



REVIEW (Chapter 1-3)

C#

OBJECT ORIENTED
PROGRAMMING

Concepts so far...

2

1. .NET Programming Framework & Languages
2. Visual design and event-driven programming
3. C# programming
4. C# program structure, syntax and other
5. C# conditional and loop structure
6. C# object oriented programming
7. Principles of OOP

Exercises-1 (CH3-page 59)

3

1. **Define a class Student**, which contains the following information about students: full name, course, subject, university, e-mail and phone number.
2. **Declare several constructors** for the class Student, which have different lists of parameters (for complete information about a student or part of it).
 - Data, which has no initial value to be initialized with null.
 - Use nullable types for all non-mandatory data.
3. **Add a static field** for the class Student, which holds the number of created objects of this class.
4. Add a method in the class Student, which **displays complete information about the student**.
5. Modify the current source code of Student class so as to encapsulate the data in the class **using properties**.
6. **Write a class StudentTest**, which has to test the functionality of the class Student.

4. Add a method in the class Student, which **displays complete information about the student.**
5. Modify the current source code of Student class so as to encapsulate the data in the class **using properties.**
6. **Write a class StudentTest**, which has to test the functionality of the class Student.
7. **Add a static method in class StudentTest**, which creates several objects of type Student and store them in static fields. Create a static property of the class to access them. Write a test program, which displays the information about them in the console.

Exercise-2 (CH3-page 103)

5

- Define an abstract **class Shape** with abstract method CalculateSurface() and fields width and height.
- Define two additional classes for a **triangle** and a **rectangle**, which implement CalculateSurface(). This method has to return the areas of the rectangle (height*width) and the triangle (height*width/2).
- Define a class for a **circle** with an appropriate constructor, which initializes the two fields (height and width) with the same value (the radius) and implement the abstract method for calculating the area.
- Create three different shapes (rectangle, triangle, circle) and calculate the area of each shape.

Exercise-1: Invoice Class

6

- Create a class called **Invoice** that a hardware store might use to represent an invoice for an item sold at the store.
 - An Invoice should include **four pieces of information** as instance variables: a part number (type String), a part description (type String), a quantity of the item being purchased (type int) and a price per item (double).
 - Your class should have a **constructor** that initializes the four instance variables.
 - Provide properties for each instance variable.
 - In addition, provide a method named **getInvoiceAmount** that calculates the invoice amount
 - (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0.
- Write a test application named **InvoiceTest** that demonstrates class Invoice's capabilities.

Exercise-2: Savings Account Class

7

- Create class **SavingsAccount**.
 - Use a static variable annualInterestRate to store the annual interest rate for all account holders. Each object of the class contains a private instance variable savingsBalance indicating the amount the saver currently has on deposit.
 - Provide method **calculateMonthlyInterest** to calculate the monthly interest by multiplying the savingsBalance by annualInterestRate divided by 12—this interest should be added to savingsBalance.
 - Provide a static method **modifyInterestRate** that sets the annualInterestRate to a new value.
- Write a program to test class SavingsAccount.
 - Instantiate two savingsAccount objects, saver1 and saver2, with balances of \$2000.00 and \$3000.00, respectively.
 - Set annualInterestRate to 4%, then calculate the monthly interest for each of 12 months and print the new balances for both savers.
 - Next, set the annualInterestRate to 5%, calculate the next month's interest and print the new balances for both savers.

10/22/2024

Exercise-3 Identify errors? (If any)

8

```
class A
{
    private int var1A;
    public int Var1A
    {
        get{return var1A;}
        set{var1A=value;}
    }
}
```

```
class B
{
    public int squareB()
    {
        int var1B=var1A*var1A;
        return var1B;
    }
}
```

- Identify and correct the errors ?

Exercise-4 Identify errors? (If any)

9

```
class A
{
    protected int var1A;
    public A(int v1)
    {
        .....
        Console.WriteLine("Hello from A");
    }
}
class B:A
{
    int var1B;
    public B()
    {
        .....
        Console.WriteLine("Hello from B");
    }
}
```

Identify and correct the errors?

10/22/2024

Exercise-5 Identify errors? (If any)

10

```
abstract class A
{
    protected int var1A;
    public abstract int method1A()
    {
        Console.WriteLine("This is method from A");
    }
}
class B:A
{
    int var1B;
    public int method1A()
    {
        Console.WriteLine("This is method from B");
    }
}
```

- Identify and correct the errors?

10/22/2024

Exercise-6 – Output?

11

```
class B
{
    public static void SM() { Console.WriteLine("Hello from B.SM()"); }
    public virtual void VIM() { Console.WriteLine("Hello from B.VIM()"); }
    public void NIM() { Console.WriteLine("Hello from B.NIM()"); }
}

class C:B
{
    public static new void SM() { Console.WriteLine("Hello from C.SM()"); }
    public override void VIM() { Console.WriteLine("Hello from C.VIM()"); }
    public new void NIM() { Console.WriteLine("Hello from C.NIM()"); }
}
```

B b = new C();

b.VIM() ①

c.VIM() ③

C c = new C();

b.NIM() ②

c.NIM() ④

10/22/2024

Exercise-7: Employee Class

12

- Create a class called **Employee** that includes three instance variables: a first name (type String), a last name (type String) and a monthly salary (double).
 - Provide a **constructor** that initializes the three instance variables.
 - Provide a properties for each instance variable.
 - If the monthly salary is not positive, do not set its value.
- Write a test application named **EmployeeTest** that demonstrates class Employee's capabilities.
 - Create two Employee objects and display each object's yearly salary.
 - Then give each Employee a 10% raise and display each Employee's yearly salary again

Exercise-8: Date Class

13

- Create a class called **Date** that includes three instance variables: a month (type int), a day (type int) and a year (type int).
 - Provide a **constructor** that initializes the three instance variables and assumes that the values provided are correct.
 - Provide a properties for each instance variable.
 - Provide a method **displayDate** that displays the month, day and year separated by forward slashes (/).
- Write a test application named **DateTest** that demonstrates class Date's capabilities.

Questions?

Thank You