Part A. The Logic of Branching

A1. (3pt) Write a fragment of code that will test whether an integer variable score contains a valid test score. Valid test scores are in the range 0 to 100.

```
if (score >=0 && score <=100)
    System.out.println("valid");
else
    System.out.println("invalid");
```

A2. (3pt) Write a fragment of code that will change the integer value stored in x as follows. If x is even, divide x by 2. If x is odd, multiply x by 3 and subtract 1.

```
if (x % 2 == 0)
    x = x/2;
else
    x = x*3 - 1;
```

A3. (6pt) Consider the following fragment of code:
a. B b. B c. A d. A e. A f. A

A4. (6pt) Consider the following fragment of code:
a. C b. C
c. A
 B
d. A
 B
e. A
f. A

A5. (4pt) What is the value of each of the following boolean expressions if x is 5, y is 10, and z is 15?
a. false b. true c. true d. true

A6. (2pt) Consider the boolean expression ((x > 10) || (x < 100)).
If x satisfies the first constraint, the computer will not compute the second constraint.
That is: if x>10, the expression is true; if x<10, the expression is still true.
The expression is always true.
So the correct expression may be ((x > 10) && (x < 100)).

A7. (2pt) Consider the boolean expression ((2 < 5) && (x < 100)).
(2 < 5) is always true. Whether the expression is true only depends on whether x < 100.
So the original expression equals to (x < 100).

A8. (4pt)Write a switch statement to convert a letter grade into an equivalent numeric value on a four point scale. Set the value of the variable gradeValue to 4.0 for an A, 3.0 for a B, 2.0 for a C, 1.0 for a D, and 0.0 for an F. For any other letter, set the value to 0.0 and display an error message.

```
double gradeValue;
switch (letterGrade)
{
    case 'A':
        gradeValue = 4.0;
        System.out.println("gradeValue = "+gradeValue);
        break;
    case 'B':
        gradeValue = 3.0;
        System.out.println("gradeValue = "+gradeValue);
        break;
    case 'C':
        gradeValue = 2.0;
        System.out.println("gradeValue = "+gradeValue);
        break;
    case 'D':
        gradeValue = 1.0;
        System.out.println("gradeValue = "+gradeValue);
        break;
    case 'F':
        gradeValue = 0.0;
        System.out.println("gradeValue = "+gradeValue);
        break;
    default:
        gradeValue = 0.0;
        System.out.println("Error: invalid letter grade.");
        break;
}
```

A9. (5pt) Suppose you are writing a program that asks the user to give a yes or no response. Assume that the program reads the user's response into the String variable response.

a. If response is yes or y, set the boolean variable accept to true ; otherwise, set it to false.

```
System.out.println("Give your response.");
Scanner keyboard = new Scanner(System.in);
String response;
response = keyboard.nextLine();
boolean accept;
if (response.equals("yes") || response.equals("y"))
```

```
        accept = true;
    else
        accept = false;
    System.out.println(accept);
```

b. How would you change the code so that it will also accept Yes and Y?
Change this statement: **if** (response.equals("yes") || response.equals("y"))
Into: **if** (response.equalsIgnoreCase("yes") || response.equalsIgnoreCase("y"))

A10. (5pt) Consider Question 8 above, but include + or – letter grades. A+ is 4.25, A- is 3.75, B+ is 3.25, B- is 2.75, etc.

a. Why can't we use one switch statement with no other conditionals to convert these additional letter grades?
Because the controlling expression of switch statement must be of an integer type or of type char.

b. Write a fragment of code that will do the conversion using a multi--- branch if---else statement.

```
    double gradeValue;
// letterGrade is a String variable.
    if (letterGrade.equals("A+"))
        gradeValue = 4.25;
    else if (letterGrade.equals("A"))
        gradeValue = 4.00;
    else if (letterGrade.equals("A-"))
        gradeValue = 3.75;
    else if (letterGrade.equals("B+"))
        gradeValue = 3.25;
    else if (letterGrade.equals("B"))
        gradeValue = 3.00;
    else if (letterGrade.equals("B-"))
        gradeValue = 2.75;
    else if (letterGrade.equals("C+"))
        gradeValue = 2.25;
    else if (letterGrade.equals("C"))
        gradeValue = 2.00;
    else if (letterGrade.equals("C-"))
        gradeValue = 1.75;
    else if (letterGrade.equals("D+"))
        gradeValue = 1.25;
    else if (letterGrade.equals("D"))
        gradeValue = 1.00;
    else if (letterGrade.equals("D-"))
```

```
            gradeValue = 0.75;
        else if (letterGrade.equals("F"))
            gradeValue = 0.00;
        else
            {
                gradeValue = 0.00;
                 System.out.println("Error: invalid letter grade.");
            }
        System.out.println("gradeValue = " + gradeValue);
```

c. Write a fragment of code that will do the conversion using nested switch statements.

```
double gradeValue;
// letterGrade is a String variable.
if (letterGrade.equals("A+")||letterGrade.equals("A")||letterGrade.equals("A-"))
{
    if (letterGrade.equals("A+"))
        gradeValue = 4.25;
    else if (letterGrade.equals("A"))
        gradeValue = 4.00;
    else
        gradeValue = 3.75;
}
else if (letterGrade.equals("B+")||letterGrade.equals("B")||letterGrade.equals("B-"))
{
    if (letterGrade.equals("B+"))
        gradeValue = 3.25;
    else if (letterGrade.equals("B"))
        gradeValue = 3.00;
    else
        gradeValue = 2.75;
}
else if (letterGrade.equals("C+")||letterGrade.equals("C")||letterGrade.equals("C-"))
{
    if (letterGrade.equals("C+"))
        gradeValue = 2.25;
    else if (letterGrade.equals("C"))
        gradeValue = 2.00;
    else
        gradeValue = 1.75;
}
else if (letterGrade.equals("D+")||letterGrade.equals("D")||letterGrade.equals("D-"))
{
    if (letterGrade.equals("D+"))
        gradeValue = 1.25;
    else if (letterGrade.equals("D"))
        gradeValue = 1.00;
    else
        gradeValue = 0.75;
}
else if (letterGrade.equals("F"))
    gradeValue = 0.00;
else
    {
        gradeValue = 0.00;
         System.out.println("Error: invalid letter grade.");
    }
System.out.println("gradeValue = " + gradeValue);
```

Part B. The Looping

B1. (5pt) Write a fragment of code that will read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal to its last character. For the required loop:

a. a while statement

```
System.out.println("Please enter words one by one.");
System.out.println("Enter the word 'done' after you have entered all the words.");
Scanner keyboard = new Scanner(System.in);
String words = keyboard.nextLine();
while (!(words.equals("done")))
{
    int length = words.length();
    char firstCharacter = words.charAt(0);
    char lastCharacter = words.charAt(length-1);
    if (firstCharacter == lastCharacter)
        System.out.println("The word " + words + " satisfies that its first character is
equal to its last character.");
    else
        System.out.println("The word " + words + " does not satisfy that its first
character is equal to its last character.");
    System.out.println("Please continue enter words.");
    words = keyboard.nextLine();
}
System.exit(0);
```

b. do-while statement

```
System.out.println("Please enter words one by one.");
System.out.println("Enter the word 'done' after you have entered all the words.");
Scanner keyboard = new Scanner(System.in);
String words = keyboard.nextLine();
do
{                                        //Assume that the first word is not "done".
    int length = words.length();
    char firstCharacter = words.charAt(0);
    char lastCharacter = words.charAt(length-1);
    if (firstCharacter == lastCharacter)
        System.out.println("The word " + words + " satisfies that its first character is
equal to its last character.");
    else
        System.out.println("The word " + words + " does not satisfy that its first
character is equal to its last character.");
    System.out.println("Please continue enter words.");
    words = keyboard.nextLine();
} while (!(words.equals("done")));
System.exit(0);
```

B2. (5pt) Write a fragment of code that will compute the sum of the first n positive odd integers. For example, if n is 5, you should compute 1 +3 + 5 +7 + 9.

```
int n = keyboard.nextInt();
int i;
 int sum = 0;
 for (i = 1; i <= 2*n-1; i = i + 2)
     sum = sum +i;
 System.out.println("sum = " + sum);
```

B3. (5pt) Convert the following code so that it uses nested while statements instead of for statements:

```
int s = 0;
int t = 1;
int i = 0;
while ( i < 10 )
{
    s = s + i;
    int j = i;
    while ( j > 0)
    {
        t = t * (j - i);
        j--;
    }
    s = s * t;
    System.out.println("T is " + t);
    i++;
}
System.out.println("S is " + s);
```

B4. (5pt) Write a for statement to compute the sum $1 + 2^2 + 3^2 + 4^2 + 5^2 + ... + n^2$.

```
    int n = keyboard.nextInt();
    int sum = 0;
    int t = 1;
    for (int i=1; t<=n; t++)
    {
        i = t*t;
        sum = sum +i;
    }
    System.out.println("sum = " + sum);
```

B5. (5pt) Write a loop that will create a new string that is the reverse of a given string.

```
    String s = keyboard.nextLine();
    int length = s.length();
    String reverse = "";
    for (int i = 0; i < length; i++)
        reverse = s.charAt(i) + reverse;
    System.out.println("the reverse string is " + reverse);
```

B6. (5pt) Develop an algorithm for a simple game of guessing a secret five-digit code. When the user enters a guess at the code, the program returns two values: the number of digits in the guess that are in the correct position and the sum of those digits. For example, if the secret code is 53840, and the user guesses 83241, the digits 3 and 4 are in the correct position. Thus, the program should respond with 2 and 7. Allow the user to guess a fixed number of times.

Algorithm:
1. Set the variable *answer* as a String type to store the secret five-digit code.
2. Read the five-digit code the user enters into the String type variable *guess*.
3. Set the int type variable *index* to record the position that points the digit that the two figures from *answer* and *guess* are being compared.
4. Set the initial *index* pointing to the first digit. Then execute a loop to compare the two figures from *answer* and *guess* by each digit. Let *index* increment by one for each step in the loop until *index* points to the fifth digit. Then the loop ends. **(An inner loop)**
5. During the comparison procedure of each step in the loop:

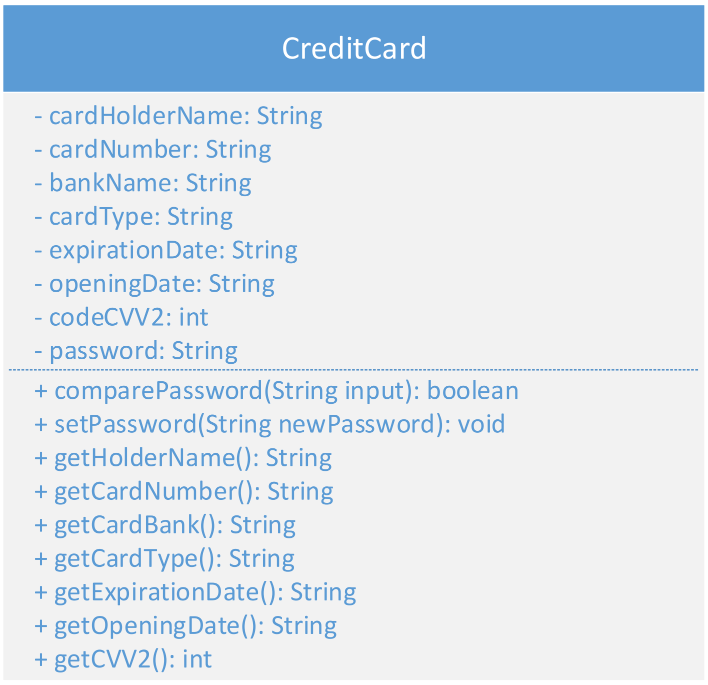
- Set the int type variable *number* (initialized by zero) to record the number of digits in the guess that are in the correct position.

- Set the int type variable *sum* (initialized by zero) to record the sum of the correct figures in all digits.
- Once the correct match appears, set *number*++ and extract the right figure to be stored as the int type variable *figure*, then set *sum* = *sum* + *figure*.
- If the match does not appear in the step, nothing will be executed.

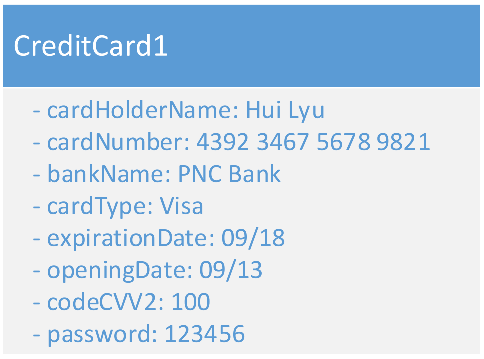
6. When the loop ends, output *number* and *sum* to prompt the user.
7. Repeat Step 2 to Step 6 above for certain times (Allow the user to guess a fixed number of times). **(An outer loop) The whole program has nested loops.**

Part C. Classes and Methods

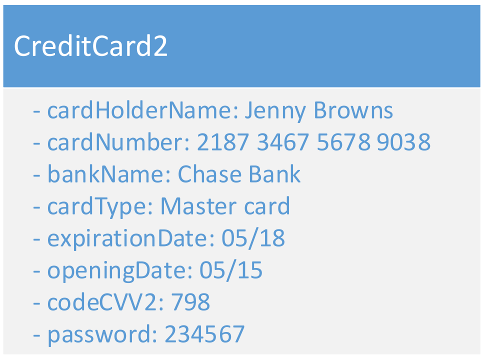
C1. (5pt) Design a class to represent a credit card. Think about the attributes of a credit card; that is, what data is on the card? What behaviours might be reasonable for a credit card? Use the answers to these questions to write a UML class diagram for a credit card class.
Give three examples of instances of this class.



```
CreditCard
- cardHolderName: String
- cardNumber: String
- bankName: String
- cardType: String
- expirationDate: String
- openingDate: String
- codeCVV2: int
- password: String
+ comparePassword(String input): boolean
+ setPassword(String newPassword): void
+ getHolderName(): String
+ getCardNumber(): String
+ getCardBank(): String
+ getCardType(): String
+ getExpirationDate(): String
+ getOpeningDate(): String
+ getCVV2(): int
```



```
CreditCard1
- cardHolderName: Hui Lyu
- cardNumber: 4392 3467 5678 9821
- bankName: PNC Bank
- cardType: Visa
- expirationDate: 09/18
- openingDate: 09/13
- codeCVV2: 100
- password: 123456
```



```
CreditCard2
- cardHolderName: Jenny Browns
- cardNumber: 2187 3467 5678 9038
- bankName: Chase Bank
- cardType: Master card
- expirationDate: 05/18
- openingDate: 05/15
- codeCVV2: 798
- password: 234567
```

CreditCard3

- cardHolderName: Emily Blake
- cardNumber: 6782 3210 5678 9873
- bankName: Chase Bank
- cardType: Visa
- expirationDate: 06/17
- openingDate: 06/14
- codeCVV2: 989
- password: 345678

C2. (5pt) Repeat C1 for a credit card account instead of a credit card. An account represents the charges and payments made using a credit card.

CreditCardAccount

- cardHolderName: String
- cardNumber: String
- password: String
- chargeDate: String
- paymentDate: String
- chargeAmount: double
- chargePerson: String
- paymentAmount: double
- purchaseItem: String

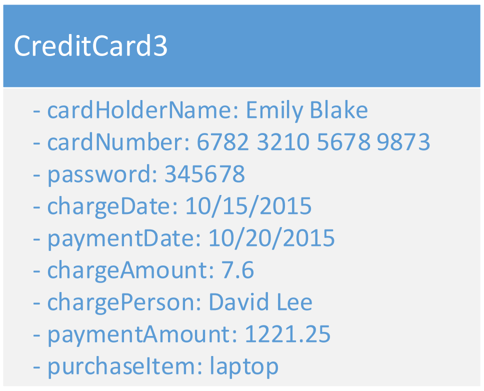
+ comparePassword(String input): boolean
+ setPassword(String newPassword): void
+ getHolderName(): String
+ getCardNumber(): String
+ getChargeDate(): String
+ setPaymentDate(String newDate): void
+ getChargeAmount(): double
+ getChargePerson(): String
+ setPaymentAmount(double newPayment): void
+ setPurchaseItem(String newPurchase): void
+ getTotalAmount(): double

CreditCard1

- cardHolderName: Hui Lyu
- cardNumber: 4392 3467 5678 9821
- password: 123456
- chargeDate: 10/17/2015
- paymentDate: 10/18/2015
- chargeAmount: 45.0
- chargePerson: Jenny Knox
- paymentAmount: 35.8
- purchaseItem: CD

CreditCard2

- cardHolderName: Jenny Browns
- cardNumber: 2187 3467 5678 9038
- password: 234567
- chargeDate: 09/30/2015
- paymentDate: 10/06/2015
- chargeAmount: 30.5
- chargePerson: Jake Dehp
- paymentAmount: 29.9
- purchaseItem: rice cooker



CreditCard3

- cardHolderName: Emily Blake
- cardNumber: 6782 3210 5678 9873
- password: 345678
- chargeDate: 10/15/2015
- paymentDate: 10/20/2015
- chargeAmount: 7.6
- chargePerson: David Lee
- paymentAmount: 1221.25
- purchaseItem: laptop