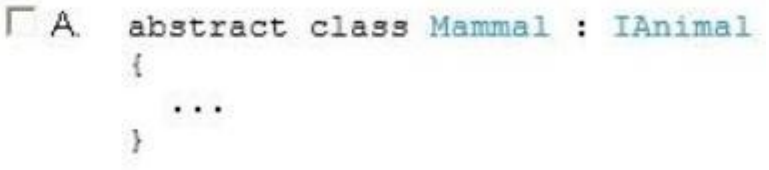
Vendor: Microsoft

Exam Code: 70-483

Exam Name: Microsoft Programming in C#

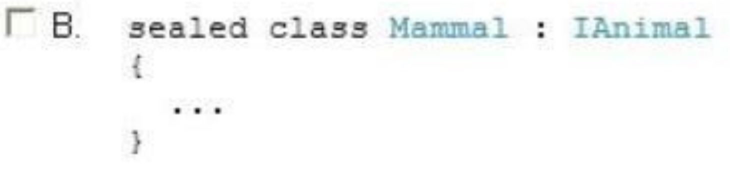
Questions and Answers No.: 41-50 (231Q&As)

QUESTION 41
You are developing an application.
The application includes classes named Mammal and Animal and an interface named IAnimal.
The Mammal class must meet the following requirements:
- It must either inherit from the Animal class or implement the IAnimal interface.
- It must be inheritable by other classes in the application.
You need to ensure that the Mammal class meets the requirements.
Which two code segments can you use to achieve this goal? (Each correct answer presents a complete solution. Choose two.)



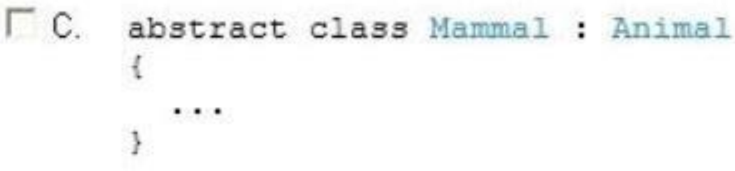
☐ A.

```
abstract class Mammal : IAnimal
{
    ...
}
```



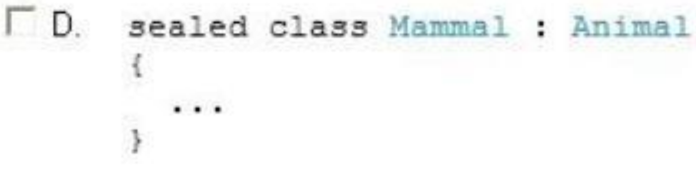
☐ B.

```
sealed class Mammal : IAnimal
{
    ...
}
```



☐ C.

```
abstract class Mammal : Animal
{
    ...
}
```



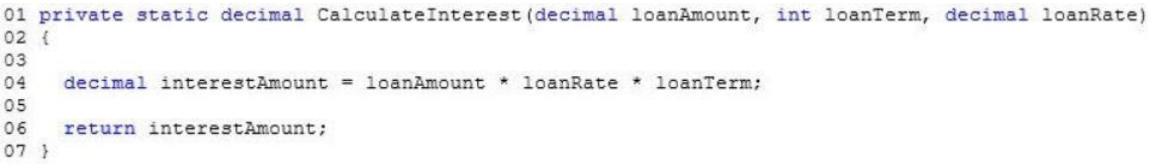
☐ D.

```
sealed class Mammal : Animal
{
    ...
}
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: AC

QUESTION 42
You are debugging an application that calculates loan interest.
The application includes the following code. (Line numbers are included for reference only.)
You have the following requirements:
- The debugger must break execution within the Calculatelnterest() method when the loanAmount variable is less than or equal to zero.
- The release version of the code must not be impacted by any changes.
You need to meet the requirements.
What should you do?



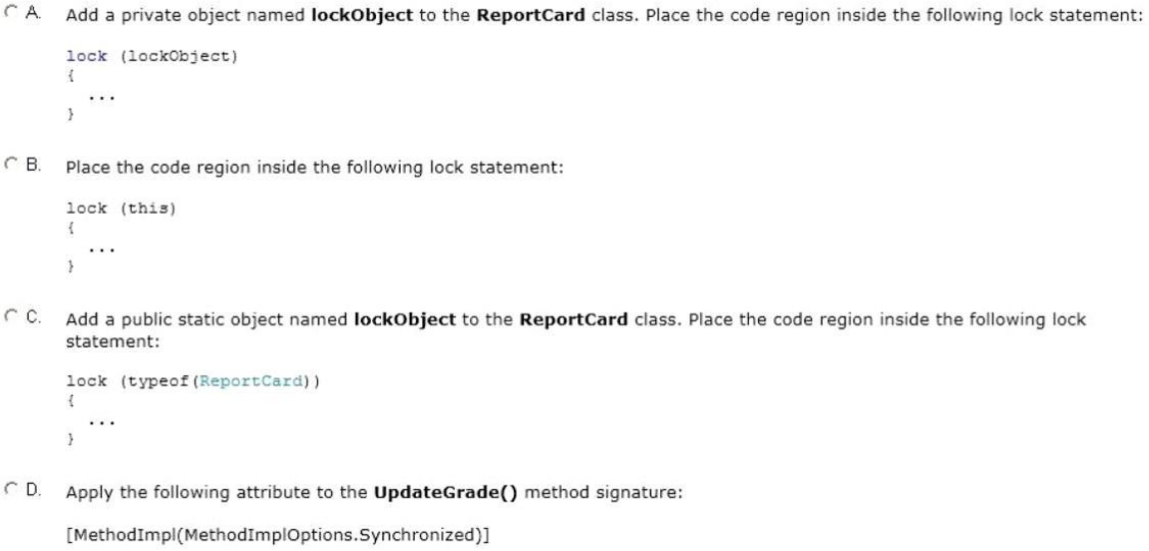
```
01 private static decimal CalculateInterest(decimal loanAmount, int loanTerm, decimal loanRate)
02 {
03
04   decimal interestAmount = loanAmount * loanRate * loanTerm;
05
06   return interestAmount;
07 }
```

A. Insert the following code segment at tine 05:
   `Debug.Write(loanAmount > 0);`
B. Insert the following code segment at line 05:
   `Trace.Write(loanAmount > 0);`
C. Insert the following code segment at line 03:
   `Debug.Assert(loanAmount > 0);`
D. Insert the following code segment at line 03:
   `Trace.Assert(loanAmount > 0);`

Answer: C

QUESTION 43

You are adding a public method named UpdateGrade to a public class named ReportCard.
The code region that updates the grade field must meet the following requirements:
- It must be accessed by only one thread at a time.
- It must not be vulnerable to a deadlock situation.
You need to implement the UpdateGrade() method.
What should you do?



○ A. Add a private object named **lockObject** to the **ReportCard** class. Place the code region inside the following lock statement:

```
lock (lockObject)
{
  ...
}
```

○ B. Place the code region inside the following lock statement:

```
lock (this)
{
  ...
}
```

○ C. Add a public static object named **lockObject** to the **ReportCard** class. Place the code region inside the following lock statement:

```
lock (typeof(ReportCard))
{
  ...
}
```

○ D. Apply the following attribute to the **UpdateGrade()** method signature:

[MethodImpl(MethodImplOptions.Synchronized)]

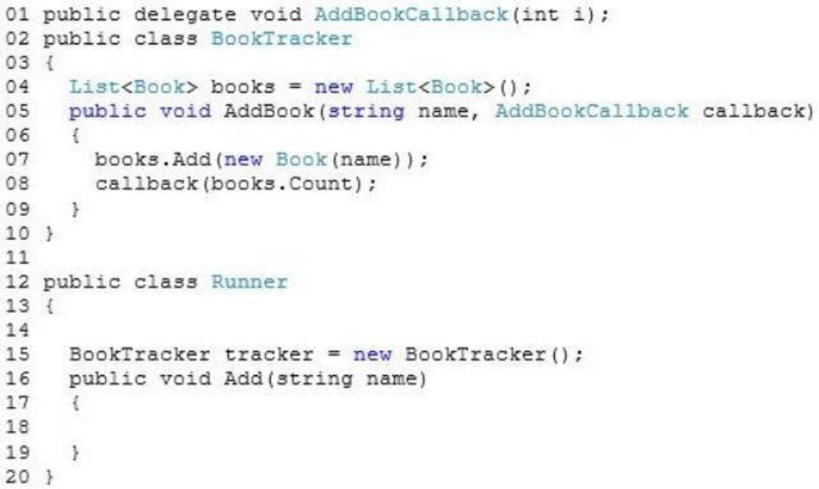
A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

QUESTION 44

You are developing an application that includes a class named BookTracker for tracking library books.

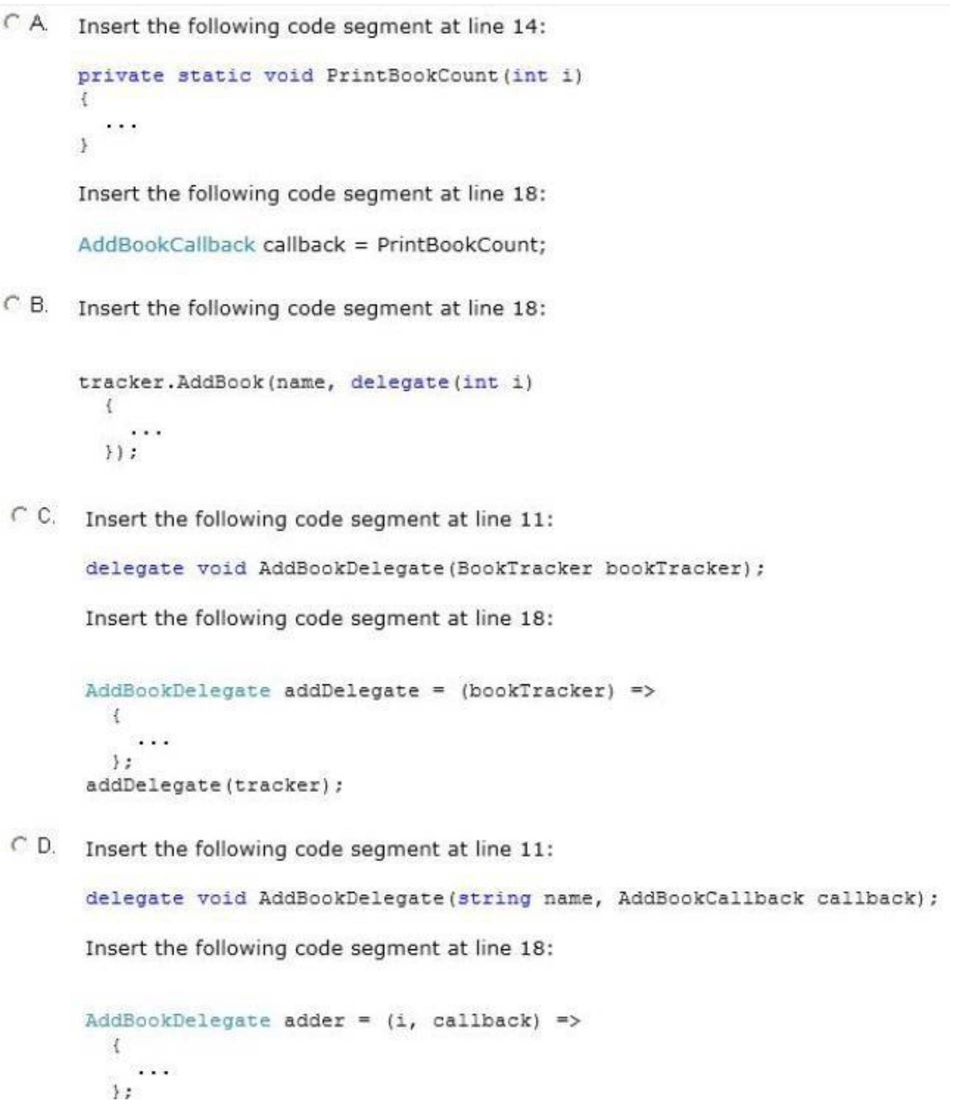
The application includes the following code segment. (Line numbers are included for reference only.)



```
01 public delegate void AddBookCallback(int i);
02 public class BookTracker
03 {
04   List<Book> books = new List<Book>();
05   public void AddBook(string name, AddBookCallback callback)
06   {
07     books.Add(new Book(name));
08     callback(books.Count);
09   }
10 }
11
12 public class Runner
13 {
14
15   BookTracker tracker = new BookTracker();
16   public void Add(string name)
17   {
18
19   }
20 }
```

You need to add a user to the BookTracker instance.

What should you do?



○ A. Insert the following code segment at line 14:

```
private static void PrintBookCount(int i)
{
    ...
}
```

Insert the following code segment at line 18:

```
AddBookCallback callback = PrintBookCount;
```

○ B. Insert the following code segment at line 18:

```
tracker.AddBook(name, delegate(int i)
  {
    ...
  });
```

○ C. Insert the following code segment at line 11:

```
delegate void AddBookDelegate(BookTracker bookTracker);
```

Insert the following code segment at line 18:

```
AddBookDelegate addDelegate = (bookTracker) =>
  {
    ...
  };
addDelegate(tracker);
```

○ D. Insert the following code segment at line 11:

```
delegate void AddBookDelegate(string name, AddBookCallback callback);
```

Insert the following code segment at line 18:

```
AddBookDelegate adder = (i, callback) =>
  {
    ...
  };
```

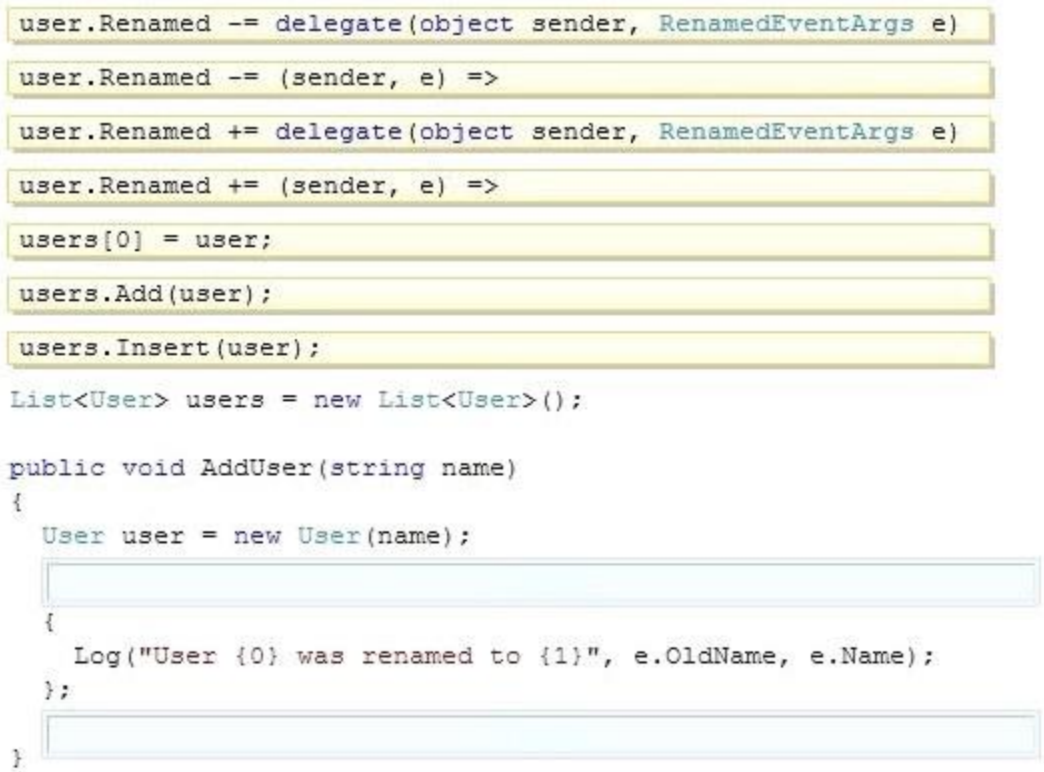
A. Option A
B. Option B
C. Option C
D. Option D

Answer: B

QUESTION 45
Drag and Drop Question
You are implementing a method that creates an instance of a class named User.
The User class contains a public event named Renamed.
The following code segment defines the Renamed event:

```
Public event EventHandler<RenameEventArgs> Renamed;
```

You need to create an event handler for the Renamed event by using a lambda expression.
How should you complete the relevant code? (To answer, drag the appropriate code segments to the correct locations in the answer area. Each code segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.)



```
user.Renamed -= delegate(object sender, RenamedEventArgs e)
user.Renamed -= (sender, e) =>
user.Renamed += delegate(object sender, RenamedEventArgs e)
user.Renamed += (sender, e) =>
users[0] = user;
users.Add(user);
users.Insert(user);
```

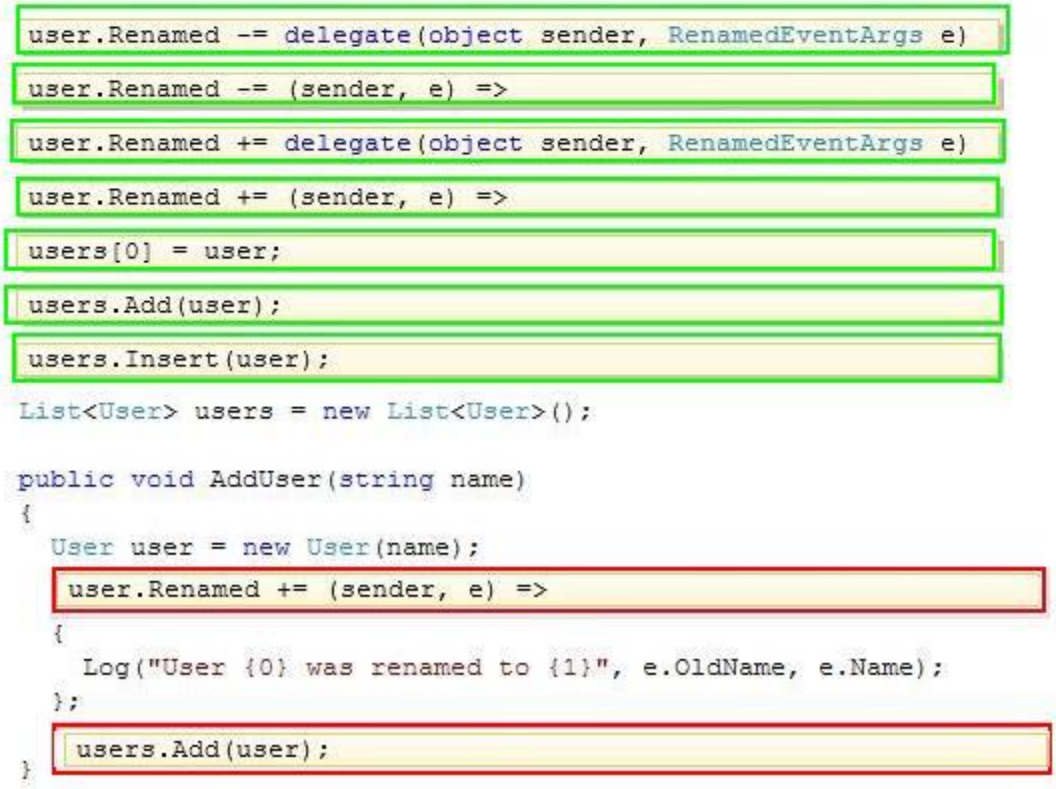
```
List<User> users = new List<User>();

public void AddUser(string name)
{
  User user = new User(name);

  {
    Log("User {0} was renamed to {1}", e.OldName, e.Name);
  };

}
```

Answer:



```
user.Renamed -= delegate(object sender, RenamedEventArgs e)
user.Renamed -= (sender, e) =>
user.Renamed += delegate(object sender, RenamedEventArgs e)
user.Renamed += (sender, e) =>
users[0] = user;
users.Add(user);
users.Insert(user);
```

```
List<User> users = new List<User>();

public void AddUser(string name)
{
  User user = new User(name);
  user.Renamed += (sender, e) =>
  {
    Log("User {0} was renamed to {1}", e.OldName, e.Name);
  };
  users.Add(user);
}
```

QUESTION 46
You are creating a console application by using C#.
You need to access the assembly found in the file named car.dll.
Which code segment should you use?

A. `Assembly.Load ();`
B. `Assembly.GetExecutJingAssembly() ;`
C. `this.GetType ();`
D. `Assembly.LoadFiie("car.dll");`

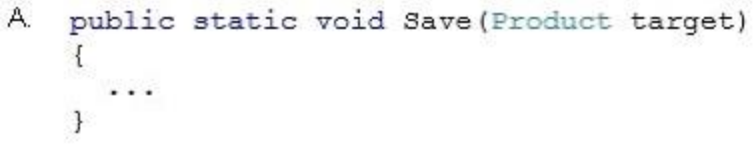
Answer: D

QUESTION 47
You are developing an application by using C#.
The application includes an object that performs a long running process.
You need to ensure that the garbage collector does not release the object's resources until the process completes.
Which garbage collector method should you use?

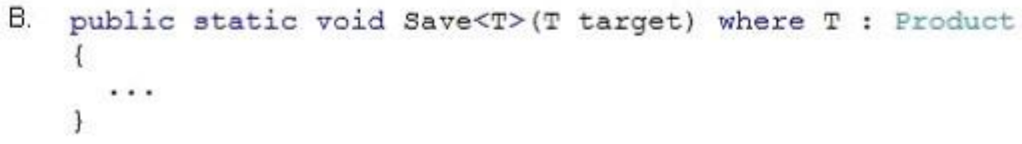
A. WaitForFullGCComplete()
B. WaitForFullGCApproach()
C. KeepAlive()
D. WaitForPendingFinalizers()

Answer: C
Explanation:
The GC.KeepAlive method references the specified object, which makes it ineligible for garbage collection from the start of the current routine to the point where this method is called.
The purpose of the KeepAlive method is to ensure the existence of a reference to an object that is at risk of being prematurely reclaimed by the garbage collector.
The KeepAlive method performs no operation and produces no side effects other than extending the lifetime of the object passed in as a parameter.

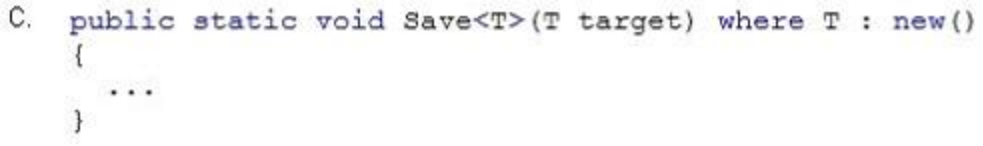
QUESTION 48
You are creating an application that manages information about your company's products.
The application includes a class named Product and a method named Save.
The Save() method must be strongly typed. It must allow only types inherited from the Product class that use a constructor that accepts no parameters.
You need to implement the Save() method.
Which code segment should you use?



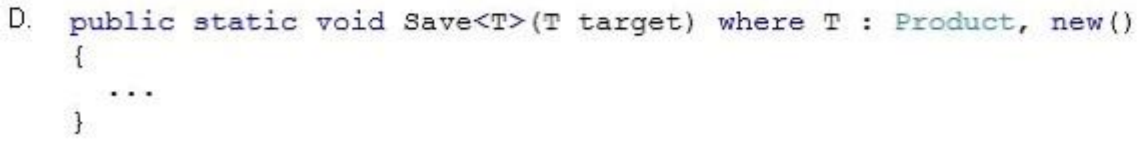
```
A.  public static void Save(Product target)
    {
       ...
    }
```



```
B.  public static void Save<T>(T target) where T : Product
    {
       ...
    }
```



```
C.  public static void Save<T>(T target) where T : new()
    {
       ...
    }
```



```
D.  public static void Save<T>(T target) where T : Product, new()
    {
       ...
    }
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: D

QUESTION 49

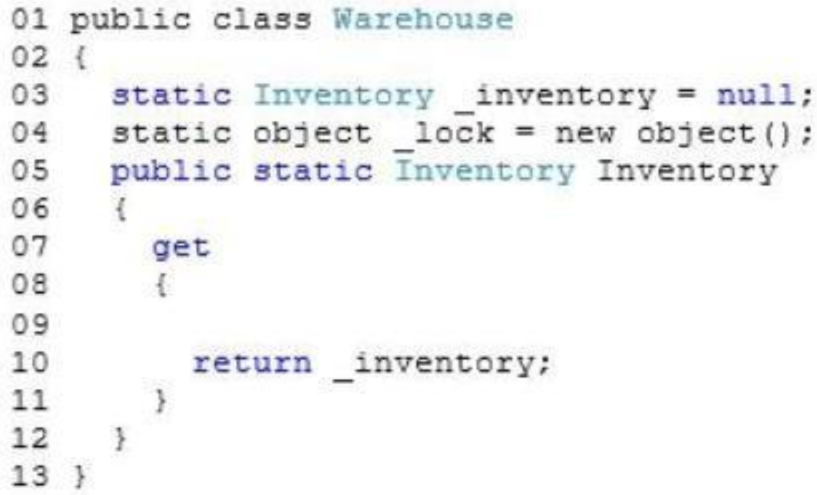
An application includes a class named Person.
The Person class includes a method named GetData.
You need to ensure that the GetData() method can be used only by the Person class and not by any class derived from the Person class.
Which access modifier should you use for the GetData() method?

A. Public
B. Protected internal
C. Internal
D. Private
E. Protected

Answer: D
Explanation:
The GetData() method should be private. It would then only be visible within the Person class.

QUESTION 50

Drag and Drop Question
You are developing an application that includes a class named Warehouse.
The Warehouse class includes a static property named Inventory-
The Warehouse class is defined by the following code segment. (Line numbers are included for reference only.)

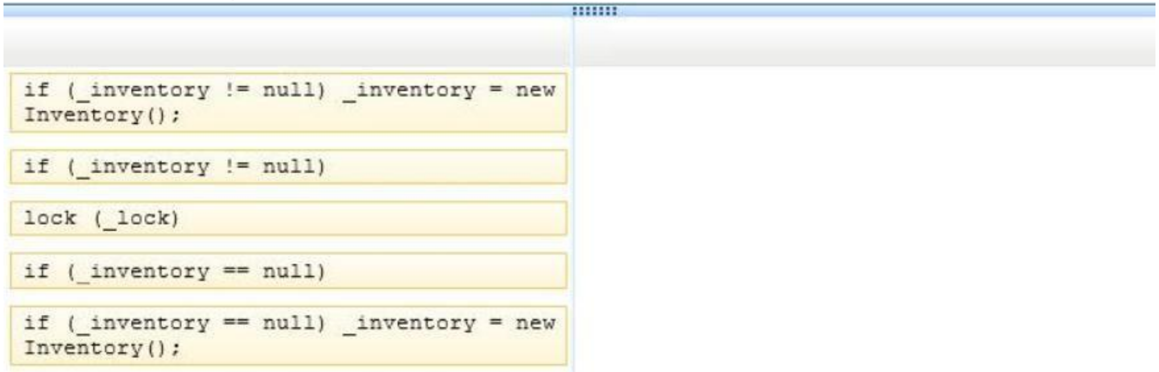


```
01 public class Warehouse
02 {
03    static Inventory _inventory = null;
04    static object _lock = new object();
05    public static Inventory Inventory
06    {
07       get
08       {
09
10          return _inventory;
11       }
12    }
13 }
```

You have the following requirements:

```
- Initialize the _inventory field to an Inventory instance.
- Initialize the _inventory field only once.
- Ensure that the application code acquires a lock only when the
_inventory object must be instantiated.
```

You need to meet the requirements.
Which three code segments should you insert in sequence at line 09? (To answer, move the appropriate code segments from the list of code segments to the answer area and arrange them in the correct order.)



```
if (_inventory != null) _inventory = new
Inventory();
```

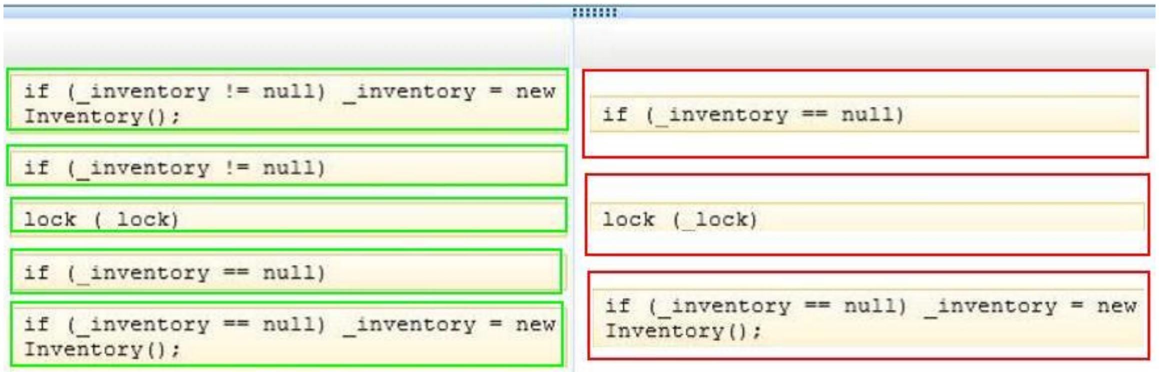
```
if (_inventory != null)
```

```
lock (_lock)
```

```
if (_inventory == null)
```

```
if (_inventory == null) _inventory = new
Inventory();
```

Answer:



```
if (_inventory != null) _inventory = new
Inventory();

if (_inventory != null)

lock ( lock)

if (_inventory == null)

if (_inventory == null) _inventory = new
Inventory();
```

```
if (_inventory == null)

lock (_lock)

if (_inventory == null) _inventory = new
Inventory();
```