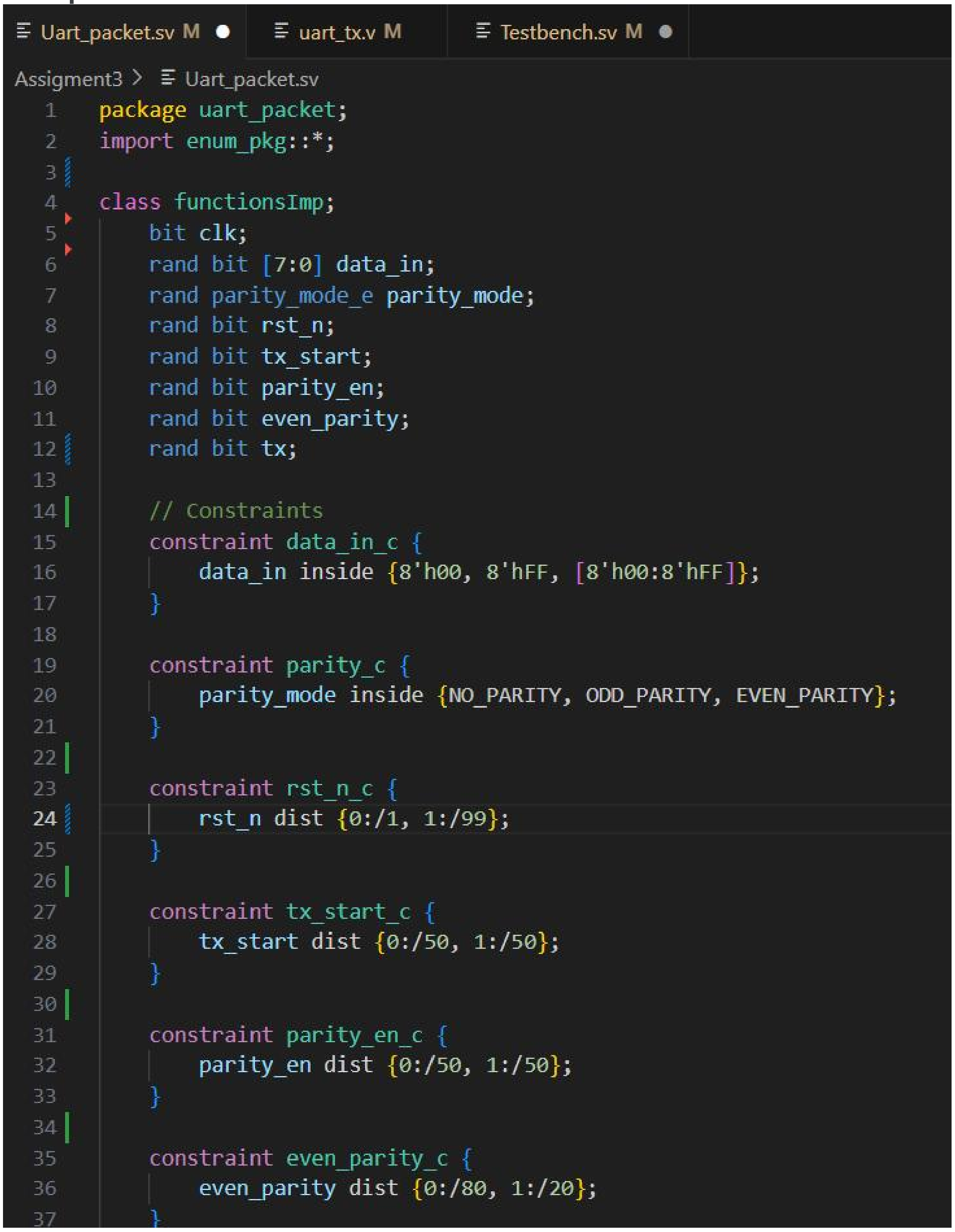
Assigment3:classes

Snippets for the implemented code

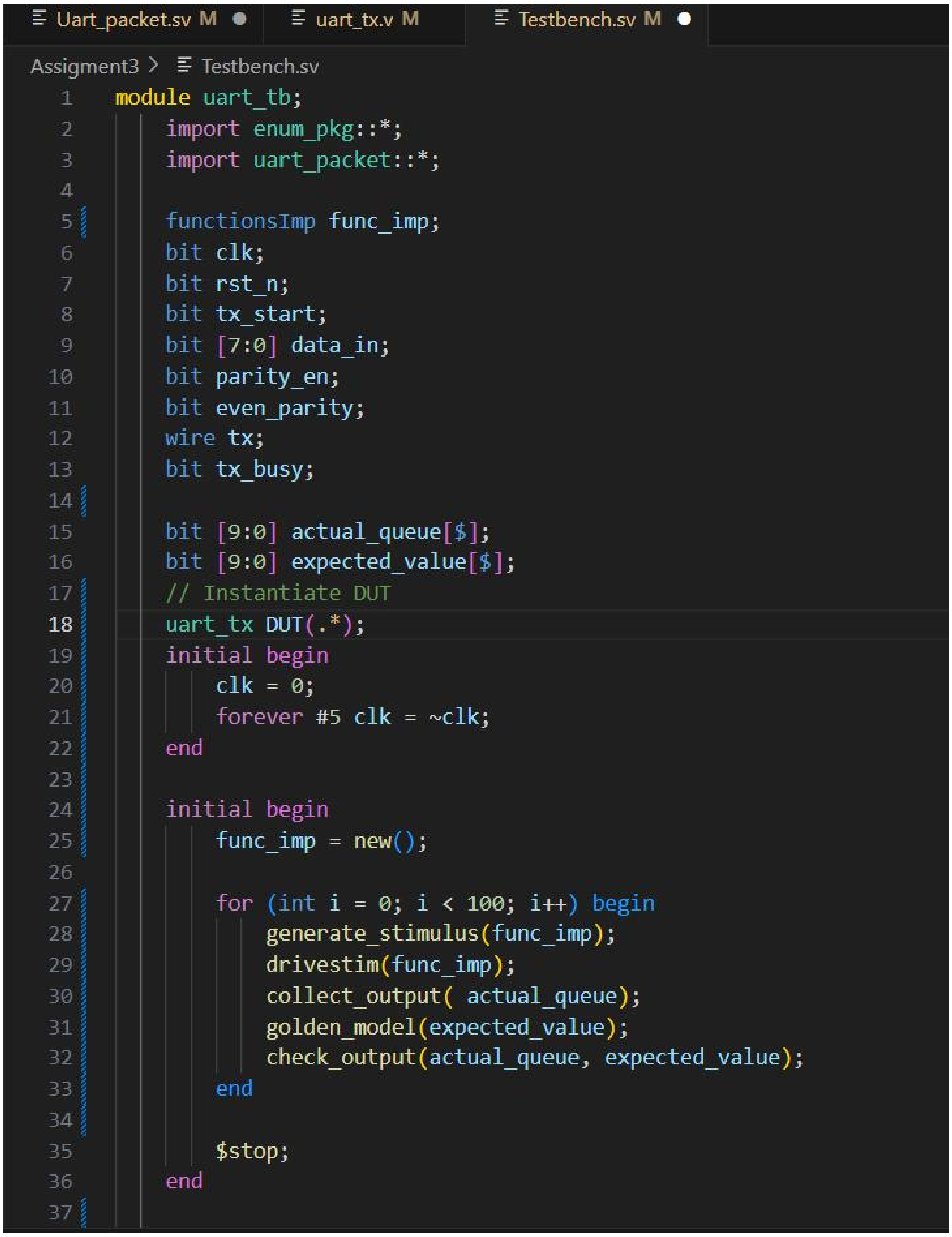
Uart-packet.sv



```
package uart_packet;
import enum_pkg::*;

class functionsImp;
    bit clk;
    rand bit [7:0] data_in;
    rand parity_mode_e parity_mode;
    rand bit rst_n;
    rand bit tx_start;
    rand bit parity_en;
    rand bit even_parity;
    rand bit tx;

    // Constraints
    constraint data_in_c {
        data_in inside {8'h00, 8'hFF, [8'h00:8'hFF]};
    }

    constraint parity_c {
        parity_mode inside {NO_PARITY, ODD_PARITY, EVEN_PARITY};
    }

    constraint rst_n_c {
        rst_n dist {0:/1, 1:/99};
    }

    constraint tx_start_c {
        tx_start dist {0:/50, 1:/50};
    }

    constraint parity_en_c {
        parity_en dist {0:/50, 1:/50};
    }

    constraint even_parity_c {
        even_parity dist {0:/80, 1:/20};
    }
```

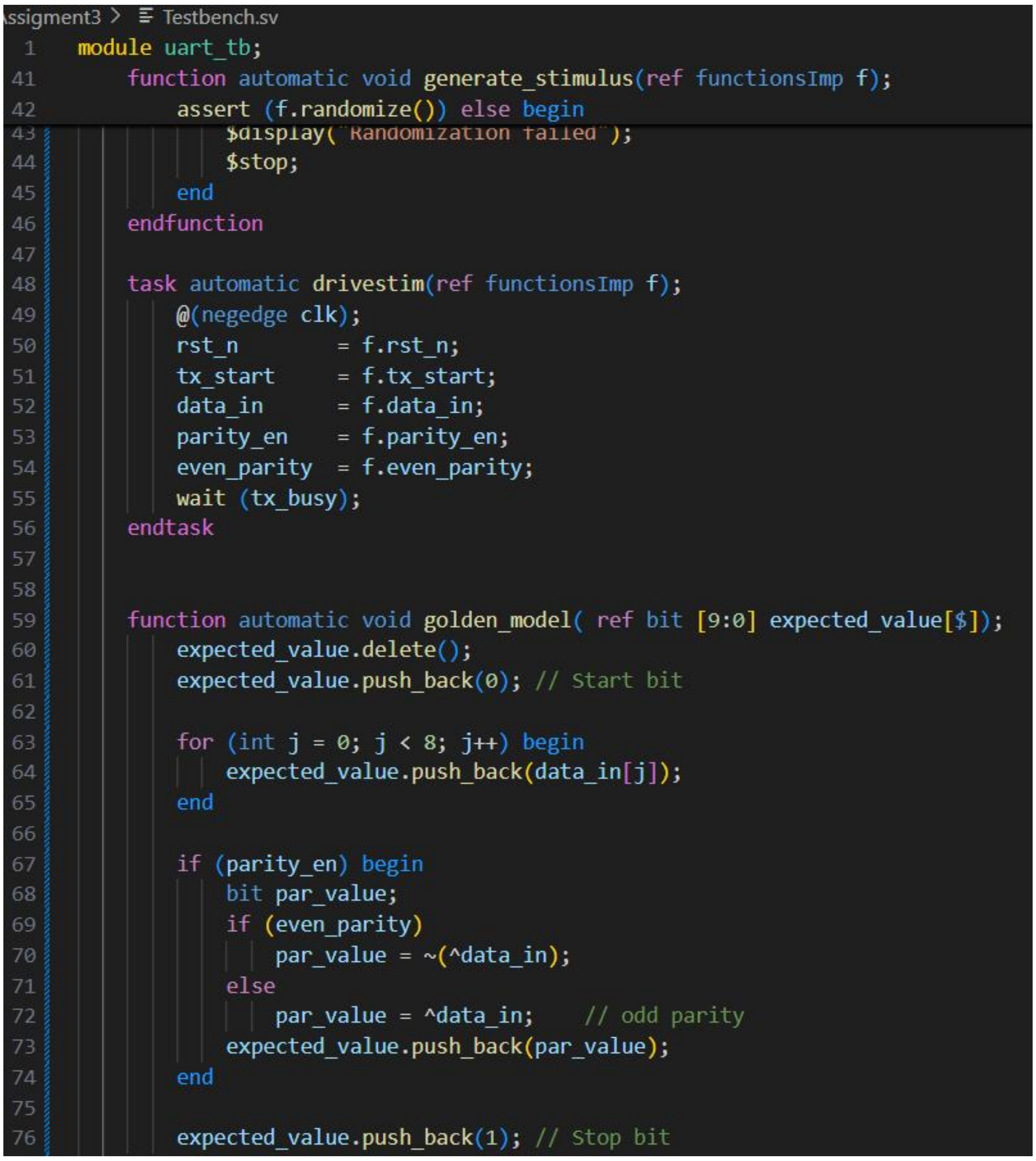
Assigment3:classes

TestBensh.sv



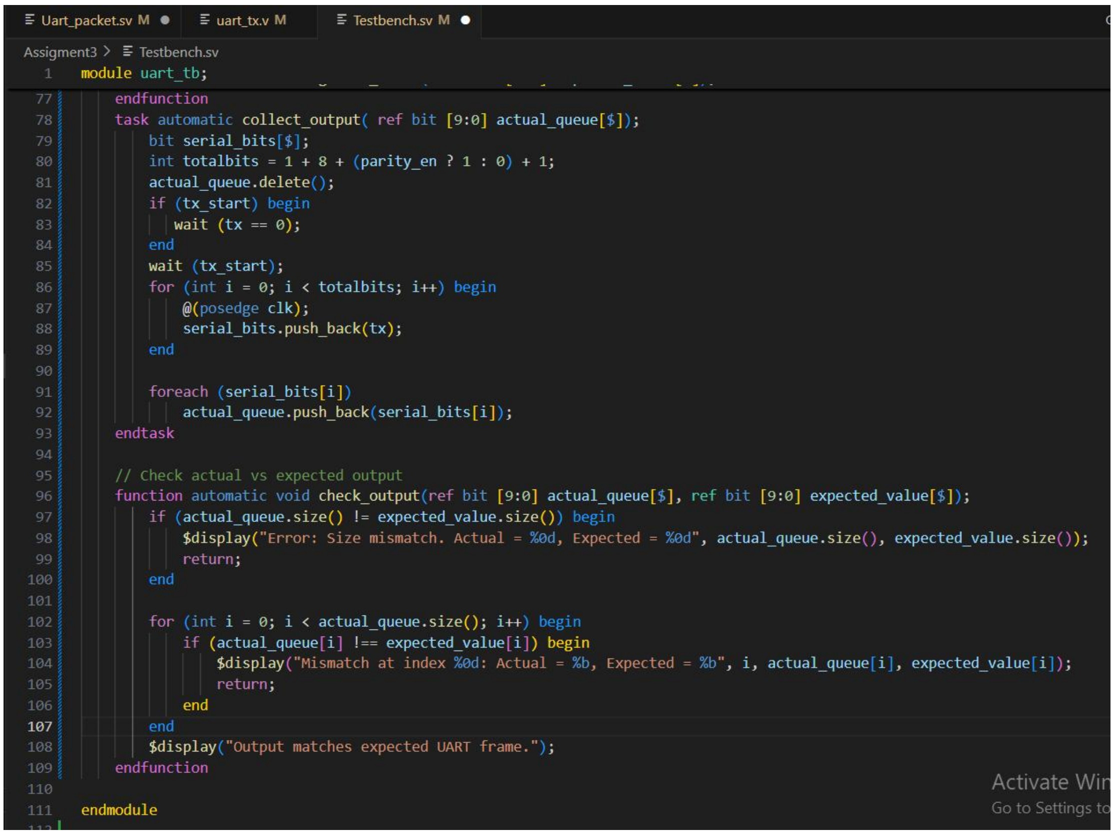
```
module uart_tb;
    import enum_pkg::*;
    import uart_packet::*;

    functionsImp func_imp;
    bit clk;
    bit rst_n;
    bit tx_start;
    bit [7:0] data_in;
    bit parity_en;
    bit even_parity;
    wire tx;
    bit tx_busy;

    bit [9:0] actual_queue[$];
    bit [9:0] expected_value[$];
    // Instantiate DUT
    uart_tx DUT(.*);
    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        func_imp = new();

        for (int i = 0; i < 100; i++) begin
            generate_stimulus(func_imp);
            drivestim(func_imp);
            collect_output( actual_queue);
            golden_model(expected_value);
            check_output(actual_queue, expected_value);
        end

        $stop;
    end
```

Assigment3:classes



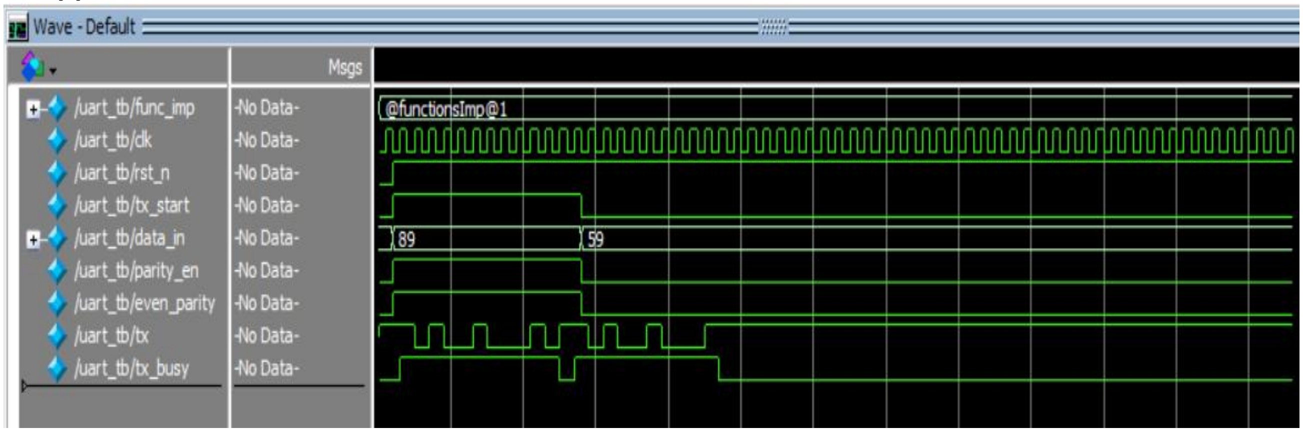
```
ssigment3 > Testbench.sv
module uart_tb;
    function automatic void generate_stimulus(ref functionsImp f);
        assert (f.randomize()) else begin
            $display("Randomization failed");
            $stop;
        end
    endfunction

    task automatic drivestim(ref functionsImp f);
        @(negedge clk);
        rst_n        = f.rst_n;
        tx_start     = f.tx_start;
        data_in      = f.data_in;
        parity_en    = f.parity_en;
        even_parity  = f.even_parity;
        wait (tx_busy);
    endtask


    function automatic void golden_model( ref bit [9:0] expected_value[$]);
        expected_value.delete();
        expected_value.push_back(0); // Start bit

        for (int j = 0; j < 8; j++) begin
            expected_value.push_back(data_in[j]);
        end

        if (parity_en) begin
            bit par_value;
            if (even_parity)
                par_value = ~(^data_in);
            else
                par_value = ^data_in;     // odd parity
            expected_value.push_back(par_value);
        end

        expected_value.push_back(1); // Stop bit
```

Assigment3:classes

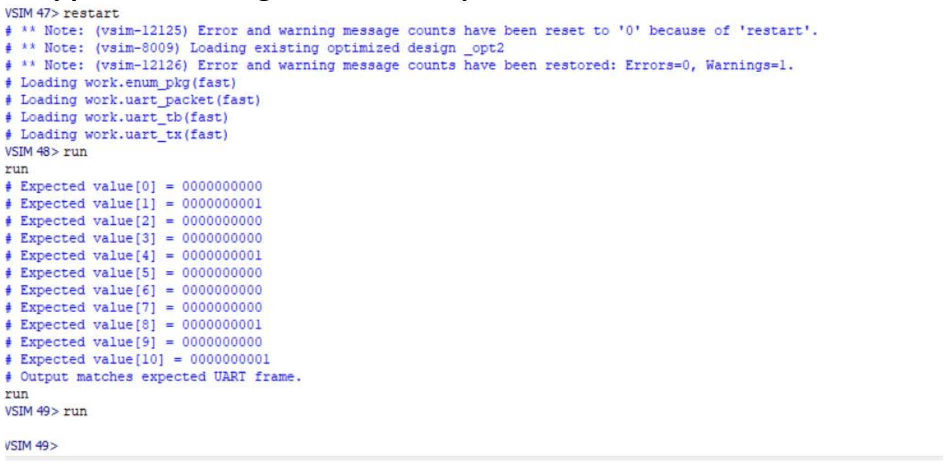


```systemverilog
module uart_tb;
    endfunction
    task automatic collect_output( ref bit [9:0] actual_queue[$]);
        bit serial_bits[$];
        int totalbits = 1 + 8 + (parity_en ? 1 : 0) + 1;
        actual_queue.delete();
        if (tx_start) begin
           wait (tx == 0);
        end
        wait (tx_start);
        for (int i = 0; i < totalbits; i++) begin
            @(posedge clk);
            serial_bits.push_back(tx);
        end

        foreach (serial_bits[i])
            actual_queue.push_back(serial_bits[i]);
    endtask

    // Check actual vs expected output
    function automatic void check_output(ref bit [9:0] actual_queue[$], ref bit [9:0] expected_value[$]);
        if (actual_queue.size() != expected_value.size()) begin
            $display("Error: Size mismatch. Actual = %0d, Expected = %0d", actual_queue.size(), expected_value.size());
            return;
        end

        for (int i = 0; i < actual_queue.size(); i++) begin
            if (actual_queue[i] !== expected_value[i]) begin
                $display("Mismatch at index %0d: Actual = %b, Expected = %b", i, actual_queue[i], expected_value[i]);
                return;
            end
        end
        $display("Output matches expected UART frame.");
    endfunction

endmodule
```

Snippet for the waveform shows the different testcases.



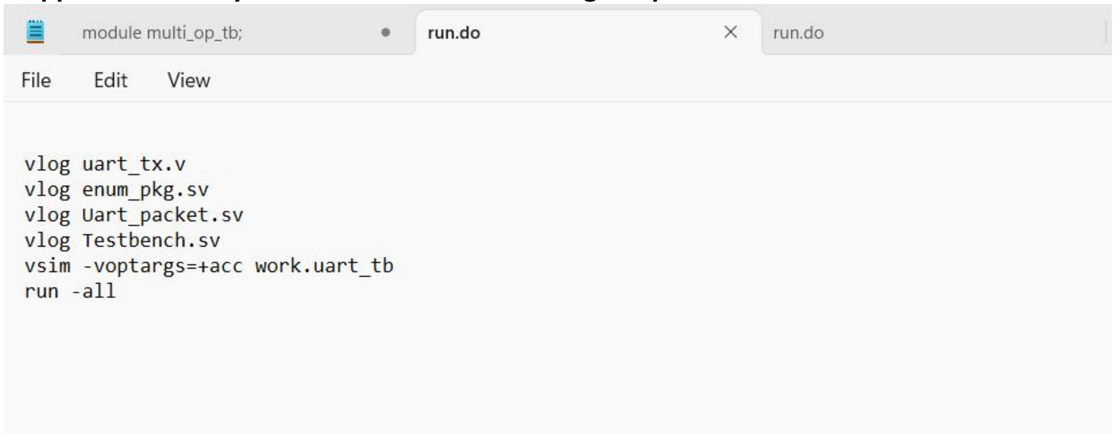
Snippets for the logs show the all printed values.



```
VSIM 47> restart
# ** Note: (vsim-12125) Error and warning message counts have been reset to '0' because of 'restart'.
# ** Note: (vsim-8009) Loading existing optimized design _opt2
# ** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=1.
# Loading work.enum_pkg(fast)
# Loading work.uart_packet(fast)
# Loading work.uart_tb(fast)
# Loading work.uart_tx(fast)
VSIM 48> run
run
# Expected value[0] = 0000000000
# Expected value[1] = 0000000001
# Expected value[2] = 0000000000
# Expected value[3] = 0000000000
# Expected value[4] = 0000000001
# Expected value[5] = 0000000000
# Expected value[6] = 0000000000
# Expected value[7] = 0000000000
# Expected value[8] = 0000000001
# Expected value[9] = 0000000000
# Expected value[10] = 0000000001
# Output matches expected UART frame.
run
VSIM 49> run

VSIM 49>
```

Assigment3:classes

Snippet for Do file you have used for automating the process



```
vlog uart_tx.v
vlog enum_pkg.sv
vlog Uart_packet.sv
vlog Testbench.sv
vsim -voptargs=+acc work.uart_tb
run -all
```