DAY-73
#100DAYSRTL

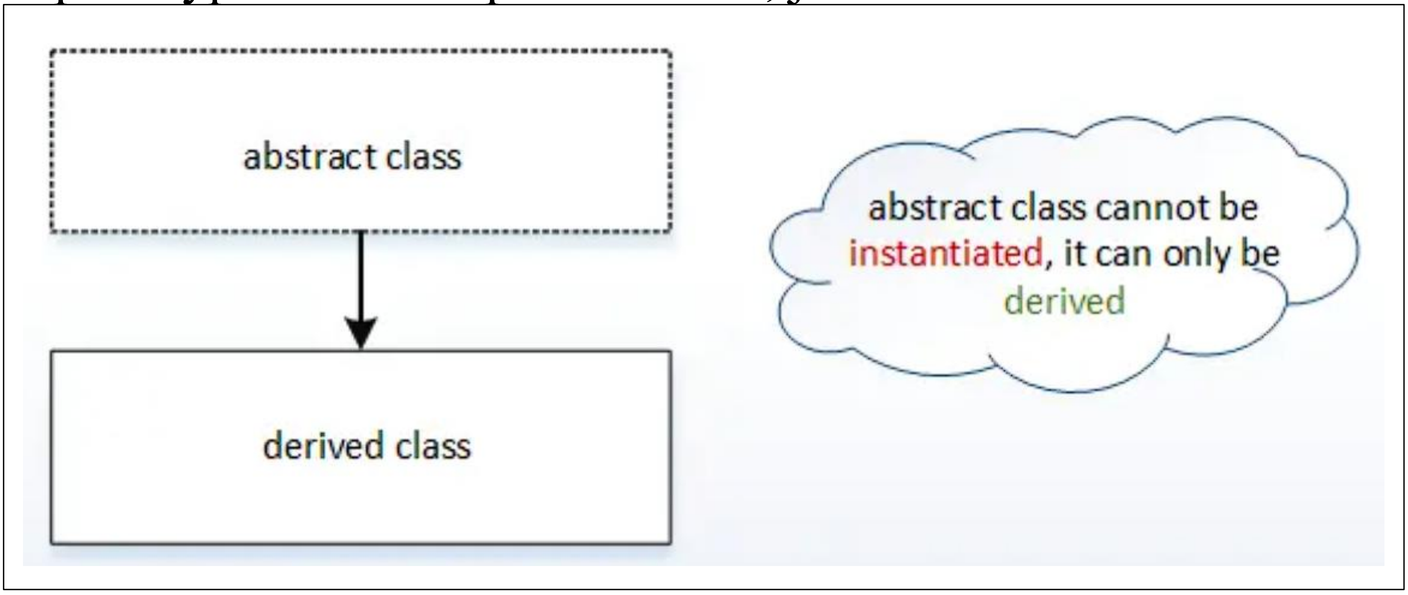
"System Verilog:-Abstraction"

"Introduction":-

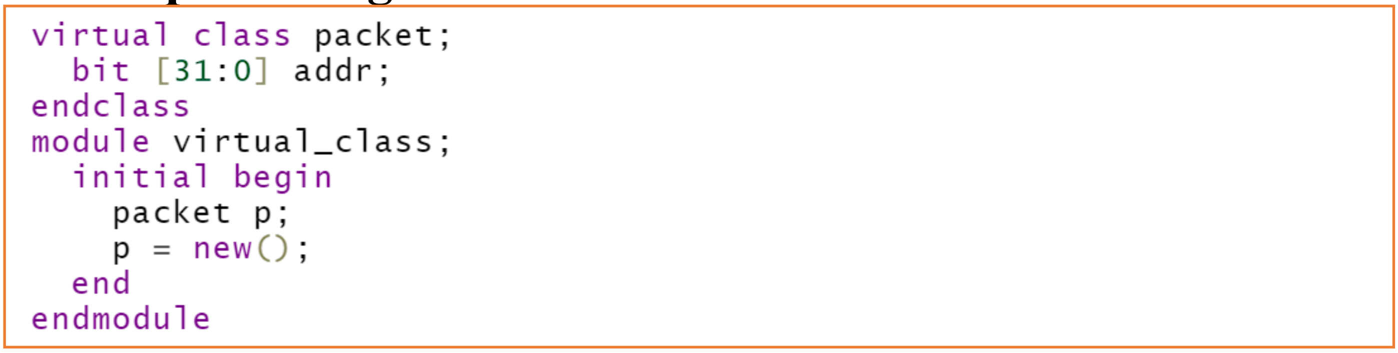
System Verilog has a feature called **virtual** classes that can be used for data abstraction. These classes are like templates that can be used to create other classes. They define common behavior and interfaces for their subclasses but leave some details to be implemented by the subclasses. However, virtual classes cannot be directly used to create objects. Instead, they are used as base classes for other classes. Think of them as a blueprint for creating other classes. The subclasses can then be used to create objects. This way, you can create a hierarchy of classes that share common behaviors and interfaces.

"Abstraction":-

- An abstract class sets out the prototype for the sub-classes.
- An abstract class cannot be instantiated, it can only be derived.
- An abstract class can contain methods for which only a prototype and no implementation, just a method declaration.

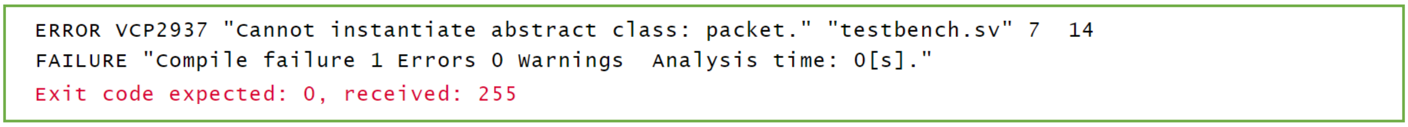

abstract class
derived class
abstract class cannot be instantiated, it can only be derived

“Code practising”:-



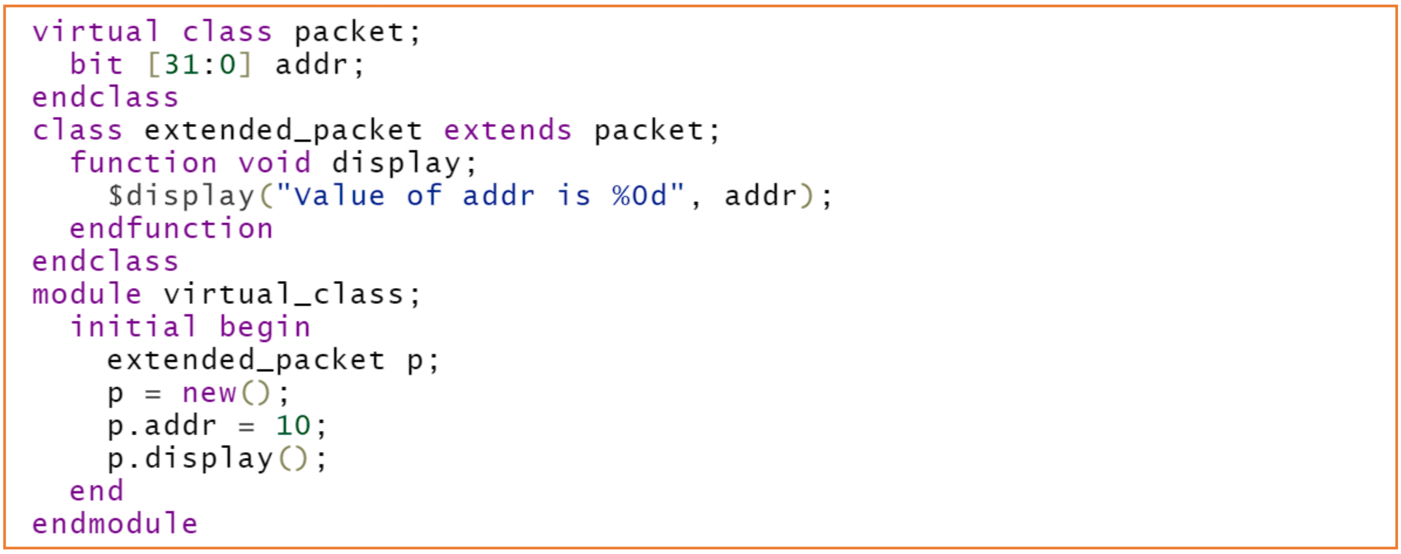
```
virtual class packet;
  bit [31:0] addr;
endclass
module virtual_class;
  initial begin
    packet p;
    p = new();
  end
endmodule
```

“Result”:-



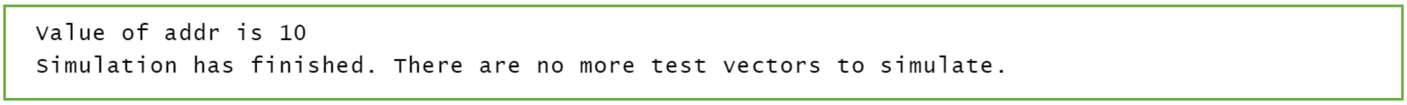
```
ERROR VCP2937 "Cannot instantiate abstract class: packet." "testbench.sv" 7  14
FAILURE "Compile failure 1 Errors 0 Warnings  Analysis time: 0[s]."
Exit code expected: 0, received: 255
```

“Code practising”:-



```
virtual class packet;
  bit [31:0] addr;
endclass
class extended_packet extends packet;
  function void display;
    $display("Value of addr is %0d", addr);
  endfunction
endclass
module virtual_class;
  initial begin
    extended_packet p;
    p = new();
    p.addr = 10;
    p.display();
  end
endmodule
```

“Result”:-

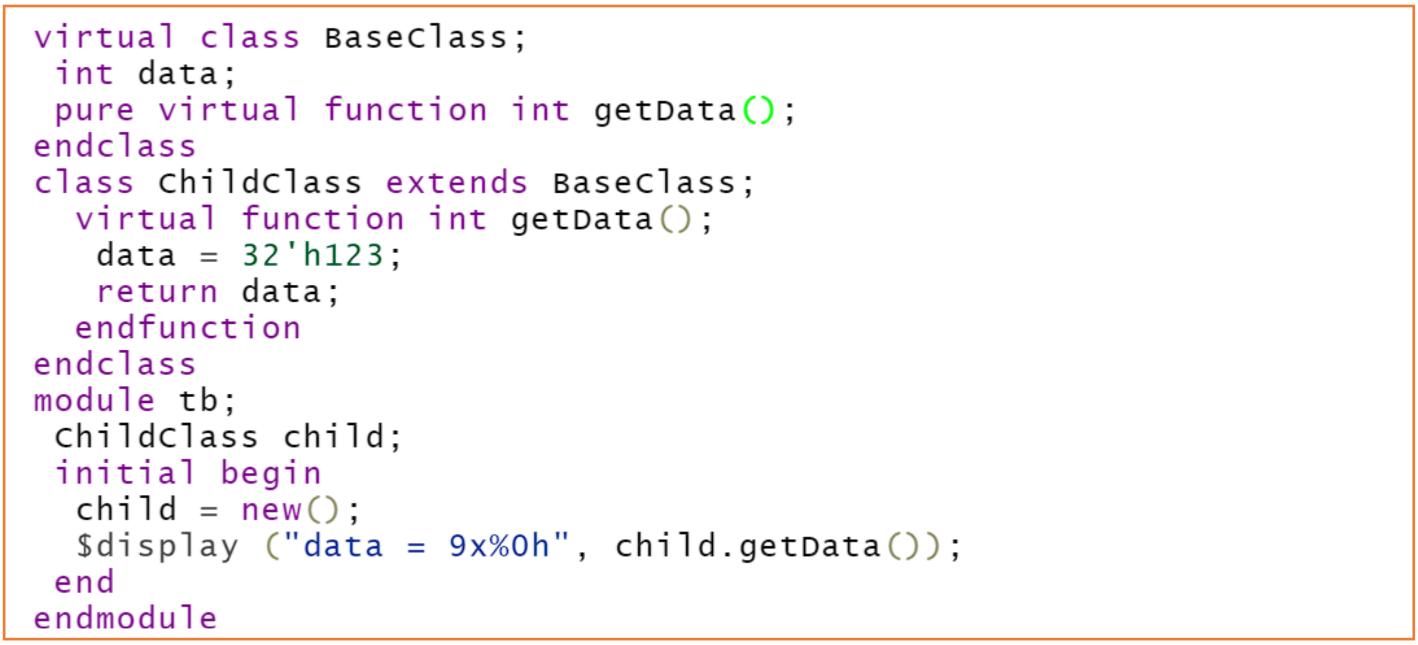


```
Value of addr is 10
Simulation has finished. There are no more test vectors to simulate.
```

“Pure Virtual Method”:-

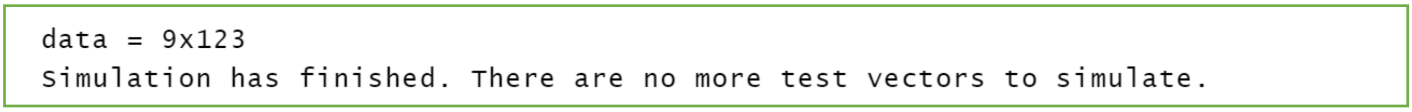
- These are special kinds of method definitions that can only be used inside a virtual class.
- By using a pure keyword with the method definition we do not need to write the body of the method inside the virtual class.
- The method body needs to be defined by all the sub-classes which are derived from an abstract class.
- If a sub-class does not provide an implementation of the pure virtual function then there will be a compilation error.

“Code Practising”:-



```
virtual class BaseClass;
 int data;
 pure virtual function int getData();
endclass
class ChildClass extends BaseClass;
  virtual function int getData();
   data = 32'h123;
   return data;
  endfunction
endclass
module tb;
 ChildClass child;
 initial begin
  child = new();
  $display ("data = 9x%0h", child.getData());
 end
endmodule
```

“Result”:-



```
data = 9x123
Simulation has finished. There are no more test vectors to simulate.
```