Object-Oriented Programming Fundamentals

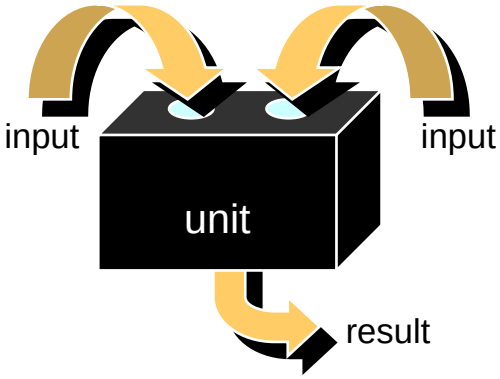
Lecture/Workshop (Week 5)

Testing

Testing strategies can be classified broadly as either *black-box* or *white-box* testing.

Black-box testing focuses on the input(s) to and output(s) from a unit or system to ensure the expected outputs are produced for given inputs.

No attention is paid to the internal working of the unit/system.



White-box testing focuses on the internal workings of the unit/system. The test data is derived from the paths of execution.

In this workshop we will look at two **black-box testing** strategies:

1. Testing each category of input
2. Testing boundary values

Category testing (a black-box testing strategy)

In category testing we decide what kinds of values are to be dealt with and test each category.

For example, the following program reads in an integer value from the keyboard and outputs whether it is positive or non-positive.

```
public static void main(String[] args)
{
   Scanner keyboard = new Scanner(System.in);
   System.out.print("Input an integer value: ");
   int value = keyboard.nextInt();

   if (value > 0)
   {
      System.out.println(value + " is a positive number");
   }
   else
   {
      System.out.println(value + " is a non-positive number");
   }
}
```

In this case we have two categories: positive values and non-positive values. To category test we would then pick a representative value from each category and run the program with it, and check the result against the expected output.

Test Description *Category or Equivalence Class*	**Input value**	**Expected result**	**Actual result**
Positive values category	5	"5 is a positive number" is output to screen	
Non-positive values category	-3	"-3 is a non-positive number" is output to screen	

This form of testing is also called *equivalence partitioning* and the categories called *equivalence classes*.

Task 1

Consider a program which reads a mark between 0 and 100 inclusive from the user and outputs the equivalent grade.

- 80 to 100 inclusive is an 'A' grade
- 70 to 79 inclusive is a 'B' grade
- 60 to 69 inclusive is a 'C' grade
- 50 to 59 inclusive is a 'D' grade
- 0 to 49 inclusive is an 'F' grade
- All other values are invalid

Develop some category test cases for this problem.

Test description	**Input value**	**Expected result**

Boundary value testing

An input value is a boundary value if it is a value at which the program changes behaviour.

In boundary value testing we identify the boundary values and test the program for those values. Boundary value testing usually involves the boundaries between equivalence classes.

For example, the following program reads in an integer value from the keyboard and outputs whether it is positive or non-positive.

```
public static void main(String[] args)
{
   Scanner keyboard = new Scanner(System.in);
   System.out.print("Input an integer value: ");
   int value = keyboard.nextInt();

   if (value > 0)
   {
      System.out.println(value + " is a positive number");
   }
   else
   {
      System.out.println(value + " is a non-positive number");
   }
}
```

In this case we expect the program's behaviour to change at the point between the equivalence class for positive values and the equivalence class for non-positive values.

To boundary test we would then pick boundary values of 1 and 0, and run the program with these values and check the results against the expected outputs.

Test Description *Category or Equivalence Class*	**Boundary value**	**Expected result**	**Actual result**
Positive values category	1 (i.e. lowest positive)	"1 is a positive number" is output to screen	
Non-positive values category	0 (i.e. highest non-positive)	"0 is a non-positive number" is output to screen	

Task 2

Consider a program which reads a mark between 0 and 100 inclusive from the user and outputs the equivalent grade.

- 80 to 100 inclusive is an ‘A’ grade
- 70 to 79 inclusive is a ‘B’ grade
- 60 to 69 inclusive is a ‘C’ grade
- 50 to 59 inclusive is a ‘D’ grade
- 0 to 49 inclusive is an ‘F’ grade
- All other values are invalid

What are the boundary value tests for this program? *Note each boundary value in the table makes up a test case.*

Test Description *Category or Equivalence Class*	**Boundary value**	**Expected result**

Unit testing

In unit testing we test each method in the program separately. Sometimes we write small programs in order to test particular methods or classes during the development process. Such programs are referred to as driver programs (or drivers).

Wanting help?

.Check out the ‘Need Help’ section of LMS for code hub sessions 😊