CIDM 4360/5360 Assignment #5: Coding Classes relationships

Total Points: 100 (10% of the course's final grade)

Note: CIDM5360 students have extra tasks in this assignment

Due: Fri Dec 6, 11:59pm

Objective: Assess your understanding of Class diagrams, and implementing relationships between classes in C#.

Skills needed: given a Class Diagram, you'll need to Know how to:

1- **How to read the class diagram**
2- **Implement the classes in C#**
   a. **Define the class**
   b. **Define the attributes with appropriate access levels**
   c. **Define the methods with appropriate access levels**
   d. **Create new Class from base class using Inheritance**
   e. **Declare abstract class, and abstract methods**
   f. **Override base class virtual and abstract methods**

3- **Identify the relationships, their types, and their multiplicities**
4- **Implement the relationships in the each participating class according to its type and multiplicity**
5- **[CIDM5360] Create object diagram form a given code**

Requirements & Submission:

1. You need to submit complete, clean, error-free and runnable C# code (zip the whole folder of your code)
2. You **must** add the **names** of the group members and the course code (CIDM4360 or CIDM5360) to the beginning of each source code file
3. Submit your compressed file using the link in WTClass ("Resources >> Assignments >> Assignment#5)

Rules and conditions:

Your code should comply with the following rules:

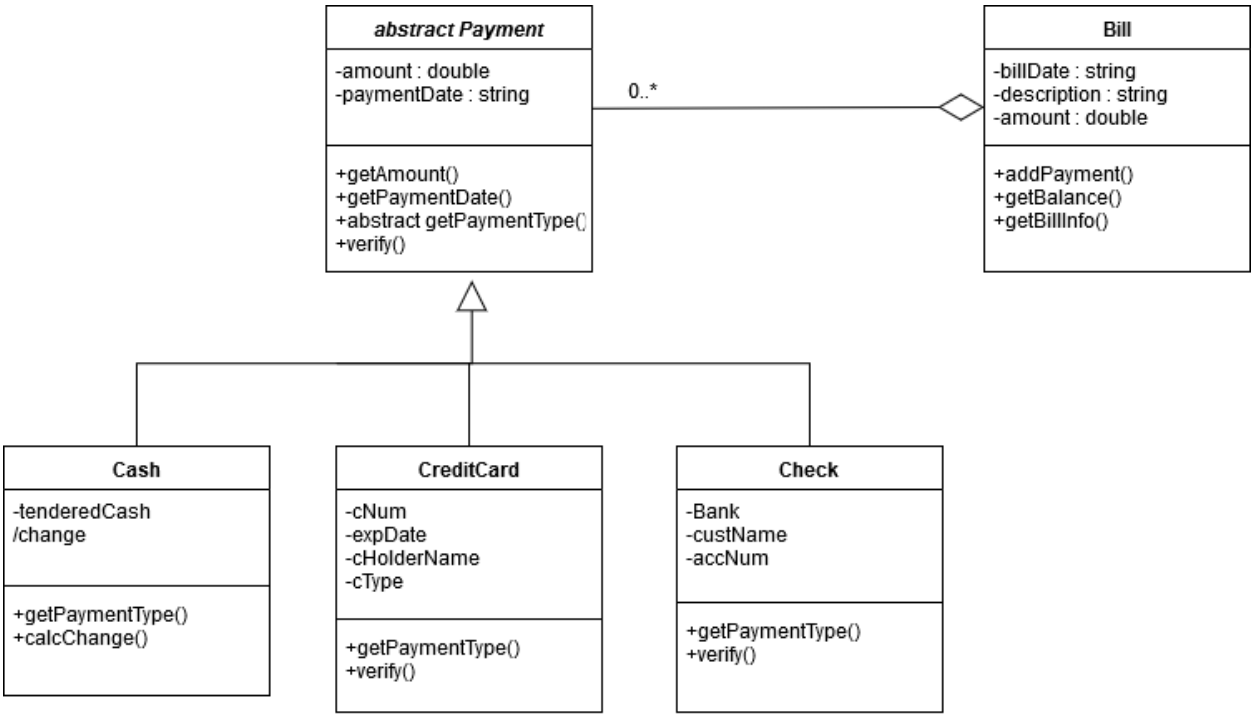
1- Don't change the given code, i.e. should work with the given code as is. I'm going to replace the main program class in your submission with the code provided, and if your code didn't work you will miss points.
2- Must implement only the classes in the given class diagram (don't add any extra classes)
3- Must implement the needed constructor(s), and call the base class constructors when needed.
4- Do not add attributes form other classes inside any of the classes
5- Do not add any new attributes to any of the classes except those you need to setup the relationships with other classes.

6- Must implement all the attributes and methods with the correct access levels (using visibility symbols in the class diagram)
7- Must implement the relationships correctly based on their types and multiplicities
    - Note that the class diagram doesn't show the attributes needed to setup the relationships between the classes.

8- No Data inputs is needed from the console. The main program has everything you need

Description:

Using the following class diagram, which has five classes "***Bill***","***Payment***", "***Cash***" , "***CreditCard***" and "***Check***", and using the ***main program C# code*** that uses these classes (*attached with this assignment in WTClass*), complete the code by **implementing** these five classes and their relationships to accomplish the functionalities in the main program.



Tasks:

Task A:[CIDM 4360 and CIDM 5360] Using the provided code as base of your code, complete the code by implementing the five classes in the diagram above and their relationships to accomplish the functionalities in the given main program.

Classes descriptions:

1-Class Bill:

Attributes:

```
string billDate;
string description ;
double amount; // bill total amount
```

Methods:

- At least one Constructor: see the main() method to understand what constructor parameters are needed
- string **getBillInfo**(): returns string contains all information in the bill including the payments. See the main() method and sample output to see what information is returned and then printed.
- double **getBalance**(): calculate the remaining balance by calculating the difference between the total of payments and the bill's original amount
- bool **addPayment**(Payment p): It calls verify() method first, and if verify is successful it adds the payment to the bill payments.
-

2-Class Payment: Abstract class Used to serve as a base class for other classes

Attributes:

```
double amount // amount of this payment
string paymentDate // date of this payment
```

Methods:

- At least one Constructor to be called by derived classes
- double **getAmount**(): returns the payment amount attribute
- string **getPaymentDate**(): returns the paymentDate attribute
- virtual bool **verify**():writes to the console the message "Payment verified", and returns true. Need to be overloaded by derived classes if needed.
- abstract string **getPaymentType**(): has no implementation (abstract) in this class, and it need to be implemented in all derived classes.

3-Class Cash: derived from Payment class

Attributes:

```
double tenderdCash // cash provided by customer, e.g. $200
double change// change should be returned to customer, e.g. if payment amount is $196, the customer provided $200, then he should get back $4
```

Methods:

- At least one Constructor: see the main() method to understand what constructor parameters are needed. It should call the base class constructor with the needed parameters
- double **calcChange**(): updates the change attribute, and returns the change. Change = tenderedCash – payment amount. **Note** that amount in base class (Payment) is private (not protected and not public)
- string **getPaymentType**(): overrides the base abstract class method,returns the string "Cash" that represents the payment type of this class. To be used when displaying Payment types, like in **getBillInfo**().
-

4-Class CreditCard : derived from Payment class

Attributes:

```
string CCNum; // credit card number
string expDate; // expiration date
string CHolderName;// card holder name
string CType; //Visa,Mastercard,..
```

Methods:

- At least one Constructor: see the main() method to understand what constructor parameters are needed
- **verify**(): overrides the base class method, it writes to the console the message "Credit card verified" and returns true;
- **getPaymentType**():overrides the base class abstract method. It returns the string "Credit Card" that represents the payment type of this class

5-Class Check: Used to represent with the following

Attributes:

```
string Bank; // bank name
string custName; // customer/client name
string accNum; // account number
```

Methods:

- At least one Constructor: see the main() method to understand what constructor parameters are needed
- **verify**(): overrides the base class method, it writes to the console the message "Signature and account Balance verified" and returns true;
- **getPaymentType**():overrides the base class abstract method. It returns the string "Check" that represents the payment type of this class

*See the **sample output** at the end of this assignment and the **main program** to understand how these classes' methods should work.*

Task B: [For CIDM 5360 students only]

1-Create object diagram for the objects in the main program
2-write a method for Bill class to print a payment plan for the remaining balance based on a given instalment.

void makePaymentPlan(double instalment)

Example 1: if the remaining balance is $70, and instalment/month is $20, then calling this method as `b.makePaymentPlan(20)` should print:

```
Payment Plan for the remaining balance of $70 with instalment/month of $20
1 : on 2/02/2019 pay $20
2 : on 3/02/2019 pay $20
3 : on 4/02/2019 pay $20
4 : on 5/02/2019 pay $10
```

Example 2: if the remaining balance is $70, and instalment/month is $5, then calling this method as `b.makePaymentPlan(5)` should print:

```
Payment Plan for the remaining balance of $70 with instalment/month of $5
1 : on 2/02/2019 pay $5
2 : on 3/02/2019 pay $5
3 : on 4/02/2019 pay $5
4 : on 5/02/2019 pay $5
5 : on 6/02/2019 pay $5
6 : on 7/02/2019 pay $5
7 : on 8/02/2019 pay $5
8 : on 9/02/2019 pay $5
9 : on 10/02/2019 pay $5
10 : on 11/02/2019 pay $5
11 : on 12/02/2019 pay $5
12 : on 1/02/2020 pay $5
13 : on 2/02/2020 pay $5
14 : on 3/02/2020 pay $5
```

(notice the year change in the payment date!)

Grading: See the following rubric

Task	*Possible Points CIDM 4360*	*Possible Points CIDM 5360*
The Class Bill:		
Declaration : Attributes, relationships(if any), accessibility	*5*	*5*
Methods: Constructor	*5*	*4*
getBillInfo()	*5*	*5*
getBalance()	*5*	*5*
addPayment()	*5*	*4*
The Class Payment		
Declaration : Attributes, relationships(if any), accessibility	*5*	*5*
Methods: Constructor	*2*	*2*
getAmount()	*2*	*2*
getPaymentDate()	*2*	*2*
verify()	*2*	*2*
getPaymentType()	*2*	*2*
The Class Cash		
Declaration : Attributes, relationships(if any), accessibility	*5*	*5*
Methods: Constructor	*4*	*3*
calcChange()	*3*	*2*
getPaymentType()	*3*	*2*
The Class CreditCard		
Declaration : Attributes, relationships(if any), accessibility	*5*	*5*
Methods: Constructor	*4*	*3*
verify()	*3*	*2*
getPaymentType()	*3*	*2*
The Class Check		
Declaration : Attributes, relationships(if any), accessibility	*5*	*5*
Methods : Constructor	*4*	*3*
verify()	*3*	*2*
getPaymentType()	*3*	*2*
Execution *Program runs without errors and gives the expected output*	*15*	*10*
Task B: *CIDM 5360 only*		
1- Create object diagram		*8*
2- makePaymentPlan method		8
Total Points	*100*	*100*

Sample Output:

```
Bill Info.:
Date :01/02/2019
Description :Utility Nov 1, 2018 to Dec 31, 2018
Amount Due: $500
Total payed : 0
Remaining balance :500

--------------new payment-------------------------
   Payment verified ...
Bill Info.:
Date :01/02/2019
Description :Utility Nov 1, 2018 to Dec 31, 2018
Amount Due: $500
On 01/15/2019 payment with Cash the amount $230 was processed
Total payed : 230
Remaining balance :270

--------------new payment-------------------------
   Signature and balance verified ...
Bill Info.:
Date :01/02/2019
Description :Utility Nov 1, 2018 to Dec 31, 2018
Amount Due: $500
On 01/15/2019 payment with Cash the amount $230 was processed
On 01/01/2019 payment with Check the amount $100 was processed
Total payed : 330
Remaining balance :170

--------------new payment-------------------------
   Credit card verified ....
Bill Info.:
Date :01/02/2019
Description :Utility Nov 1, 2018 to Dec 31, 2018
Amount Due: $500
On 01/15/2019 payment with Cash the amount $230 was processed
On 01/01/2019 payment with Check the amount $100 was processed
On 01/01/2019 payment with Credit Card the amount $100 was processed
Total payed : 430
Remaining balance :70
```