# CS 371 Windows Application Development

# HW2

* Background Material: Chapters 4 – 7 & Study Slides

## MY NAME: DUE: Thurs, Jan 5 Received: .

**GRADE:**

|  |  |  |
| --- | --- | --- |
| **CATEGORY** | **POINTS** |  |
| EX2\_01 |  | 10 |
| EX2\_02 |  | 10 |
| EX2\_03 |  | 10 |
| EX2\_04 |  | 10 |
| EX2\_05 |  | 20 |
| EX2\_06 |  | 20 |
|  |  |  |
| **TOTAL** |  | 80 |

**EX02\_01**: Modify class Account (Fig 4.17) to provide a method called *Debit* that withdraws money from an *Account*. Ensure that the debit amount does not exceed the Account’s balance. If it does, the balance should be left unchanged and the method should display a message indicating “Debit amount exceeded account balance.” Modify class *AccountTest* (Fig 4.18) to test method *Debit*.

**EX02\_02**: 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 either instance variables or automatic properties – a part number (type *string*), a part description (type *string*), a quantity of the item being purchased (type *int*) and a price per item (*decimel*). Your class should have a constructor that initializes the four values. Provide a property with a get and set accessor for any instance variables. For the *Quantity* and *PricePerItem* properties, if the value passed to the set accessor is negative, the value of the instance variable should be left unchanged. In addition, provide a method named *GetInvoiceAmount* that calculates the invoice amount (i.e. multiples the quantity by the price per item) then returns the amount as a decimal value. Write a test application named *InvoiceTest* that demonstrates class Invoice’s capabilities.

**EX02\_03**: Create a class called *Employee* that includes three pieces of information as either instance variables or automatic properties – a first name (type *string*), a last name (type *string*) and a monthly salary (type *decimal*). Your class should have a constructor that initializes the three values. Provide a property with a get and set accessor for any instance variables. If the monthly salary is negative, the set accessor should leave the instance variable unchanged. 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.

**EX02\_04**: Create a class called *Date* that includes three pieces of information as automatic properties – a month (type *int*), a day (type *int*) and a year (type *int*). Your class should have a constructor that initializes the three automatic properties and assumes that the values provides are correct. 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.

**EX02\_05**: Write an application that plays “*Guess the number*” as follows: Your application chooses the number to be guessed by selecting a **random** integer in the range 1 to 1000. The application displays the prompt “*Guess a number between 1 to 1000*”. The player inputs a first guess. If the player’s guess is incorrect, your application should display “*Too high. Try again.*” Or “*Too low. Try again.*”. The application should prompt the user for the next guess. When the user enters the correct answer, display “*Congratulations! You guessed the number.*” and allow the user to choose whether to play again.

**EX02\_06**: Modify the application in **EX02\_05** to count the number of guesses the player makes. If the number is 10 or fewer, display “*Either you know the secret or you got lucky!*”. If the player guesses the number in 10 tries, display “*Aha! You know the secret!*” If the player makes more than 10 guesses, display “*You should be able to do better!*”

**Consider why should this game take no more than 10 guesses?**