/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Delegates and Events: Part 5

\*

\* Core Topics:

\* 1. How to capture return values from multicast delegates.

\* 2. Use GetInvocationList method to loop through all method

\* calls one-by-one that are contained in the multicast

\* delegate.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

usingSystem**;**

namespaceSignalAndWarningSystem

**{**

internaldelegatevoidDisplayMenu**();**

internaldelegatevoidToggleSystemPower**();**

internaldelegateboolManageWater**();**

internaldelegateintRaiseWarning**();**

internaldelegatevoidPowerUpComplete**();**

internalclassHotWaterTransferSystem

**{**

privatebool\_powerOn**;**

publicHotWaterTransferSystem**()**

**{**

PowerOn=false**;**

**}**

internalvoidPowerUp**()**

**{**

// Power up request came in and system is off.

if **(**!PowerOn**)**

**{**

Console.WriteLine**(**"Starting up the system..."**);**

// It takes time to power up the system. Simulate that

// with a pause in the operation.

System.Threading.Thread.Sleep**(**10000**);**

PowerOn=true**;**

Console.WriteLine**(**"System is running."**);**

**}**

else

**{**

// Power up request came in and system is already on.

Console.WriteLine**(**"System is already running."**);**

**}**

**}**

// This is a new PowerUp method that takes two delegates, one

// to call the real PowerUp method and one for the callback

// method.

internalvoidPowerUp**(**ToggleSystemPowerpowerUpProcess**,**

PowerUpCompletecallbackMethod**)**

**{**

// Call the PowerUp process.

powerUpProcess**();**

// Now call the callback method to notify the client code

// that the reading process is done.

callbackMethod**();**

**}**

internalvoidPowerDown**()**

**{**

// Power down request came in and system is on.

if **(**PowerOn**)**

**{**

Console.WriteLine**(**"Shutting down the system..."**);**

// It takes time to power down the system. Simulate that

// with a pause in the operation.

System.Threading.Thread.Sleep**(**7000**);**

PowerOn=false**;**

Console.WriteLine**(**"System is powered down."**);**

**}**

else

**{**

// Power down request came in and system is already off.

Console.WriteLine**(**"System is already shut down."**);**

**}**

**}**

internalboolPowerOn

**{**

get **{** return\_powerOn**; }**

privateset **{** \_powerOn=value**; }**

**}**

internalboolTransferHotWaterOut**()**

**{**

boolresult=true**;**

if **(**PowerOn**)**

**{**

Console.WriteLine**(**"Purging HOT water..."**);**

// Simulate the transfer of hot water out of the system

// with a pause in the operation

System.Threading.Thread.Sleep**(**5000**);**

Console.WriteLine**(**"Hot water transfer complete."**);**

**}**

else

**{**

Console.WriteLine

**(**"System is not on. Hot water tranfer aborted."**);**

result=false**;**

**}**

returnresult**;**

**}**

internalboolTransferColdWaterIn**()**

**{**

boolresult=true**;**

if **(**PowerOn**)**

**{**

Console.WriteLine**(**"Filling COLD water..."**);**

// Simulate the transfer of hot water out of the system

// with a pause in the operation

System.Threading.Thread.Sleep**(**5000**);**

Console.WriteLine**(**"Cold water transfer complete."**);**

**}**

else

**{**

Console.WriteLine

**(**"System is not on. Cold water tranfer aborted."**);**

result=false**;**

**}**

returnresult**;**

**}**

internalintAlertTransferStationFloor**()**

**{**

Console.WriteLine

**(**"\*\*\* FLOOR - The HWT System is malfunctioning!"**);**

return1**;**

**}**

internalintAlertControlRoom**()**

**{**

Console.WriteLine

**(**"\*\*\* CONTROL - The HWT System is malfunctioning!"**);**

return2**;**

**}**

internalintAlertManagement**()**

**{**

Console.WriteLine

**(**"\*\*\* MANAGEMENT - The HWT System is malfunctioning!"**);**

return3**;**

**}**

**}**

classControlRoom

**{**

bool\_exitSystem**;**

HotWaterTransferSystem\_hwtSystem**;**

publicControlRoom**()**

**{**

ExitSystem=false**;**

HWTSystem=newHotWaterTransferSystem**();**

**}**

privateHotWaterTransferSystemHWTSystem

**{**

get **{** return\_hwtSystem**; }**

set **{** \_hwtSystem=value**; }**

**}**

privateboolExitSystem

**{**

get **{** return\_exitSystem**; }**

set **{** \_exitSystem=value**; }**

**}**

privatevoidPowerUpProcessComplete**()**

**{**

Console.WriteLine**(**"CALLBACK: Power Up process is complete."**);**

**}**

privateboolRunOperation**(**stringoperation**)**

**{**

boolsuccess=false**;**

stringsystemOperation=operation.ToUpper**();**

systemOperation=systemOperation.Substring**(**0**,** 1**);**

switch **(**systemOperation**)**

**{**

case"1"**:** // Turn on the system.

if **(**HWTSystem!=null**)**

**{**

// The delegate used for the callback when powerup

// completes.

PowerUpCompletepuc=

newPowerUpComplete**(**PowerUpProcessComplete**);**

ToggleSystemPowertogglePower=

newToggleSystemPower**(**HWTSystem.PowerUp**);**

// Call the PowerUp method that takes the two

// delegates.

HWTSystem.PowerUp**(**togglePower**,** puc**);**

success=true**;**

**}**

break**;**

case"2"**:** // Turn off the system.

if **(**HWTSystem!=null**)**

**{**

ToggleSystemPowertogglePower=

newToggleSystemPower**(**HWTSystem.PowerDown**);**

togglePower**();**

success=false**;**

**}**

break**;**

case"3"**:** // Purge hot water.

if **(**HWTSystem!=null**)**

**{**

ManageWatermanager=

newManageWater**(**HWTSystem.TransferHotWaterOut**);**

if **(**manager**())**

**{**

success=true**;**

**}**

**}**

break**;**

case"4"**:** // Fill cold water.

if **(**HWTSystem!=null**)**

**{**

ManageWatermanager=

newManageWater**(**HWTSystem.TransferColdWaterIn**);**

if **(**manager**())**

**{**

success=true**;**

**}**

**}**

break**;**

case"5"**:** // Correct system errors.

if **(**HWTSystem!=null**)**

**{**

// See if the system is on. If not, turn it on!

if **(**!HWTSystem.PowerOn**)**

**{**

HWTSystem.PowerUp**();**

**}**

// Purge the hot water and then fill with cold water.

ManageWatermanager=

newManageWater**(**HWTSystem.TransferHotWaterOut**);**

if **(**manager**())**

**{**

manager=new

ManageWater**(**HWTSystem.TransferColdWaterIn**);**

if **(**manager**())**

**{**

success=true**;**

**}**

**}**

// Turn the system off.

if **(**HWTSystem.PowerOn**)**

**{**

HWTSystem.PowerDown**();**

**}**

Console.WriteLine**(**"System errors have been fixed "+

"and the system was shutdown successfully."**);**

**}**

break**;**

case"X"**:** // Exit the control program.

ExitSystem=true**;**

success=true**;**

break**;**

default**:**

Console.WriteLine**(**"Menu option {0} is not valid."**,**

operation**);**

break**;**

**}**

returnsuccess**;**

**}**

staticvoidMain**(**string**[]** args**)**

**{**

RaiseWarningwarning=null**;**

boolstatus=true**;**

boolemergencyOccured=false**;**

boolcontrolRoomWarningAdded=false**;**

boolmanagementWarningAdded=false**;**

// Create the DisplayMenu delegate and use an anonymous method

// to contain the code.

DisplayMenumenu=delegate**()**

**{**

Console.WriteLine**();**

Console.WriteLine**(**"Hot Water Transfer System Control Menu"**);**

Console.WriteLine**();**

Console.WriteLine**(**"\t1. Turn on system"**);**

Console.WriteLine**(**"\t2. Turn off system"**);**

Console.WriteLine**(**"\t3. Purge hot water from system"**);**

Console.WriteLine**(**"\t4. Fill system with cold water"**);**

Console.WriteLine**(**"\t5. Correct system errors"**);**

Console.WriteLine**(**"\tX. Exit HWTS control program"**);**

Console.WriteLine**();**

Console.Write**(**"Enter option: "**);**

**};**

// Create the control room object.

ControlRoomcr=newControlRoom**();**

// Continue to run until the user exits the application.

while **(**!cr.ExitSystem**)**

**{**

// Display the control menu.

menu**();**

// Get the option from the user.

stringoption=Console.ReadLine**();**

Console.WriteLine**();**

// Process the option.

status=cr.RunOperation**(**option**);**

if **(**!status**)**

**{**

Console.WriteLine

**(**"WARNING: Is there a problem in the system?"**);**

// If this is the first emergency, create the warning

// delegate and point it to the alert mechanism that

// will notify the people working on the floor with

// the system hardware.

if **(**!emergencyOccured**)**

**{**

emergencyOccured=true**;**

warning=

newRaiseWarning

**(**cr.HWTSystem.AlertTransferStationFloor**);**

**}**

else

**{**

// If the first warning was already posted, add in

// another warning to the control room. Now both

// the floor and the control room will be notified.

if **(**!controlRoomWarningAdded**)**

**{**

controlRoomWarningAdded=true**;**

warning+=

newRaiseWarning

**(**cr.HWTSystem.AlertControlRoom**);**

**}**

elseif

**(**controlRoomWarningAdded&&

!managementWarningAdded**)**

**{**

// At this point, management needs to be notified

// because the warnings have not been cleared

// yet. Add another method to the delegate's list

// of methods to call.

managementWarningAdded=true**;**

warning+=

newRaiseWarning

**(**cr.HWTSystem.AlertManagement**);**

**}**

**}**

// If there are any methods in this delegate's

// list, call them now

if **(**warning!=null**)**

**{**

//warning();

intalertStatus=0**;**

foreach

**(**RaiseWarningwinwarning.GetInvocationList**())**

**{**

alertStatus=w**();**

Console.WriteLine**(**"ALERT STATUS: {0}"**,**

alertStatus**);**

**}**

**}**

**}**

else

**{**

// All warnings have been corrected, or none have

// occured. Clear out the various flags and unload

// the multicast delegate by setting it to null.

emergencyOccured=false**;**

controlRoomWarningAdded=false**;**

managementWarningAdded=false**;**

warning=null**;**

**}**

**}**

**}**

**}**

**}**