CSc 110 Assignment 3:

Nested Loops, parameters and return values and Scanner

How to hand it in:

Submit files **RocketLaunch.java** through the Assignment 3 link on the CSC110 conneX site.

Learning Outcomes:

When you have completed this assignment, you should understand:

- How to use a nested for loop to repeat a statement a specified number of times.
- The effect of escape sequences on printed strings.
- More practice with static methods that take parameters and return a value.
- Using Scanner to get input from the command line
- The flow of control (i.e. the effects of method calls and assignment statements, parameters passed in and return values).

Part 1 – Rocket Launch

Write a set of methods that will output an ascii image as shown above. Your implementation must include the following static method:

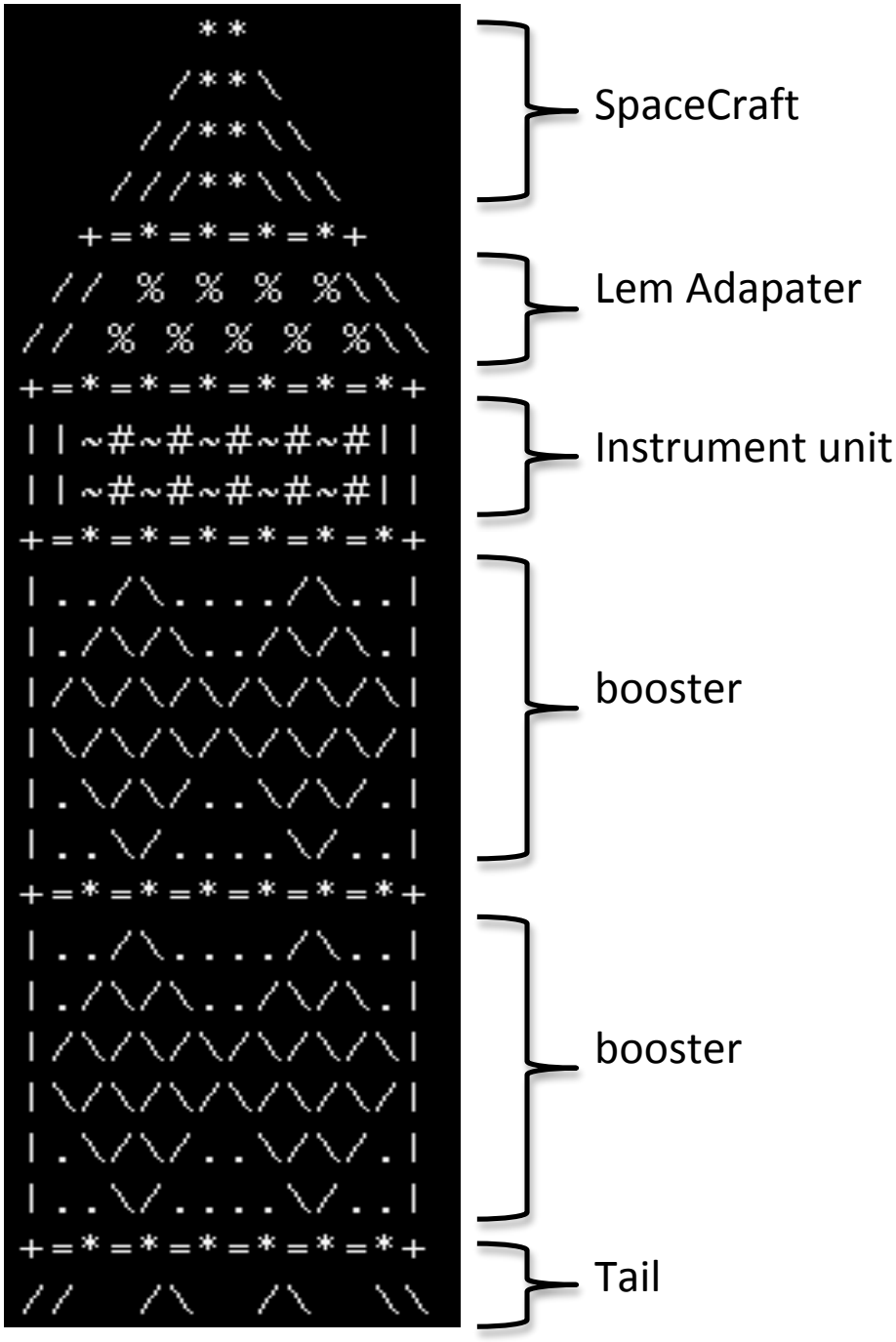
```
/*
  PURPOSE:  to print a ascii rocket ship
  INPUT:    int size, controls the proportions of the rocket
            int numBoosters, controls how many boosters are drawn
  OUTPUT:   an ascii representation of Apollo Spaceship
*/
public static void drawRocket(int size, int numBoosters)
```

To receive full marks you must decompose your implementation into multiple methods. That is, your drawRocket method should call other methods to draw the pieces of the rocket. We have not specified what these methods to give you some control over the design of your program.

In your main method you must call your drawRocket method with values collected from the console. For example, here the program is prompting for the size of the rocket and then for the number of boosters. The image on the following page shows the output given this input and the method call: drawRocket(2, 2).

```
Celinas-MacBook-Air:A3 celinag$ java RocketLaunch
What size rocket would you like to build?
2
How many boosters?
2
```

The following is a screenshot of a sample run of drawRocket with a scale size of 2 and 2 boosters. drawRocket(2, 2). The pieces of the spaceship are labeled to give you ideas for deciding what your methods should be. Two more sample runs are provided on the following page.



```
Celinas-MacBook-Air:A3 celinag$ java RocketLaunch
What size rocket would you like to build?
2
How many boosters?
3
      **
     /**\
    //**\\
   ///**\\\
  +=*=*=*=*+
 // % % % %\\
// % % % % %\\
+=*=*=*=*=*=*+
||~#~#~#~#~#~#||
||~#~#~#~#~#~#||
+=*=*=*=*=*=*+
|../\..../\..|
|./\/\../\/\.|
|/\/\/\/\/\/\|
|\/\/\/\/\/\/|
|.\/\/..\/\/.|
|..\/....\/..|
+=*=*=*=*=*=*+
|../\..../\..|
|./\/\../\/\.|
|/\/\/\/\/\/\|
|\/\/\/\/\/\/|
|.\/\/..\/\/.|
|..\/....\/..|
+=*=*=*=*=*=*+
|../\..../\..|
|./\/\../\/\.|
|/\/\/\/\/\/\|
|\/\/\/\/\/\/|
|.\/\/..\/\/.|
|..\/....\/..|
+=*=*=*=*=*=*+
//  /\  /\  \\
```

```
Celinas-MacBook-Air:A3 celinag$ java RocketLaunch
What size rocket would you like to build?
3
How many boosters?
1
        **
       /**\
      //**\\
     ///**\\\
    ////**\\\\
   /////**\\\\\
  +=*=*=*=*=*=*+
 // % % % % % %\\
// % % % % % % %\\
+=*=*=*=*=*=*=*=*+
||~#~#~#~#~#~#~#||
||~#~#~#~#~#~#~#||
+=*=*=*=*=*=*=*=*+
|.../\......./\...|
|../\/\..../\/\..|
|./\/\/\../\/\/\.|
|/\/\/\/\/\/\/\/\|
|\/\/\/\/\/\/\/\/|
|.\/\/\/..\/\/\/.|
|..\/\/....\/\/..|
|...\/......\/...|
+=*=*=*=*=*=*=*=*+
//  /\  /\  /\  \\
```

Part II – Encryption/Decryption

With the release of the movie The Imitation Game, a story Alan Turing's Life, it seems fitting to think about code breaking. AND every rocket launcher should be able to encrypt/decrypt messages ☺

You are required to write an encryption and a decryption method as described below.

```
/*
  PURPOSE:  encrypts msg given an encryption key between 1 and 95
  INPUT:    String msg, the message to be encrypted
            int key, the amount to increase the ascii character by
                  to encrypt the message
  OUTPUT:   an encrypted version of msg returned as a String
*/
public static String encrypt(String msg, int key)

/*
  PURPOSE:  decrypts msg given an encryption key between 1 and 95
  INPUT:    String msg, the message to be decrypted
            int key, the amount to shift the ascii character by
                  to decrypt the message
  OUTPUT:   a decrypted version of msg returned as a String
*/
public static String decrypt(String msg, int key)
```

A very basic encryption algorithm is:

```
message String comes in as a parameter
Create a new empty String
For each character in the message String
      convert each character to its integer value,
      add the encryption key value to the integer value
      convert the new integer back to a character
      append the new character to the new String
//end of for loop
Return the new String
```

A similar, algorithm will work for decryption

```
message String comes in as a parameter
Create a new empty String
For each character in the encrypted message String
      convert each character to its integer value,
      subtract the encryption key value to the integer value
      convert the new integer back to a character
      append the new character to the new String
Return the new String
```

The following ASCII table will help you with the conversion of a char to its decimal value.
https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html
Your methods must support the conversion of the characters with integer values 32-126 (SPACE character to ~ character).

RECALL: You can cast a character to an integer to get its integer value.
Use the table link above to verify the comments in the following example…

```
char c = 'a';    // c is character a here
int x = (int) c;  // x is 97 here
x = x + 10;      // I add 10 to x so x is now 107
char newC = (char)x;  //newC is the char equivalent of x(107), k
```

NOTICE: the algorithm above does not take into account when the conversion from a char to an int and the addition of the key results in an integer value out of range.
Ie. The '~' character has an integer value of 126 if I add an encryption key of 5; we get an integer value of 131 which does not have a character value. Your method implementation should account for this.

One suggested solution is provided here for you. There are other ways to do this (use of the mod (%) operator and you may implement it however you chose. This following is just a suggestion.

Given our original algorithm…

```
message String comes in as a parameter
Create a new empty String
For each character in the message String
    convert each character to it's integer value,
    add the encryption key value to the integer value
    convert the new integer back to a character
    append the new character to the new String
//end of for loop
Return the new String
```

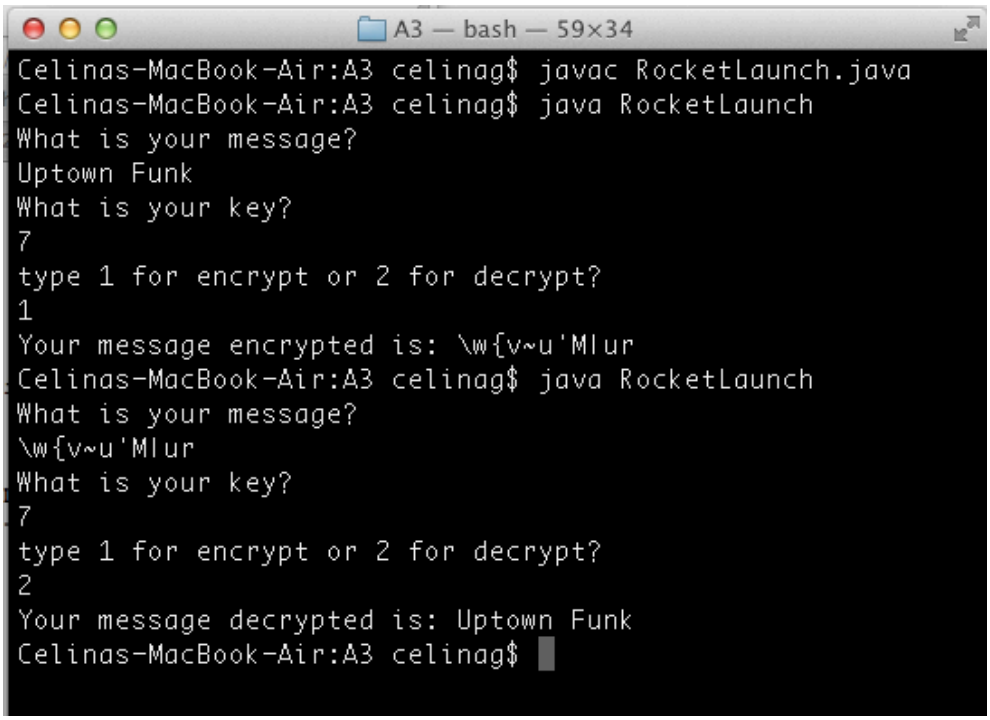
What if this new value is > our highest character value of 126?
You will want to account for this case.
One way to do it is using an if statement that checks for this condition and adjusts accordingly…

```
    if new integer value is >126
        get integer to increment correct amount from 32 (first character)
```

An example…
Converting character 'z' which has an integer value of 122
The key is specified to be 9
New integer value is 122+9 = 131
131 is > 126
So we must wrap around to the beginning
And increment from 32 the correct number of times (4 in this case)
New integer value is 32 + 4 = 36
New character is '$'

FINALLY: In your main method you must ask your user if they want to decrypt or encrypt. You will then prompt your user for a message and a key to pass as parameters to the correct method. This sample run on the following page shows an encryption of "Uptown Funk" with a key of 5 followed by the decryption of this String.

CHALLENGE: Get your implementation to work with a key > 95. Hint: the mod operator (%) might be useful.



Grading:

Marks will be allocated for the following…

- Your code must compile and run.
- Your code should produce the output exactly as requested.
- Your code must be indented and documented appropriately. Please follow the guidelines in `Style_Guidelines.pdf`, available in the Resources folder on conneX.