

# T estpassport問題集



---

更に上のクオリティ      更に上のサービス

一年で無料進級することに提供する  
[Http://www.testpassport.jp](http://www.testpassport.jp)

**Exam : 70-547(VB)**

**Title : PRO:Design and Develop  
Web-Basd Apps by Using  
MS.NET Frmwk**

**Version : DEMO**

1. You create Web-based applications. You are creating an Internet banking application. The application will be used by bank account holders.

You are creating a method to withdraw money from an account. The method must change the account balance according to one of the following rules:

- If the amount that is being withdrawn is less than or equal to the account balance, then subtract the amount from the balance.

- If the amount that is being withdrawn is greater than the account balance by up to 500 dollars, then subtract the amount and a 35-dollar fee from the balance.

- If the amount that is being withdrawn is greater than the account balance by more than 500 dollars, then generate an error.

You are translating the specification given here into pseudo code. You start by writing the following code.

Method

public void Withdraw

Input parameters

decimal amount

Class field

decimal balance

Pseudo code

//your pseudo code

You need to insert the correct pseudo code.

Which code segment should you insert?

A. If amount < balance then balance - = amount

    If amount < balance + 500 then balance = balance - (amount + 35)

    If amount > balance + 500 then throw exception

B. If amount <= balance then balance - = amount

    If amount <= balance + 500 then balance = balance - (amount + 35)

    If amount > balance + 500 then throw exception

C. If amount < balance then balance - = amount

    Else If amount < balance + 500 then balance = balance - (amount + 35)

Else throw exception

D. If amount <= balance then balance - = amount

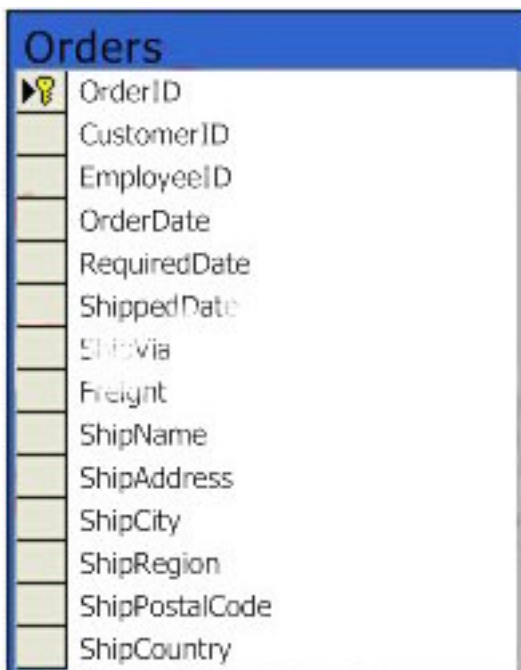
Else If amount <= balance + 500 then balance = balance - (amount + 35)

Else throw exception

**Answer: D**

2. You create Web-based client applications. You create a component named Orders for a company named Northwind Traders.

This component is used to retrieve and update data in the Orders table of the company's database. The schema of the Orders table is as shown in the following Exhibit. (Click the Exhibit button.)



| Orders |                |
|--------|----------------|
| ►      | OrderID        |
|        | CustomerID     |
|        | EmployeeID     |
|        | OrderDate      |
|        | RequiredDate   |
|        | ShippedDate    |
|        | ShipVia        |
|        | Freight        |
|        | ShipName       |
|        | ShipAddress    |
|        | ShipCity       |
|        | ShipRegion     |
|        | ShipPostalCode |
|        | ShipCountry    |

The Orders component permits the client application to perform the following tasks:

- Retrieve all orders for a Customer.
- Retrieve all orders for an Employee.
- Retrieve all orders that are shipped.

An instance of the Order class represents a single order that is identified by the OrderID parameter. An instance of the Order class permits the client application to perform the following tasks:

- Ascertain whether an order is shipped.
- Modify the RequiredDate field of the Order table for an existing order.
- Delete an order.

You need to create the design for the component.

## What should you do?

To answer, drag the appropriate members to the correct locations in the member type column.

[illegible]

**Answer:**

| Members                                                               | Member Type                                                           |
|-----------------------------------------------------------------------|-----------------------------------------------------------------------|
| public static Order[] GetOrdersForCustomer(int CustomerID)            | Methods                                                               |
| public bool OrderWasShipped(int EmployeeID)                           |                                                                       |
| public bool DeleteOrder(int EmployeeID)                               |                                                                       |
| public Order GetOrdersForCustomer(int EmployeeID)                     |                                                                       |
| public bool UpdateOrderDate(DateTime NewRequiredDate)                 |                                                                       |
| public Order GetOrdersForEmployee(int CustomerID)                     |                                                                       |
| public static Order[] UnshippedOrders ()                              |                                                                       |
| public Order UnshippedOrders(int EmployeeID)                          |                                                                       |
| public bool UpdateOrderDate(int EmployeeID, DateTime NewRequiredDate) |                                                                       |
| public bool DeleteOrder()                                             |                                                                       |
| public bool OrderWasShipped                                           |                                                                       |
| public static Order[] GetOrdersForEmployee(int EmployeeID)            |                                                                       |
| public Orders(int OrderID)                                            |                                                                       |
|                                                                       | Property                                                              |
|                                                                       | public Order UnshippedOrders(int EmployeeID)                          |
|                                                                       | public bool UpdateOrderDate(int EmployeeID, DateTime NewRequiredDate) |
|                                                                       | public bool DeleteOrder()                                             |
|                                                                       | public bool OrderWasShipped                                           |
|                                                                       | public static Order[] GetOrdersForEmployee(int EmployeeID)            |
|                                                                       | public Orders(int OrderID)                                            |

3. You create components for Web-based client applications. You are creating a BankAccount class. The BankAccount class contains an AccountNumber property and a CreateAccount method. The CreateAccount method is used to create a new account. The method generates a unique random value for the actNumber field.

You need to ensure that the BankAccount class is extendable, and that it serves as the base class for other derived classes. You also need to ensure that each derived class can have its own guidelines to generate account numbers in the CreateAccount method.

Which code segment should you use?

A. Public Class BankAccount

```
Protected actNumber As Long
Public ReadOnly Property AccountNumber() As Long
    Get
        Return actNumber
    End Get
End Property
Public Overridable Function CreateAccount() As BankAccount
    ...
End Function
End Class
```

B. Public Class BankAccount

```
Private actNumber As Long
Public ReadOnly Property AccountNumber() As Long
    Get
        Return actNumber
    End Get
End Property
Public Overridable Function CreateAccount() As BankAccount
    ...
End Function
End Class
```

C. Public Class BankAccount

```
Protected actNumber As Long
Public ReadOnly Property AccountNumber() As Long
    Get
```

```
        Return actNumber
    End Get
End Property
Public Function CreateAccount() As BankAccount
    ...
End Function
End Class
```

D. Public Class BankAccount

```
    Private actNumber As Long
    Public ReadOnly Property AccountNumber() As Long
        Get
            Return actNumber
        End Get
    End Property
    Public Function CreateAccount() As BankAccount
        ...
    End Function
End Class
```

**Answer: A**

4. You create Web-based client applications. You create a class library that is named Fabrikam.dll. Ten applications will use Fabrikam.dll.

Fabrikam.dll contains two classes that are named Order and OrderDetail. The class library must meet the following requirements:

- Both the classes in Fabrikam.dll are available to client applications of Fabrikam.dll.
- Each instance of the OrderDetail class is associated with an instance of the Order class.
- Code segments in client applications do not instantiate the OrderDetail class.
- The OrderDetail class contains no static members.

You need to design the interface for the OrderDetail class.

Which code segment should you choose?

A. Public NotInheritable Class OrderDetail

```
    Friend Sub New()
    End Sub
```

End Class

B. Friend NotInheritable Class OrderDetail

Friend Sub New()

End Sub

End Class

C. Public NotInheritable Class OrderDetail

Public Sub New()

End Sub

End Class

D. Public NotInheritable Class OrderDetail

Private Sub New()

End Sub

End Class

**Answer: A**

5. You create Web-based client applications. You are creating a class named Product. The Product class will be used by a Web-based application to retrieve and modify product information.

When you create an instance of the Product class, you retrieve the current information from the Products table. The Product class contains a static member named CreateNewProduct. The CreateNewProduct method is used to add a new product to the database and return the primary key. The Products table contains the following fields:

·ProductID (primary key)

·ProductName

·Description

·CategoryID

·CurrentPrice

You need to create the constructor for the Product class.

Which code segment should you use?

A. Public Sub New(ByVal ProductID As Integer, ByVal ProductName As String, ByVal Description As String, ByVal CategoryID As Integer, ByVal CurrentPrice As Decimal)

'Insert code here

End Sub

B. Public Sub New(ByVal ProductID As Integer, ByVal ProductName As String)

'Insert code here

End Sub

C. Public Sub New()

'Insert code here

End Sub

D. Public Sub New(ByVal ProductID As Integer)

'Insert code here

End Sub

**Answer: D**

6. You create Web-based client applications. You are creating a class library that will be used by an e-commerce Web-based application. The library has an abstract class that is named Product. The Product class serves as a base class for the other classes and provides a default ProductID property. Each class other than the base class represents a type of product that is sold by your company. There is a ProductID property and a GetProductDetails procedure for each product type.

You need to ensure that the application meets the following requirements:

- The shopping cart in your Web-based application processes all product types in the same manner.
- Each class retrieves its data from a different source.
- The GetProductDetails procedure retrieves the data from the appropriate source for the product type.

What should you include in the Product class?

- A. a MustOverride ProductID property and an overridable GetProductDetails procedure
- B. an overridable ProductID property and an overridable GetProductDetails procedure
- C. an overridable ProductID property and a MustOverride GetProductDetails procedure
- D. a MustOverride ProductID property and a MustOverride GetProductDetails procedure

**Answer: C**

7. You create Web-based applications. You create a loan application form.

The loan application form is used to calculate the monthly payment of loans. The monthly payment is based on the loan amount, rate, and number of months. The form contains four text boxes and a button. There are no other controls in the form. The application event handler has the following lines of code. (Line numbers are included for reference only.)

```
01 Protected Sub GetPayment(ByVal sender As Object, ByVal e As EventArgs)
02     Try
03         Dim rate As Decimal = Decimal.Parse(txtRate.Text)
04         Dim loanAmount As Decimal = Decimal.Parse(txtLoan.Text)
05         Dim period As Integer = Integer.Parse(txtPeriod.Text)
06         Dim monthlyPayment As Decimal = CalcPayment(rate, loanAmount, period)
07         txtPayment.Text = monthlyPayment.ToString("C")
08     Catch ex As OverflowException
09         ...
10     Catch ex As InvalidCastException
11         ...
12     Catch ex As Exception
13         ...
14     End Try
15 End Sub
```

You must prevent exceptions whenever possible to meet the application requirements.

You need to evaluate the current exception handling mechanism.

What should you conclude?

- A. The current exception handling mechanism meets the requirements. Nothing needs to be changed.
- B. The current exception handling mechanism does not meet the requirements. A required field validator and a range validator control must be added to validate each text box before the button is clicked.
- C. The current exception handling mechanism does not meet the requirements. A required field validator control must be added to validate each text box before the button is clicked.
- D. The current exception handling mechanism does not meet the requirements. A regular expression validator control must be added to validate each text box before the button is clicked.

**Answer: B**

8. You create Web-based client applications. You are reviewing a Web application page that populates the

list of all employees for your company.

The following code segment loads the list of employees from a database.

```
Private Shared Function LoadEmployeesFromDatabase() _
    As List(Of CEmployee)?
    Dim factory As DbProviderFactory = _
        DbProviderFactories.GetFactory("System.Data.SqlClient")
    Dim lstEmployees As List(Of CEmployee) = Nothing
    ' Extract the connection string from configuration data
    Dim connString As ConnectionStringSettings = _
        ConfigurationManager.ConnectionStrings("EmployeeStore")
    ' Create the connection and open it
    Dim conn As DbConnection = factory.CreateConnection()
    conn.ConnectionString = connString.ConnectionString
    conn.Open()
    ' Get the employees. The connection to the database is?
    ' given as parameter

    lstEmployees = GetEmployees(conn)
    ' Close the connection to the employee data store
    conn.Close()

    Return lstEmployees
End Function
```

You analyze the code segment. You find that the database connection fails to close properly when the

GetEmployees method throws an exception.

You need to recommend a change in the code segment to ensure that every possible code path closes the database connection.

Which code segment should you recommend?

A. ' Create the connection and open it

```
Using conn As DbConnection = factory.CreateConnection()  
    conn.ConnectionString = connString.ConnectionString  
    conn.Open()  
    ' Get the employees. The connection to the database  
    ' is given as parameter  
    lstEmployees = GetEmployees(conn)  
End Using
```

B. ' Create the connection and open it

```
Dim conn As DbConnection = factory.CreateConnection()  
conn.ConnectionString = connString.ConnectionString  
conn.Open()  
' Get the employees. The connection to the database is  
' given as parameter  
lstEmployees = GetEmployees(conn)  
If lstEmployees Is Nothing Then  
    conn.Dispose()  
Else  
    conn.Close()  
End If
```

C. Dim coll As HandleCollector = \_

```
New HandleCollector("Connections", 0, 5)  
' Create the connection and open it  
Dim conn As DbConnection = factory.CreateConnection()  
conn.ConnectionString = connString.ConnectionString  
conn.Open()  
coll.Add()  
' Get the employees. The connection to the database is  
' given as parameter  
lstEmployees = GetEmployees(conn)
```

```
' Close the connection to the employee data store
```

```
conn.Close()
```

```
coll.Remove()
```

D. Using factory As IDisposable = \_

```
TryCast(DbProviderFactories.GetFactory(
```

```
"System.Data.SqlClient"), IDisposable)
```

```
Dim conn As DbConnection = factory.CreateConnection()
```

```
conn.ConnectionString = connectionString
```

```
conn.Open()
```

```
' Get the employees. The connection to the database
```

```
' is given as parameter
```

```
lstEmployees = GetEmployees(conn)
```

```
End Using
```

**Answer: A**

9. You create Web-based client applications. You are reviewing a Web application page that populates a list of all employees of your company.

You analyze code and find that the Web application page does not prevent exceptions from traveling to the browser.

You need to ensure that the Web application page intercepts exceptions and presents an error message to the browser.

What change should you suggest?

A. Add the following code segment to the Web.config file.

```
<system.web>
```

```
<compilation debug="true"/>
```

```
</system.web>
```

B. Add the following code segment to the page.

```
Protected Sub Page_Error(ByVal sender As Object, _
```

```
ByVal e As System.EventArgs) Handles Me.Error
```

```
Response.Redirect("error.aspx")
```

```
End Sub
```

C. Add the following code segment to the Web.config file.

```
<system.web>
```

```
<customErrors mode="Off"/>
```

```
</system.web>
```

D. Change the Load event handler to the following code segment.

```
Protected Sub Page_Load(ByVal sender As Object, _  
    ByVal e As System.EventArgs) Handles Me.Load  
    Try  
        LoadEmployees()  
    Catch  
        Response.Redirect("error.aspx")  
    End Try  
End Sub
```

**Answer: B**

10. You create Web-based client applications. You create a Web site that will be used to simulate different types of loans. You are writing a method to calculate the payment on a simple loan.

You write the following lines of code for the method. (Comments are included for reference only.)

```
Public Shared Function Payment(ByVal loanAmount As Decimal, _  
ByVal period As Integer, ByVal rate As Decimal) As Decimal  
    If Not (loanAmount > 0) OrElse _  
        Not (period > 1) OrElse _  
        Not (rate > 0) Then ' Line A  
        Throw New Exception("Invalid input!") ' Line B  
    Else  
        'code to calculate payment  
        Return 877.57D ' Line C: return a calculated payment  
    End If  
End Function  
  
Public Shared Function CheckBalance(ByVal account As ULong) _  
As Decimal  
    Return 877.57D ' Line D: return calculated balance  
End Function
```

You write the following code for the unit test.

```
<TestMethod> _  
Public Sub PaymentTest()  
    Dim payment As Decimal = _  
        Loan.Payment(100000, 360, 10) ' Line E  
    Assert.AreEqual(_payment, 877.57D) ' Line F  
End Sub
```

You enable coverage testing for this unit test.

You need to identify the coverage of your test.

Which lines are covered by the test?

- A. Lines commented A, B, and C
- B. Lines commented A and C
- C. Lines commented A, B, C, D, E, and F
- D. Lines commented A, B, C, E, and F

**Answer: B**