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

\*

\* Delegates and Events: Part 1

\*

\* Core Topics:

\* 1. Declare and use a delegate.

\* 2. Use delegate inference as an alternate way of declaring and

\* loading a method call into a delegate.

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

usingSystem**;**

namespaceSignalAndWarningSystem

**{**

internaldelegatevoidToggleSystemPower**(**stringcontroller**);**

internaldelegateboolManageWater**(**stringcontroller**);**

internalclassHotWaterTransferSystem

**{**

privatebool\_powerOn**;**

publicHotWaterTransferSystem**()**

**{**

PowerOn=false**;**

**}**

// Method called by delegate.

internalvoidPowerUp**(**stringcontrolRoomOperator**)**

**{**

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

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

**{**

Console.WriteLine**(**"{0}: Starting up the system..."**,**

controlRoomOperator**);**

// 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**(**"{0}: System is running."**,**

controlRoomOperator**);**

**}**

else

**{**

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

Console.WriteLine**(**"{0}: System is already running."**,**

controlRoomOperator**);**

**}**

**}**

// Method called by delegate.

internalvoidPowerDown**(**stringcontrolRoomOperator**)**

**{**

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

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

**{**

Console.WriteLine**(**"{0}: Shutting down the system..."**,**

controlRoomOperator**);**

// 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**(**"{0}: System is powered down."**,**

controlRoomOperator**);**

**}**

else

**{**

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

Console.WriteLine**(**"{0}: System is already shut down."**,**

controlRoomOperator**);**

**}**

**}**

internalboolPowerOn

**{**

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

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

**}**

// Method called by delegate.

internalboolTransferHotWaterOut**(**stringcontrolRoomOperator**)**

**{**

boolresult=true**;**

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

**{**

Console.WriteLine**(**"{0}: Purging HOT water..."**,**

controlRoomOperator**);**

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

// with a pause in the operation

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

Console.WriteLine**(**"{0}: Hot water transfer complete."**,**

controlRoomOperator**);**

**}**

else

**{**

Console.WriteLine

**(**"{0}: System is not on. Hot water tranfer aborted."**,**

controlRoomOperator**);**

result=false**;**

**}**

returnresult**;**

**}**

// Method called by delegate.

internalboolTransferColdWaterIn**(**stringcontrolRoomOperator**)**

**{**

boolresult=true**;**

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

**{**

Console.WriteLine**(**"{0}: Filling COLD water..."**,**

controlRoomOperator**);**

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

// with a pause in the operation

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

Console.WriteLine**(**"{0}: Cold water transfer complete."**,**

controlRoomOperator**);**

**}**

else

**{**

Console.WriteLine

**(**"{0}: System is not on. Cold water tranfer aborted."**,**

controlRoomOperator**);**

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**; }**

**}**

privatevoidDisplayMenu**()**

**{**

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**,** stringcontrolOperator**)**

**{**

boolsuccess=false**;**

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

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

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

**{**

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

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

**{**

ToggleSystemPowertogglePower=

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

togglePower**(**controlOperator**);**

success=true**;**

**}**

break**;**

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

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

**{**

// Traditional approach to creating an instance of a delegate.

ToggleSystemPowertogglePower=

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

// Use delegate inference for creating an instance of a delegate

// to simplify code.

//ToggleSystemPower togglePower = HWTSystem.PowerDown;

togglePower**(**controlOperator**);**

success=false**;**

**}**

break**;**

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

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

**{**

ManageWatermanager=

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

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

**{**

success=true**;**

**}**

**}**

break**;**

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

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

**{**

ManageWatermanager=

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

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

**{**

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**;**

stringcontrolOperator**;**

// Create the control room object.

ControlRoomcr=newControlRoom**();**

// Get operator's name.

Console.Write**(**"Enter your name as the operator: "**);**

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

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

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

**{**

// Display the control menu.

cr.DisplayMenu**();**

// Get the option from the user.

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

Console.WriteLine**();**

// Process the option.

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

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

**{**

Console.WriteLine

**(**"{0}: WARNING: Is there a problem in the system?"**,**

controlOperator**);**

**}**

**}**

**}**

**}**

**}**