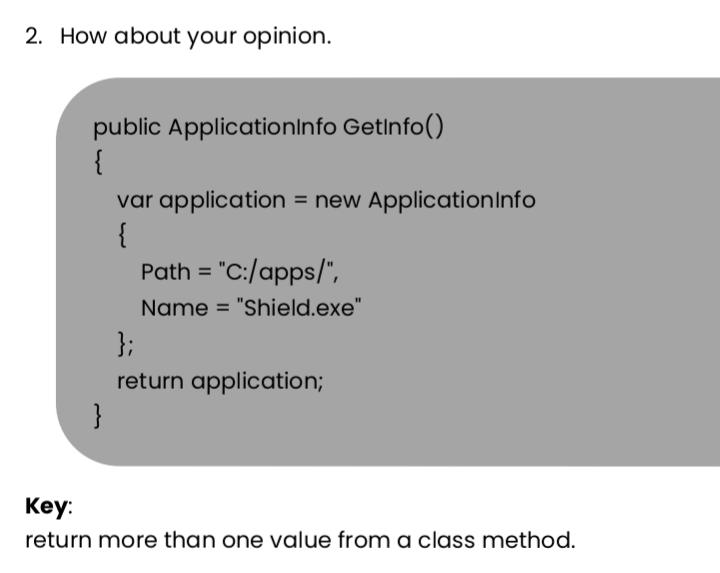
Jawaban :

If( ( aplication != Null ) || ( aplication.protected != Null ))

{ Return aplication.protected.shieldlastrun;}

Jawaban :

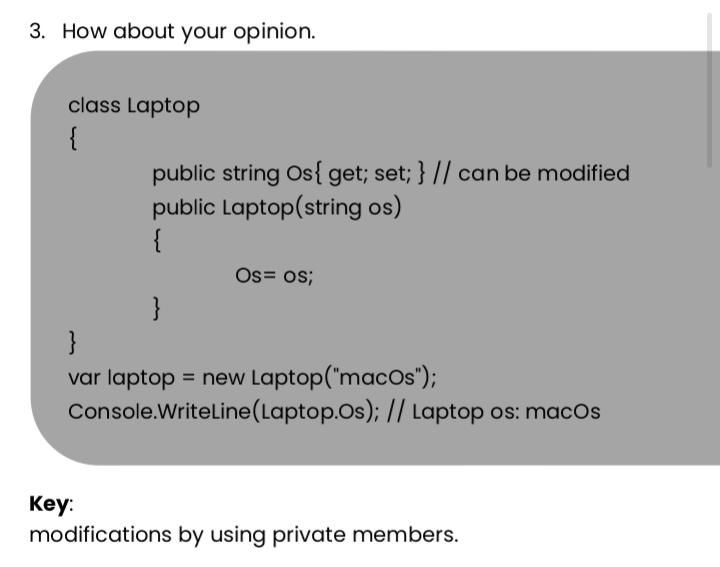
Public ApplicationInfo GetInfo()

{ Var application = new ApplicationInfo;

{ Path = “C:/apps/” ;

Name = “Shield.exe”;}

Return application; }



Jawaban :

Class Laptop

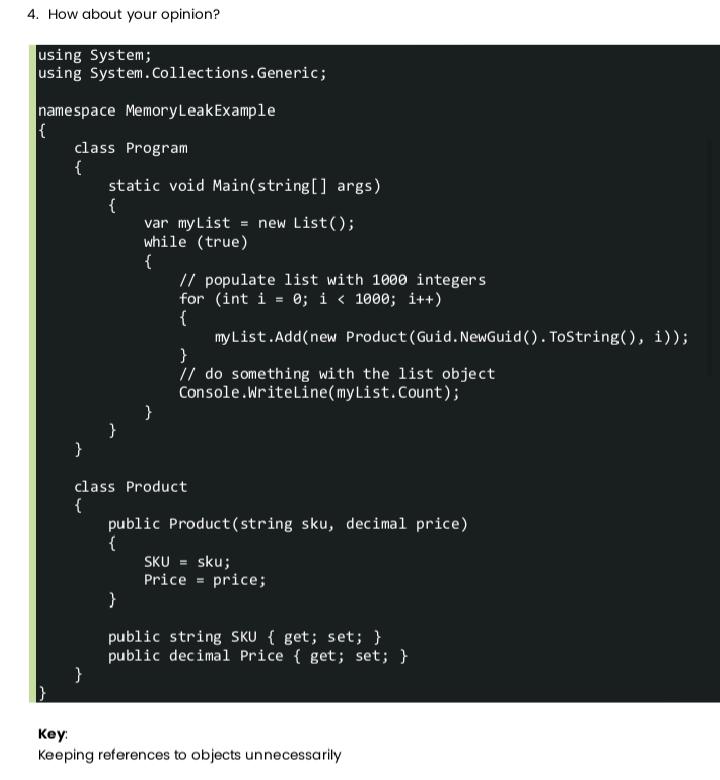
{ Public string Os{ get; set; }

Public Laptop(string os)

{Os= os;}}

Var laptop = new Laptop(“macOs”);

Console.WriteLine(Laptop.Os);



Jawaban :

Using System;

Using System.Collections.Generic;

Namespace MemoryLeakExample

{ Class Program

{ Static void Main(string[] args)

{ Var myList = new List();

While (true)

{

For (int i = 0; i < 1000; i++)

{ myList.Add(new Product(Guid.NewGuid().ToString(), i)); }

Console.WriteLine(myList.Count);

}

}

}

Class Product

{

Public Product(string sku, decimal price) {

SKU = sku;

Price = price;

}

Public string SKU { get; set; }

Public decimal Price { get; set; } } }



Using System;

Namespace MemoryLeakExample

{

Class Program

{

Static void Main(string[] args)

{

Var publisher = new EventPublisher();

While (true)

{Var subscriber = new EventSubscriber(publisher); }

}

Class EventPublisher

{ Public event EventHandler MyEvent;

Public void RaiseEvent()

{ MyEvent?.Invoke(this, EventArgs.Empty); } } Class EventSubscriber

{ Public EventSubscriber(EventPublisher publisher)

{ Publisher.MyEvent += OnMyEvent; }

Private void OnMyEvent(object sender, EventArgs e)

{ Console.WriteLine(“MyEvent raised”); } } } }

Using System;

Using System.Collections.Generic;

Namespace MemoryLeakExample

{

Class Program

{

Static void Main(string[] args)

{

Var rootNode = new TreeNode();

While (true)

{

// create a new subtree of 10000 nodes

Var newNode = new TreeNode();

For (int i = 0; i < 10000; i++)

{

Var childNode = new TreeNode();

newNode.AddChild(childNode);

}

rootNode.AddChild(newNode);

}

}

}

Class TreeNode

{

Private readonly List<TreeNode> \_children = new List<TreeNode>();

Public void AddChild(TreeNode child)

{

\_children.Add(child);

}

}

}



Using System;

Using System.Collections.Generic;

Class Cache

{

Private static Dictionary<int, object> \_cache = new Dictionary<int,

Object>();

Public static void Add(int key, object value)

{

\_cache.Add(key, value);

}

Public static object Get(int key)

{

Return \_cache[key];

}

}

Class Program

{

Static void Main(string[] args)

{

For (int i = 0; i < 1000000; i++)

{

Cache.Add(i, new object());

}

Console.WriteLine(“Cache populated”);

Console.ReadLine();

}

}