



```

1. class System {
2. public:
3.     Producer p;
4.     Consumer c;
5.     FIFO fifo;
6.     void configuration() {
7.         bindPorts(p.pPush, fifo.pPush);
8.         bindPorts(c.pPull, fifo.pPull);
9.     }
10. }
11. class IPull {
12. public: virtual Data* pull() = 0;
13. }
14. class IPush {
15. public:
16.     virtual void push(Data& data) = 0;
17. }
18. class Producer {
19. public: RequiredPort<IPush> pPush;
20. };
21. class Consumer {
22. public: RequiredPort<IPull> pPull;
23. };
24. class FIFO : public IPush, IPull {
25. public:
26.     ProvidedPort<IPush> pPush;
27.     ProvidedPort<IPull> pPull;
28.     Data* pull() { //fine-grained code }
29.     void push(Data& data) { //.. }
30.     //attributes + methods...
31. }
32. StateMachine FIFOMachine {
33.     InitialState Idle;
34.     State SignalChecking {
35.         StateEntry entryCheck;
36.         StateExit exitCheck;
37.     };
38.     State DataQueuing {
39.         StateEntry entryQueue;
40.         State Queuing;
41.     };
42.     State Discarding;
43.     PseudoChoice dataChoice;
44.     CallEvent(DataPushEvent, push(Data&));
45.     TransitionTable {
46.         ExT(Idle, SignalChecking,
47.             DataPushEvent, NULL, signalCheck);
48.         ExT(SignalChecking, dataChoice,
49.             NULL, NULL, NULL);
50.         ExT(dataChoice, Queuing, NULL, valid, NULL);
51.     };
52. void entryCheck() { //fine-grained code }
53. void exitCheck() { //fine-grained code }
54. void entryError() { //fine-grained code }
55. void signalCheck(Data& item) {
56.     //trans effect from Idle to SignalChecking
57. }
58. bool valid() { return isValid && isQueueFull(); }
59. }

```

d