|  |  |
| --- | --- |
| **MediatorDemo.java** | |
| **Line** | **Code** |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52**  **53**  **54**  **55**  **56**  **57**  **58**  **59**  **60**  **61**  **62**  **63**  **64**  **65**  **66**  **67**  **68**  **69**  **70**  **71**  **72**  **73**  **74**  **75**  **76**  **77**  **78** | **using** System;  **using** System.Collections.Generic;  **using** System.Runtime.InteropServices;  **using** System.Text;  **namespace** Mediator {  **class** Program {  /\*  The Mediator maintains a list of colleagues and specifies the  communication methods that it can mediate, in this case, Send.  Receive is implemented at Colleague level and called via a delegate  supplied by the colleagues to the mediator on signon  \*/  **class** Mediator {  **public delegate void** Callback(**string** message, **string from**);  Callback respond;  **public void** SignOn(Callback method) { respond += method; }  **public void** Block(Callback method) { respond -= method; }  **public void** Unblock(Callback method) { respond += method; }  // Send is implemented as a broadcast  **public void** Send(**string** message, **string from**) {  respond(message, **from**);  Console.WriteLine();  }  }  **class** Colleague {  Mediator mediator;  **protected string** name;  **public** Colleague(Mediator mediator, **string** name) {  **this**.mediator = mediator;  mediator.SignOn(Receive);  **this**.name = name;  }  **public virtual void** Receive(**string** message, **string from**) {  Console.WriteLine(name + " received from " + **from** + ": " + message);  }  **public void** Send(**string** message) {  Console.WriteLine("Send (From " + name + "): " + message);  mediator.Send(message, name);  }  }  **class** ColleagueB : Colleague {  **public** ColleagueB(Mediator mediator, **string** name):**base**(mediator, name){  }  // Does not get copies of own messages  **public override void** Receive(**string** message, **string from**) {  **if** (!String.Equals(**from**, name))  Console.WriteLine(name + " received from " + **from** + ": " + message);  }  }  **static void** Main(**string**[] args) {  Console.OutputEncoding = Encoding.UTF8;  Mediator m = **new** Mediator();  Colleague john = **new** Colleague(m, "John");  ColleagueB jack = **new** ColleagueB(m, "Jack Ma");  Colleague akiko = **new** Colleague(m, "Akiko");  john.Send("Meeting on Tuesday, please all ack");  jack.Send("收到"); // by design does not get a copy  m.Block(jack.Receive); // temporarily gets no messages  john.Send("Still awaiting some Acks");  akiko.Send("ありがとう");  m.Unblock(jack.Receive); //open again  john.Send("Thanks all");  Console.ReadKey();  }  }  } |

**Output:**

