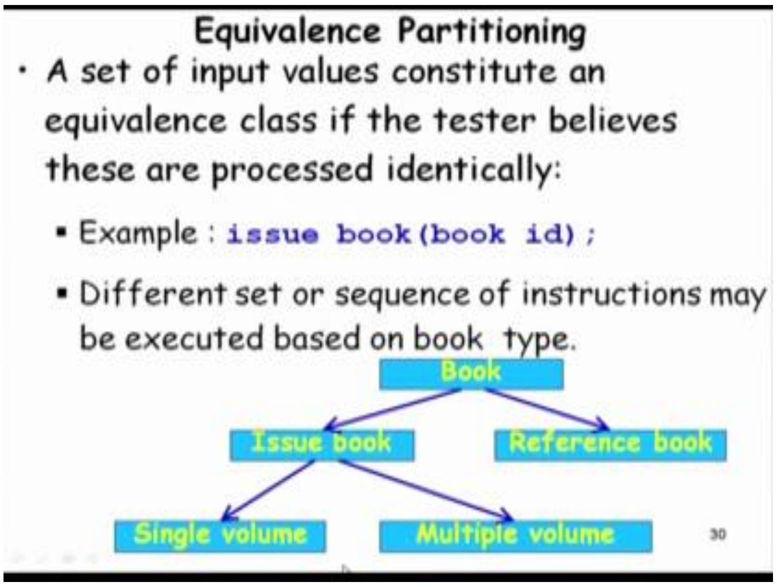
Example (cont.)

- The test suite must include:
  - Representatives from each of the three equivalence classes:
  - A possible test suite can be: {-5,500,6000}.

Invalid
1
valid
5000
Invalid

This is trivial example, 1 to 5000 and it produces the square root, so we will consider three equivalence classes for this; 1 valid and 2 invalid. If we have a set of numbers we have 1 valid and 1 invalid.

(Refer Slide Time: 04:21)



Equivalence Partitioning

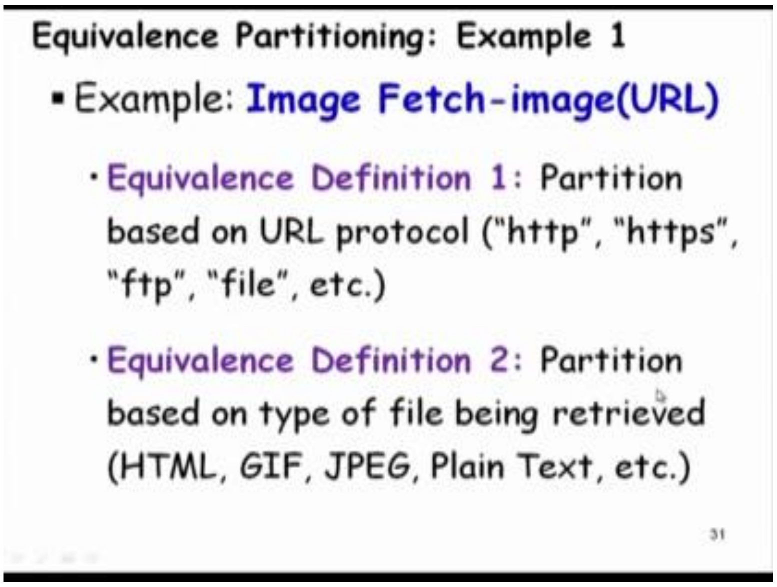
- A set of input values constitute an equivalence class if the tester believes these are processed identically:
  - Example : `issue book(book id);`
  - Different set or sequence of instructions may be executed based on book type.

Book
Issue book
Reference book
Single volume
Multiple volume
30

Now let's say, we have a unit which is a function, and the function is issue book and the parameter it takes book id. So we need to design equivalence classes, we need to look at the functional description of this and check what are the scenarios of operation.

And the scenarios of operation can also be identified by looking at the output. So the scenarios of operation may be that, their books are which can be issued out, books which cannot be issued out, which are reference books and those books which can be issued out that can be single volume or multiple volume, and these form our set of equivalence classes. Reference book, multiple volume and single volume these are our equivalence classes.

(Refer Slide Time: 05:37)


Equivalence Partitioning: Example 1

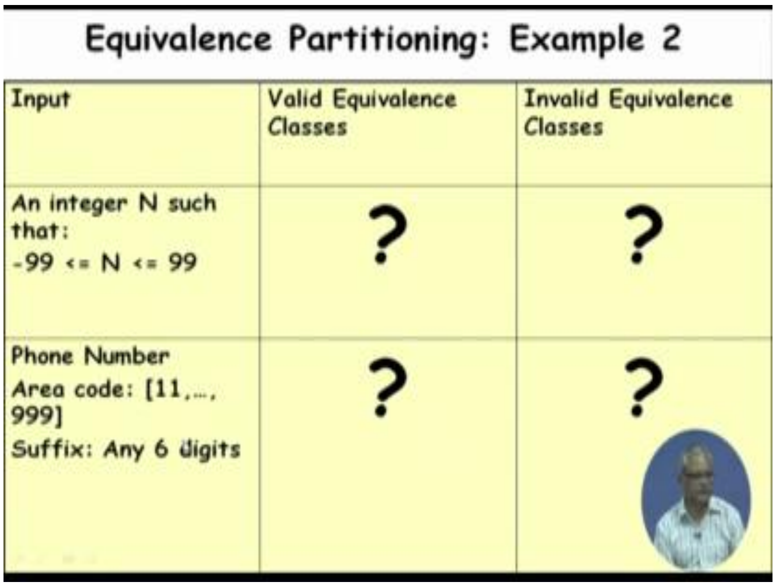
- Example: Image Fetch-image(URL)
    - Equivalence Definition 1: Partition based on URL protocol ("http", "https", "ftp", "file", etc.)
    - Equivalence Definition 2: Partition based on type of file being retrieved (HTML, GIF, JPEG, Plain Text, etc.)

31

Now let us look at another example. Let us say, we have a function, name of the function is Fetch-image it takes a URL and returns an image. So what will be the equivalence classes for this function? Fetch-image, which takes URL and returns an image - in this case, we can have multiple definition of equivalence and therefore we have to consider all of them. For example, one equivalence class can be, the URL can be based on “http”, “https’, “ftp”, “file”, etcetera all these are valid URL categories. Then we have to consider each of them as an equivalence class for the input.

Similarly, the value that is stored in the URL can be different types of images. For example, HTML, GIF, JPEG, etcetera, Plain Text, this can be our images. So, does it handle all these images, categories of images? For some data item we can construct multiple types of equivalence classes. So that is the complexity for some data items.

(Refer Slide Time: 07:17)



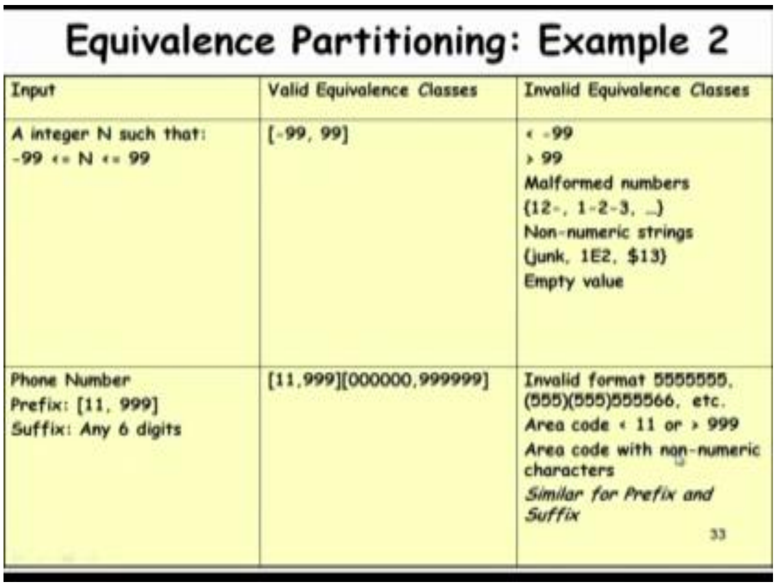
Equivalence Partitioning: Example 2

Input	Valid Equivalence Classes	Invalid Equivalence Classes
An integer N such that: -99 <= N <= 99	?	?
Phone Number Area code: [11,…, 999] Suffix: Any 6 digits	?	?

Let us look at another example. Let's say, the input is an integer of a number let us say between 99 and minus 99. So, what will the valid equivalence and the invalid equivalence classes? The answer is very simple, it will be between minus 99 and 99 inclusive, any value here is a valid equivalence class greater than 99 is invalid equivalence class and less than 99 is another invalid equivalence class. There will be two invalid equivalence class and one-in-one valid equivalence class.

Now, what about a phone number which is given by area code and a suffix. The area code is between 11 to 999, and the suffix is any 6 digits. So, what will be the invalid equivalence class? And then what will the set of invalid equivalence classes? Because there are different types invalid equivalence classes for this example.

(Refer Slide Time: 08:37)



Equivalence Partitioning: Example 2

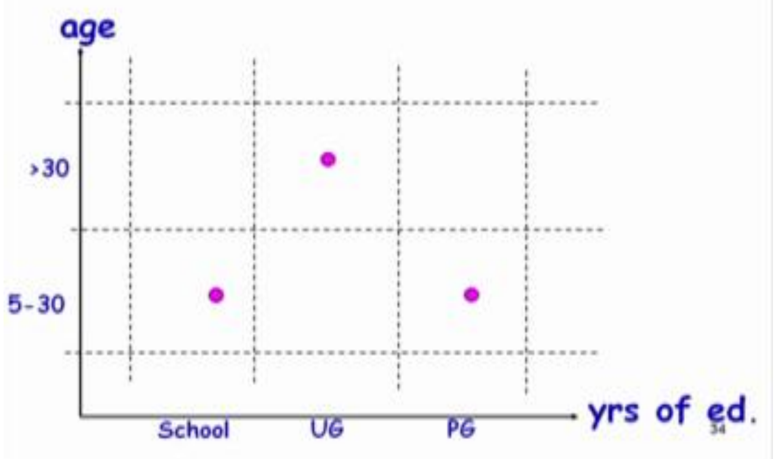
Input	Valid Equivalence Classes	Invalid Equivalence Classes
A integer N such that: -99 <= N <= 99	[-99, 99]	< -99 > 99 Malformed numbers {12-, 1-2-3, ...} Non-numeric strings {junk, 1E2, $13} Empty value
Phone Number Prefix: [11, 999] Suffix: Any 6 digits	[11,999][000000,999999]	Invalid format 5555555, (555)(555)555566, etc. Area code < 11 or > 999 Area code with non-numeric characters *Similar for Prefix and Suffix*

33

So, the valid equivalence class is when we have this, the prefix is between 11 to 999 and the suffix is 0000 to 99999. And, the invalid equivalence classes can be different types. For example, we might have invalid format for the prefix, invalid format for the suffix, numeric characters for area code and so on. We need to really define different types of invalid equivalence classes.

(Refer Slide Time: 09:27)

Weak Equivalence Class Testing

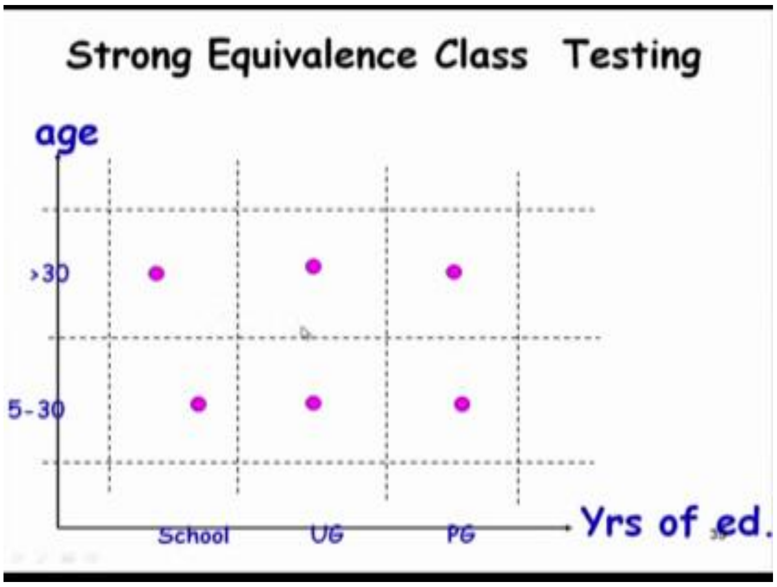


But, one thing we need to now consider is that if unit takes multiple data values, say one data value we said that the complexity might arise if we can have different notions of equivalence classes depending on different considerations. For example, the URL we said that there are multiple notions of equivalence classes depending on whether the URL is specified by http, https, ftp or file. Similarly, we can define another set of equivalence classes for same input value depending on whether the image stored at the URL is a JPEG image or is it a png image or is it a text type of image and so on.

Now, the complexity arises when we have a function or a unit that takes two parameters. Let say, we have a function that takes two parameters age and years of education. So, the years of education, let us say we identify that there are three equivalence class for this parameter; years of education is either school, UG, PG or something which is not even school not even PG. So, it basically display whether he is school qualified UG or PG and on the other axis we have the age is whether it is between 5 to 30, the person is 5 to 30 or is greater than 30. So, these are, let's say different equivalence classes for these two parameters.

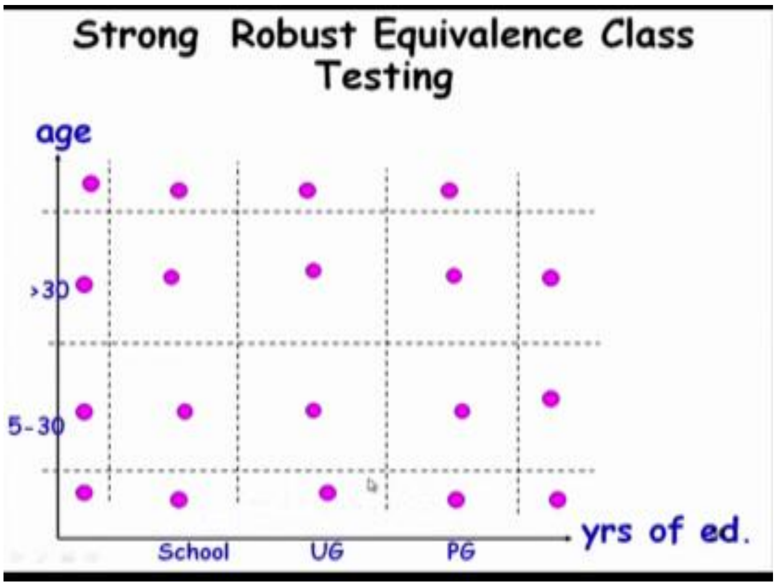
Now the issue that we are trying to address now is that, how do we combine these two equivalence classes? One simple thing, the weakest testing for this case is called as Weak Equivalence Class Testing, where we check we form the test cases such that all classes for one parameter is considered and same for the other parameter. So, both 30 and 5 to 30 are covered, and similarly here PG, UG and school are covered. This called as Weak Equivalence Testing.

(Refer Slide Time: 12:25)



We might also design Strong Equivalence Class Testing. In strong equivalence class testing we consider all possible combinations of the two parameters that is; it is school 5 to 30, school greater than 30, UG 5 to 30, UG greater than 5 to 30, PG 5 to 30, PG greater than 30 and so on. This is strong equivalence class testing, in which we consider all possible combinations of the equivalence classes for the parameters.

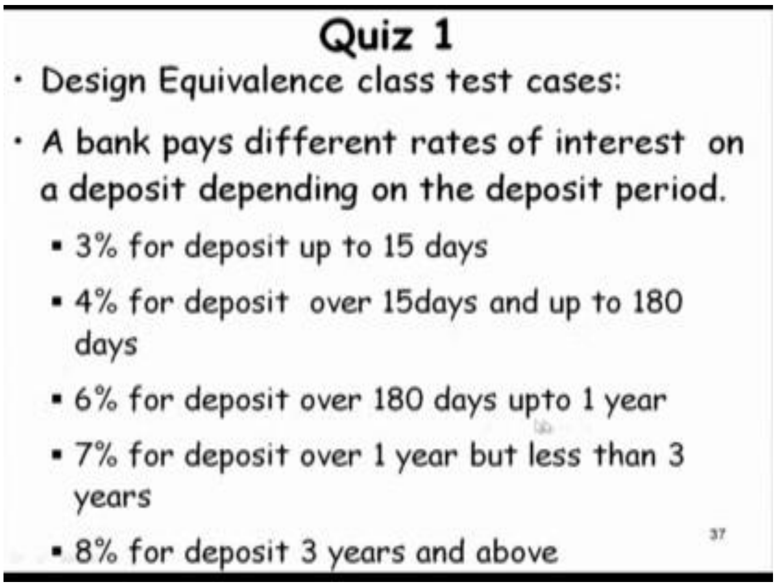
(Refer Slide Time: 13:00)



Strong Robust Equivalence Class Testing

Similarly, we can have a Strong Robust Equivalence Class Testing. Where, we consider the invalid values, so less than school less than 5, less than school between 5 to 30 and so on. So we will have one more here, we consider all possible combinations the input even the invalid once that we call as Strong Robust Equivalence class testing.

(Refer Slide Time: 13:42)



Quiz 1

- Design Equivalence class test cases:
- A bank pays different rates of interest on a deposit depending on the deposit period.
  - 3% for deposit up to 15 days
  - 4% for deposit over 15days and up to 180 days
  - 6% for deposit over 180 days upto 1 year
  - 7% for deposit over 1 year but less than 3 years
  - 8% for deposit 3 years and above

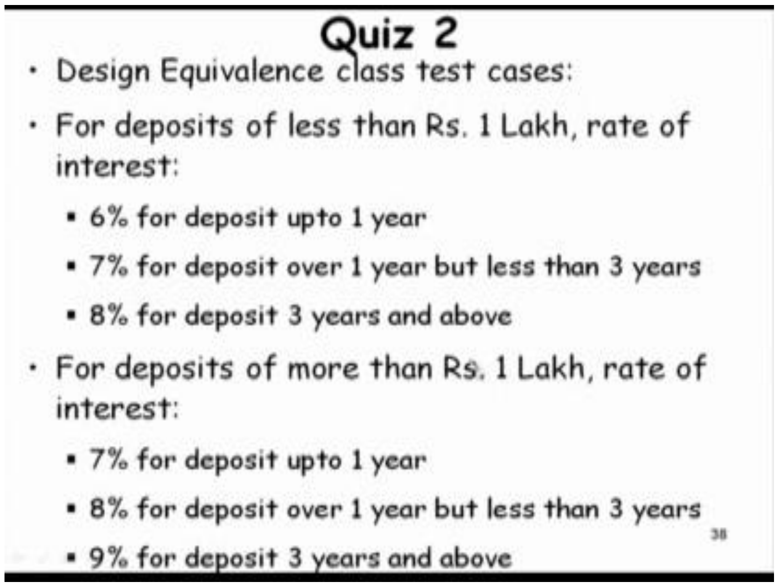
37

Now let us try to do some problems. The first one is the simplest will have two or three problems now. The first one is simplest, let us read through the problem statement. We need to Design Equivalence class test cases for a unit which is a functional that has been written.

The function implements the following; bank pays different rates of interest on a deposit depending on the deposit period. So, the parameter to this function will be the deposit period and the output will be the deposit rate. So, 3 percent for deposit up to 15 days, 4 percent for deposit over 15 and up to 180, 6 percent for deposit over 180 days to 1 year, 7 percent for 1 year but less than 3 year, and 8 percent for deposit 3 years and above. So, what will be the set of equivalence classes?

Now, let us look at the valid set of equivalence classes. The invalid of course, will be less than 0 days negative, by chance somebody enters that what will the software do. Similarly, here the other side and above, so there is no equivalence class on the right side of this, left side less than 0 and then each one is a scenario and therefore each one becomes an equivalence class. So, 0 to 15 days, 15 to 180 days, 180 to 1 year and 1 year but less than 3 years, 3 years and above. Each one is the scenario and each one will become equivalence class rather straight forward.

(Refer Slide Time: 16:12)



Quiz 2

- Design Equivalence class test cases:
- For deposits of less than Rs. 1 Lakh, rate of interest:
  - 6% for deposit upto 1 year
  - 7% for deposit over 1 year but less than 3 years
  - 8% for deposit 3 years and above
- For deposits of more than Rs. 1 Lakh, rate of interest:
  - 7% for deposit upto 1 year
  - 8% for deposit over 1 year but less than 3 years
  - 9% for deposit 3 years and above

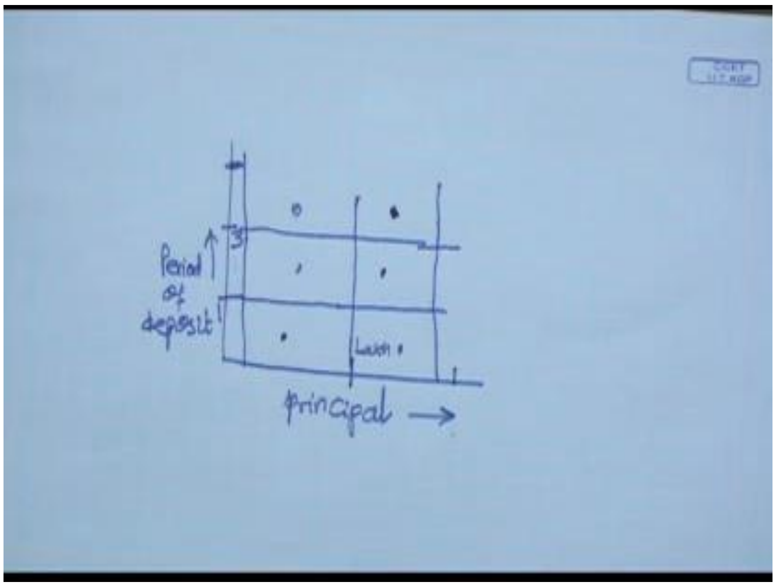
38

Now, let us look at the next problem. A problem is to design equivalence class for a similar problem but with the small change here. The function again displays the interest payable, but there are two parameters here; one is principal and the other is the deposit period. If deposit is for less than 1 lakh then the rate of interest is 6 percent, 7 percent, and 8 percent; up to 1 year, 1 year but less than 3 years. We looked at simple example and how to design the equivalence classes, but as we are saying that it can be really challenging. Now let us look at a small variation of the problem that we looked at it and see whether we are able to do it. So, that is Quiz 2, a small variation of the problem that we have looked at.

Here, the unit or the function takes two parameters; one is the principal and other is the period for which the deposit is being made. If the principal is less than 1 lakh, then 6 percent interest is payable for deposit up to 1 year and 7 percent interest is payable for deposit over 1 year but less than 3 years and 8 percent interest for deposit for 3 year and above. For deposits of more than 1 lakh, that is principal greater than 1 lakh the rate of interest is 7 percent for deposit up to 1 year, 8 percent for deposit over 1 year but less than 3 years and 9 percent for deposits over 3 year and above. So how do we do this?

As you can notice that the function takes two parameters. So,we will have equivalence class tests depending on whether we are doing weak equivalence class test, strong equivalence class test or robust and strong equivalence class test. If we are asked to do only or if we want to do only weak equivalence class test, then we will consider both these principal 1 lakh and 2 lakh and then both these scenarios of less than 1 year between 1 to 3 year and 3 year and above.

(Refer Slide Time: 19:32)

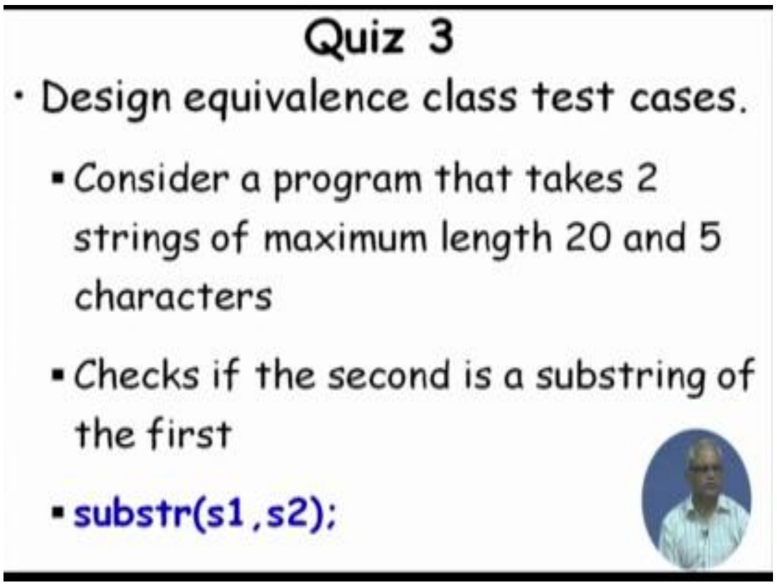


If we look at the two parameters, one is the principal, and the other is the period of deposit. We can identify two equivalence classes of principal; one is less than 1 lakh and the other is more than 1 lakh. And for the period of deposit we can identify three equivalence classes which are up to 1 year, 1 to 3 year and 3 year and above. If we do a weak equivalence class testing, we can take principal less than 1 lakh and deposit less than 1 year, we can take principal more than 1 lakh and period of deposit between 1 to 3 and we can any of these that period is greater than 3 and principal is greater than to 1 lakh and period is 3 year and above.

If we want to do a strong equivalence class testing we will have to consider all of these. So, principal is less than 1 lakh deposit is for 1 year, principal greater than 1 lakh and deposit is for less than 1 year, principal less than 1 lakh and deposit is between 1 to 3 year and so on. But these are the valid set of values. So these are strong equivalence testing, but what about robust technique, robust strong equivalence testing then we will have to consider the invalid inputs.

Invalid input can be negative period of deposit; it can be a non integer value and so on. Then we will have to consider combinations of those also, a negative value for period of deposit and negative value for the principal, negative value for the principal and period of deposit is 1 year and so on. Similarly, for a non integer value; so that will be a robust strong equivalence class testing for the simple case of two parameters, but we can have much more complex examples.

(Refer Slide Time: 22:50)

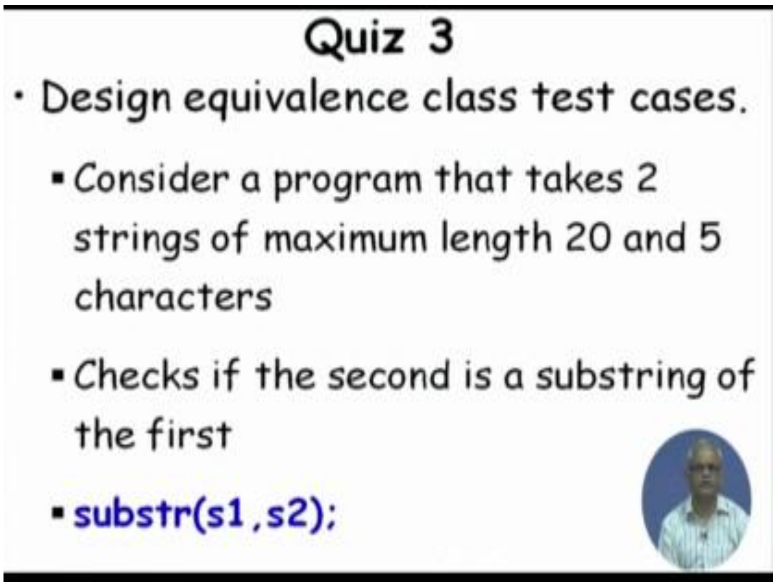

Quiz 3

- Design equivalence class test cases.
  - Consider a program that takes 2 strings of maximum length 20 and 5 characters
  - Checks if the second is a substring of the first
  - substr(s1,s2);

Now, let us try to look at another example. This is Designed Equivalence Class Test for a function which takes 2 strings. The maximum length of string can be 20 and 5 for these two parameters and then the function checks whether the second parameter is a sub string of the first parameter. So, whether s2 is a sub string of s1, the name of the function is substr - sub string and it takes s1 and s2 as the two parameters and need to check and it checks whether s2 is a sub string of s1.

So what will be the equivalence classes here? One is that we have to look at the scenarios that is the first thing we need to do and the scenario is that whether s2 is a sub string of s1, so it will display that it is sub a string. The other scenario is that s2 is not a sub string of s1, so it will display it is not a sub string. And if it needs to display multiple occurrences then we will have that as a scenario, but then it does not tell the problem is not addressing that. Then the next thing comes is about the valid and invalid.

(Refer Slide Time: 24:48)

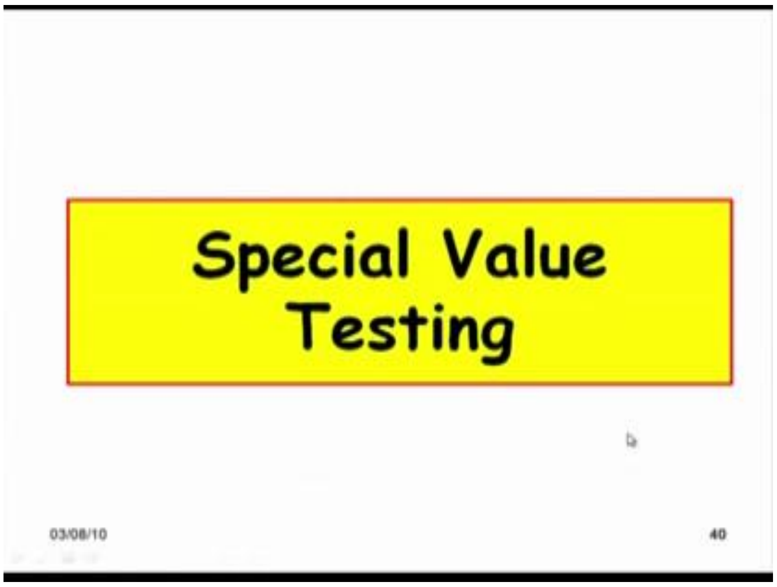


Quiz 3

- Design equivalence class test cases.
  - Consider a program that takes 2 strings of maximum length 20 and 5 characters
  - Checks if the second is a substring of the first
  - substr(s1,s2);

For the first parameter s1, there are two invalid equivalence classes; one is greater than 20 and then invalid character may be a control character. And the second one also two invalid equivalence classes; one is greater than 5 and the other is a control character or may be not a character. We have to consider combination of this and will have our equivalence class test suite ready. So, we will request you to work out this one for the weak equivalence, strong equivalence and the robust testing.

(Refer Slide Time: 25:31)



Special Value Testing

03/08/10

40

Now, let us look at Special Value Testing. In equivalence testing we just look at the input data, we look at the output data, and we look at the scenarios and then we try to find the equivalence classes. If there are multiple parameters then we need to look at the combinations of the equivalence classes. Then once we have done that the actual test case design is simple, we just pick one value from each one. The idea behind equivalence class testing is that we assume that input for one class is processed in similar way compared to any input for the same equivalence class. But then experienced programmers they know some special values where a program might fail.

So let us see what are the special values at which it will fail, we will look at one special value is a boundary and there are other special values where the tester has suspicion that the programmer might not have considered those conditions. Those form the special values. We say that a tester has a knack for knowing where the program will fail and enters only those values. So those are special value, where he believes that the programmer would have made a mistake so that is called a special value testing. We will stop this session now and will continue with special value testing in the next session.

Thank you.