Computer Programming Assignment 2

Checkpoints

1. You should do the assignment in your own. You are not allowed to share code with others and/or copy code from other resources. If you are caught, as in the syllabus, you will get a failing grade.
2. Grading will be done in Linux environment using java 1.8.
3. Program failed to compile/run will result 0.
4. Do not loop your program to repeat unless you are told so.
5. Do not change input/output format unless you are told so.
6. Do not color console.
7. Write your name and student number at top of program as a comment
8. DO NOT include Korean (and any other language than English) comment
    - Korean comment will cause compilation errors in Linux environment, which will result 0 for your grade

Submission

1. Submit your assignment on eTL.
2. Zip your file (or tar) as '<Student ID>-assign2.zip'
3. Due of this assignment is Oct 27(Fri) .
4. No late submission is allowed

Problem 1 Shift Cipher

Write a java program Assignment2_1.java that takes a line of text, a char (e or d) indicating operation, and an int key value.

Input text will consist of {a~z}, {A~Z}, and {1~9}. If input text contains a character not in bound, print error message.

Key value will be between -100~100

Depending on operation, program produces following result:
Encryption(e):
Shift an input text by given key value. ex.) key = 1: a→b, b→c … z→A, A→B … Z→1 … 9→a

Decryption(d):
Shift an input text by given key value. ex.) key = 1: a→9 ... 1→Z … A→z ... b→a

```
Enter string : abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789
Enter operation : e
Enter key value : 10
klmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789abcdefghij
```

```
Enter string : klmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789abcdefghij
Enter operation : d
Enter key value : 10
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789
```

```
Enter string : 00001
Enter operation : e
Enter key value : 1
Error, input value 0 is out of range
```

Problem 2

Write a java program to simulate the following simple four-player card game.

- Represent each card as a string of two characters consisting of a suite (one of H, D, C or S) and a value (one of A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q or K).

- First, deal (distribute) 10 cards to each player from a deck of 52 cards. To simulate random distribution, create a string consisting of all 52 cards in any fixed order. Then repeatedly generate a random number (within the range of the remaining number of cards), and deal that card to the next player. Display the result of this step by printing each player's `hand' (the cards that the player received), as a string.

- Distributed cards must not have duplicate cards since they are drawn from a deck of 52 cards.

- In the next step, each player (in a round robin fashion) plays a card from his hand. Whenever the *suit* of a player's card matches *suit* of the card played by the previous player, the latter player gets 4 points; and whenever the *value* of a player's card matches *value* of the card played by the previous player, the latter player gets 13 points. The game ends when all the players play their 10 cards.

- Drawing a card from a player does not need to be random.

- Display the sequence of cards played by the players (in 10 lines, with 4 `cards' per line)

- Display the scores of the 4 players at the end of the game.

[output]

Player 1 : [P1_card1...10]
Player 2 : [P2_card1...10]
Player 3 : [P3_card1...10]
Player 4 : [P4_card1...10]
[empty_line]
[P1_card1] [P2_card1] [P3_card1] [P4_card1]
[P1_cardk] [P2_cardk] [P3_cardk] [P4_cardk]
[...]
[P1_card10] [P2_card10] [P3_card10] [P4_card10]
Player 1 score : [P1 score]
Player 2 score : [P2 score]
Player 3 score : [P3 score]
Player 4 score : [P4 score]