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

\*

\* Delegates and Events: Part 3

\*

\* Core Topics:

\* 1. Use a delegate to implement a callback to notify the client

\* when the called method is complete.

\* 2. Identifies the steps involved in setting up and

\* implementing a callback.

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

usingSystem**;**

namespaceSignalAndWarningSystem

**{**

//1. Declare delegates

internaldelegatevoidDisplayMenu**();**

internaldelegatevoidToggleSystemPower**();**

internaldelegateboolManageWater**();**

internaldelegatevoidPowerUpComplete**();**

internalclassHotWaterTransferSystem

**{**

privatebool\_powerOn**;**

publicHotWaterTransferSystem**()**

**{**

PowerOn=false**;**

**}**

//3. Implement target method to be called by the delegate.

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."**);**

**}**

**}**

//2. Define the method that takes the delegate(s) and calls them.

// 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 using the Invoke method of the Delegate class.

powerUpProcess**();**

// Now call the callback method, using the Invoke method of the

// delegate class, 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**;**

**}**

**}**

classControlRoom

**{**

bool\_exitSystem**;**

HotWaterTransferSystem\_hwtSystem**;**

publicControlRoom**()**

**{**

ExitSystem=false**;**

HWTSystem=newHotWaterTransferSystem**();**

**}**

privateHotWaterTransferSystemHWTSystem

**{**

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

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

**}**

privateboolExitSystem

**{**

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

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

**}**

//3. Implement target method to be called by the delegate.

privatevoidPowerUpProcessComplete**()**

**{**

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

**}**

//private void DisplayMenu()

//{

// 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("\tX. Exit HWTS control program");

// Console.WriteLine();

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

//}

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

**{**

boolsuccess=false**;**

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

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

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

**{**

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

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

**{**

//4. Create instance of delegate in the client code.

// The delegate used for the callback when powerup

// completes.

PowerUpCompletepuc=

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

//4. Create instance of delegate in the client code.

ToggleSystemPowertogglePower=

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

//5. Pass the instances of delegates to a method that will call them.

// 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"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**)**

**{**

boolstatus=true**;**

// 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**(**"\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.

//cr.DisplayMenu();

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?"**);**

**}**

**}**

**}**

**}**

**}**