01. Caesar Cipher Submissions

Instructions

- Comment on code by highlighting it with your mouse, right clicking the highlighted text, then selecting "Comment"
- If your feedback is to rename a variable or another very small change, highlight the code you want to change and type some stuff
- Give constructive feedback, as if a coworker wanted to submit this code and asked you for help
- Learn from others' code and comment if you find anything new, helpful, or interesting
    - If you see something you like but don't want to write about it, just comment "+1", "nice", or whatever you want
-

Instructions

1 CaesarCipher.java

2 CaesarCipher.java

3 CaesarCipher.java

4 CaesarCipher.java

5 CaesarCipher.java

6 CaesarCipher.java

7 CaesarCipher.java

8 CaesarCipher.java

9 CaesarCipher.java

10 CaesarCipher.java

11 CaesarCipher.java

12 CaesarCipher.java

13 CaesarCipher.java

14 CaesarCipher.java

15 CaesarCipher.java

16 CaesarCipher.java

17 CaesarCipher.java

18 CaesarCipher.java

19 CaesarCipher.java

20 CaesarCipher.java

21 CaesarCipher.java

22 CaesarCipher.java

23 CaesarCipher.java

24 CaesarCipher.java

25 CaesarCipher.java

26 CaesarCipher.java

27 CaesarCipher.java

28 CaesarCipher.java

29

CaesarCipher.java 30
CaesarCipher.java 31
CaesarCipher.java 32
CaesarCipher.java 33
CaesarCipher.java

1

CaesarCipher.java

```
package hw01;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Scanner;

public class CaesarCipher
{
    public static String caesarCipher(String message, int shiftAmount)
{
    String encodedMessage = "";
    int[] lowerCase = {97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122};
    int[] upperCase = {65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90};
    double numLoop;
    double numWithdecimal;
    BigDecimal whichLetter;
    final BigDecimal NUM_OF_ALPHABET = new BigDecimal(26.0);
    int intNumtoShift;
    int num, num2;
    int initialTest;

    for (char letter : message.toCharArray())
    {
        boolean isNuminLowerRange = false;
    boolean isNuminUpperRange = false;

        if (letter >= 97 && letter <= 122)//dealing with lowercase a through z
            {
                //this code deals with converting the shiftAmount to a number(-26 to 26) that tells us how many letters to shift
                numLoop = shiftAmount/26.0;//tells us how many times it shifts throught he alphabet, but we are just interested in the remainder
                numWithdecimal = numLoop - (long)numLoop;//getting just the part after the decimal (or remainder)
                BigDecimal decimal = new BigDecimal(numWithdecimal);//gives us a more accurate decimal for safety
                whichLetter = NUM_OF_ALPHABET.multiply(decimal); //gets the value of the appropriate shiftAmount but with multiple digits
                BigDecimal numToShift = whichLetter.setScale(0, RoundingMode.HALF_UP);//round whichLetter back to integer to closest integer
                intNumtoShift = numToShift.intValue(); //gives us the final shiftAmount in int form to work with

                //trying to see if just adding the shift will be in the range of lowercase values, if so then we can go ahead and print this letter, if not we need some added calculations
                initialTest = letter + intNumtoShift;

                for(int i: lowerCase)
                {
                    if(i == initialTest)
                    {
                        isNuminLowerRange = true;
                        break;
                    }
                }

                if(intNumtoShift < 0)//handling the case where the shift amount is negative
                {
                    if(isNuminLowerRange)//here we just print what adding the shift will give us since it is in the range of values, otherwise we need an else to "loop" around

                    {
                        num2 = initialTest;
                    }
                    else
                    {
                        num = 26 + intNumtoShift;
```

```
                num2 = letter + num;
                }

                encodedMessage += (char)num2; //concatenating the resulting char letter to the final string to print
        }
        if(intNumtoShift >= 0)//handling the case when shift amount is greater than zero or equal to zero
        {


                if(isNuminLowerRange)
                {

                        num2 = initialTest;
                }
                else
                {
                        num = 26 - intNumtoShift;
                        num2 = letter - num;
                }
                encodedMessage += (char)num2;
        }
}

        else if (letter >= 65 && letter <= 90)//dealing with UPPERCASE A through Z, all comments from above apply to uppercase since it is doing the same
thing, just to different ASCI values
        {
                //this code deals with converting the shiftAmount to a number(-26 to 26) that tells us how many letters to shift
                        numLoop = shiftAmount/26.0;//tells us how many times it shifts throught he alphabet, but we are just interested in the remainder
                        numWithdecimal = numLoop - (long)numLoop;//getting just the part after the decimal (or remainder)
                        BigDecimal decimal = new BigDecimal(numWithdecimal);//gives us a more accurate decimal for safety
                        whichLetter = NUM_OF_ALPHABET.multiply(decimal); //gets the value of the appropriate shiftAmount but with multiple digits
                        BigDecimal numToShift = whichLetter.setScale(0, RoundingMode.HALF_UP);//round whichLetter back to integer to closest
integer
                        intNumtoShift = numToShift.intValue(); //gives us the final shiftAmount in int form to work with

                        initialTest = letter + intNumtoShift;

                for(int i: upperCase)
                {
                        if(i == initialTest)
                        {
                                isNuminUpperRange = true;
                                break;
                        }
                }

                        if(intNumtoShift < 0)
                {
                                if(isNuminUpperRange)
                                {
                                        num2 = initialTest;
                                }
                                else
                                {
                                        num = 26 + intNumtoShift;
                                num2 = letter + num;
                                }

                        encodedMessage += (char)num2;
                }
                if(intNumtoShift >= 0)
                {


                        if(isNuminUpperRange)
                        {
                                num2 = initialTest;
                        }
                        else
                        {
                                num = 26 - intNumtoShift;
                                num2 = letter - num;
                        }

                        encodedMessage += (char)num2;
                }
```

```
                }

                else //if the character in the message is not a lowercase letter a-z or UPPERCASE letter A-Z
                {
                        encodedMessage += letter;
                }
        }


        System.out.println(encodedMessage);
        return encodedMessage;
}

public static void main(String args[])
{

        int howMany = args.length;
        String theMessage = "";

        int one = Integer.valueOf(args[0]);

        for(int i = 1; i< howMany && i >=1; i++)
        {
                theMessage = args[i];
                caesarCipher(theMessage, one);
}



//String to int
/*
String userMessage;
String shiftAmount = "";
int shiftAmount2 = 0;

System.out.print("Enter shift amount and message, for example '2 abzy': " ); //using Scanner object to get an input to encode
Scanner input = new Scanner(System.in);
userMessage = input.nextLine();




if(!userMessage.isEmpty())//dealing with the situation if someone does not print anything
{
        if(userMessage.substring(0, 1).equals("-") && userMessage.substring(2, 3).equals(" ")) // dealing with negative numbers of one digit
        {
                shiftAmount = userMessage.substring(1, 2);
                shiftAmount2 = (-1) * Integer.parseInt(shiftAmount);
                userMessage = userMessage.substring(3);
        }
        else if(userMessage.substring(0, 1).equals("-") && userMessage.substring(3, 4).equals(" ")) // dealing with negative numbers of two digits
        {
                shiftAmount = userMessage.substring(1, 3);
                shiftAmount2 = (-1) * Integer.parseInt(shiftAmount);
                userMessage = userMessage.substring(4);
        }
        else if(userMessage.substring(0, 1).equals("-") && userMessage.substring(4, 5).equals(" ")) // dealing with negative numbers of three digits
        {
                shiftAmount = userMessage.substring(1, 4);
                shiftAmount2 = (-1) * Integer.parseInt(shiftAmount);
                userMessage = userMessage.substring(5);
        }
        else if(userMessage.substring(1, 2).equals(" ")) // dealing with one digit positive numbers
        {
                shiftAmount = userMessage.substring(0, 1);
                shiftAmount2 = Integer.parseInt(shiftAmount);
                userMessage = userMessage.substring(2);
        }
        else if(userMessage.substring(2,3).equals(" ")) // dealing with two digit positive numbers
        {
                shiftAmount = userMessage.substring(0, 2);
                shiftAmount2 = Integer.parseInt(shiftAmount);
                userMessage = userMessage.substring(3);
        }
        else if(userMessage.substring(3,4).equals(" ")) // dealing with three digit positive numbers
```

```
                {
                        shiftAmount = userMessage.substring(0, 3);
                        shiftAmount2 = Integer.parseInt(shiftAmount);
                        userMessage = userMessage.substring(4);
                }
        }
        else System.out.println("Invalid"); //if nothing is printed


        caesarCipher(userMessage, shiftAmount2); //run caesarCipher to the message
        */
    }

}
```

2

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

    public static String CaesarCipher(String message, int shiftAmount) {
String encodedArg = "";

int newAsciiVal;
int finalAsciiVal = 0;

int newShiftAmount;
int finalShiftAmount = 0;


newShiftAmount = shiftAmount % 26;

for (char asciiVal : message.toCharArray()) {
        finalAsciiVal = newShiftAmount + asciiVal;
        if (asciiVal >= 97 && asciiVal <= 122) {
                if (newShiftAmount > 0) {
                        if (finalAsciiVal > 122) {
                                finalShiftAmount = 26 - newShiftAmount;                         // ???
                finalAsciiVal = asciiVal - finalShiftAmount;
                encodedArg += (char) finalAsciiVal;
                        }
                        else {
                                finalAsciiVal = asciiVal - finalShiftAmount;
                encodedArg += (char) finalAsciiVal;
                        }

        }
        else if (newShiftAmount < 0) {
                if (finalAsciiVal < 97) {
                        finalShiftAmount = 26 + newShiftAmount;                         // ???
                finalAsciiVal = asciiVal - finalShiftAmount;
                encodedArg += (char) finalAsciiVal;
                }
                else {
                        finalAsciiVal = asciiVal - finalShiftAmount;
                encodedArg += (char) finalAsciiVal;
                }
        }
}

else if (asciiVal >= 65 && asciiVal <= 90) {
        if (newShiftAmount > 0) {
                finalShiftAmount = 26 - newShiftAmount;
        finalAsciiVal = asciiVal - finalShiftAmount;
        encodedArg += (char) finalAsciiVal;
        }
        else if (newShiftAmount < 0) {
                finalShiftAmount = 26 + newShiftAmount;
                finalAsciiVal = asciiVal - finalShiftAmount;
                encodedArg += (char) finalAsciiVal;
        }
}
```

```
                else {
                        encodedArg += (char) finalAsciiVal;
                }
        }
        return message;
    }



    public static void main(String args[]) {

        String stringMessage = "";

        int shiftAmount = Integer.valueOf(args[0]);

        for (int i = 1; i < args.length && i >= 1 ; i++) {
                stringMessage = args[i];
                System.out.println(CaesarCipher(stringMessage, shiftAmount));
        }

    }
}
```

3

CaesarCipher.java

```
package hw01;

public class CaesarCipher {
    // Return a String that is the same as message but where all the alphabetic
    // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
    // All non-alphabetic characters should remain unchanged. For example,
    // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
    // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
    public static String caesarCipher(String message, int shiftAmount) {
        // for each character, c in message:
        //   if c >= 'a' and <= 'z'
        //     shift c by shiftAmount
        //   else if c >= 'A' and <= 'Z'
        //     shift c by shiftAmount
        // add c to output
        char[] New_message = message.toCharArray();
        for (int i = 0; i < message.length(); i++){
            if(New_message[i] >= 'a' && New_message[i] <= 'z'){
                int Convert_message = (int)New_message[i] + shiftAmount;
                while (Convert_message > 'z'){
                    Convert_message -= 26;
                }
                while(Convert_message < 'a'){
                    Convert_message += 26;
                }
                New_message[i] = (char) Convert_message;
            }
            else if(New_message[i] >= 'A' && New_message[i] <= 'Z') {
                int Convert_message = (int)New_message[i] + shiftAmount;
                while (Convert_message > 'Z'){
                    Convert_message -= 26;
                }
                while (Convert_message < 'A' ){
                    Convert_message += 26;
                }
                New_message[i] = (char) Convert_message;
            }
        }
        return String.valueOf(New_message);
    }

    // Print the cipher of args[1], args[2], etc. on their own lines
    // shifted by args[0].
    public static void main(String args[]) {
        // Read shiftAmount from args[0]
        // for each arg in args[1], args[2], etc.:
        //   print caesarCipher(...)
        Integer shiftAmount = Integer.valueOf(args[0]);
        String message = "";
```

```
        for (int i = 1; i < args.length; i++){
            message = message + args[i] + "\n";
        }
        caesarCipher(message,shiftAmount);
        System.out.print(caesarCipher(message,shiftAmount));

    }
}
```

4

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

    // Return a String that is the same as message but where all the alphabetic
    // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
    // All non-alphabetic characters should remain unchanged. For example,
    // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
    // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
    public static String caesarCipher(String message, int shiftAmount) {
        // for each character, c in message:
        // add c to output
//        if (shiftAmount > 26) {
//            shiftAmount = (shiftAmount % 26);
//        }
//        else if (shiftAmount < 0) {
//            shiftAmount = (shiftAmount % 26) + 26;
//        }

        // Convert negative shift value to a positive number
        shiftAmount = ((shiftAmount % 26) + 26) % 26;
        String encodedMessage = "";

        for (int i = 0; i <message.length(); i++) {
            // if c >= 'a' and <= 'z' OR if c >= 'A' and <= 'Z'
            //   shift c by shiftAmount
            if ((message.charAt(i) >= 'a' && message.charAt(i) <= 'z')
                    || (message.charAt(i) >= 'A' && message.charAt(i) <= 'Z')) {
                // if c + shiftAmount > 'z' and c <= 'z' OR if c + shiftAmount > 'Z' and c <= 'Z'
                //   shift c by shiftAmount - 26 to wrap back
                if (((char)(message.charAt(i) + shiftAmount) > 'z' && message.charAt(i) <= 'z')
                        || ((char)(message.charAt(i) + shiftAmount) > 'Z' && message.charAt(i) <= 'Z')) {
                    encodedMessage = encodedMessage + (char)(message.charAt(i) + shiftAmount - 26);
                }
                else {
                    encodedMessage = encodedMessage + (char)(message.charAt(i) + shiftAmount);
                }
            }
            else {
                encodedMessage = encodedMessage + message.charAt(i);
            }
        }
        return encodedMessage;
    }

    // Print the cipher of args[1], args[2], etc. on their own lines
    // Shifted by args[0]
    public static void main(String args[]) {
        // Read shiftAmount from args[0]
        int shift = 0;
        try {
            shift = Integer.valueOf(args[0]);
        }
        catch (NumberFormatException ex) {
            System.out.println("Enter integer value as first argument");
            System.exit(0);
        }

        // For each arg in args[1], args[2], etc print caesarCipher(...)
        for (int i = 1; i < args.length ; i++) {
            System.out.println(caesarCipher(args[i],shift));
        }
```

```
    }
}
```

5

CaesarCipher.java

```
package hw01;


public class CaesarCipher {

    // Return a String that is the same as message but where all the alphabetic
    // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
    // All non-alphabetic characters should remain unchanged. For example,
    // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
    // caesarCipher("cd za!\n", -2).equals("ab xy!\n")

    public static String caesarCipher(String message, int shiftAmount) {
        // for each character, c in message:
        //   if c >= 'a' and <= 'z'
        //     shift c by shiftAmount
        //   else if c >= 'A' and <= 'Z'
        //     shift c by shiftAmount
        // add c to output
String encryptedString;
encryptedString = "";

shiftAmount = (shiftAmount % 26);

for(int i = 0; i < message.length(); i++)
{

        char encryptedChar;

        if ('a' <= message.charAt(i) && message.charAt(i) <= 'z')
        {

                if(shiftAmount < 0)
                {

                        encryptedChar = (char) (((message.charAt(i) - 'a' + shiftAmount + 26) % 26) + 'a');
                encryptedString =  encryptedString + encryptedChar;
                }

                else
                {
                        encryptedChar = (char) (((message.charAt(i) - 'a' + shiftAmount) % 26) + 'a');
                encryptedString =  encryptedString + encryptedChar;
                }
        }

        else if ('A' <= message.charAt(i) && message.charAt(i) <= 'Z')
        {

                if(shiftAmount < 0)
                {
                        encryptedChar = (char) (((message.charAt(i) - 'A' + shiftAmount + 26) % 26) + 'A');
                encryptedString =  encryptedString + encryptedChar;
                }

                else
                {
                        encryptedChar = (char) (((message.charAt(i) - 'A' + shiftAmount) % 26) + 'A');
                encryptedString =  encryptedString + encryptedChar;
                }

        }

        else
        {
                encryptedString = (encryptedString + message.charAt(i));
        }
}
```

```
        System.out.println(encryptedString);

        return encryptedString;

}



        // Print the cipher of args[1], args[2], etc. on their own lines
// shifted by args[0].
public static void main(String args[])
{
    // Read shiftAmount from args[0]
    // for each arg in args[1], args[2], etc.:
    //   print caesarCipher(...)


        for(int i = 1; i < args.length; i++)
        {
                caesarCipher(args[i], Integer.parseInt(args[0]));
        }


    }
}
```

6

CaesarCipher.java

```
package hw01;

public class CaesarCipher
{

        // Return a String that is the same as message but where all the alphabetic
        // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
        // All non-alphabetic characters should remain unchanged. For example,
        // caesarCipher("abxy!",  2).equals("cdza!")
        // caesarCipher("cdza!", -2).equals("abxy!")
        public static String caesarCipher(String message, int shiftAmount)
        {

                // for each character, c in message:
                //   if c >= 'a' and <= 'z'
                //     shift c by shiftAmount
                //   else if c >= 'A' and <= 'Z'
                //     shift c by shiftAmount
                // add c to output
                // conversion from string to char
                char[] messageCharArray = message.toCharArray();

                shiftAmount = shiftAmount % 26;


                for ( int i = 0; i < messageCharArray.length ; i++)
                        {
                                char messageChar = messageCharArray[i];

                                if (Character.isLetter(messageChar)
)
                                {
//                                      messageChar += shiftAmount;


                                        if ( messageChar >= 'a' && messageChar <='z')
                                         {
                                                messageCharArray[i] += shiftAmount;
                                                if ( messageCharArray[i] < 'a' )
                                                 {
                                                        messageCharArray[i] += 26;
                                                }
                                                else if (messageCharArray[i] > 'z')
```

```
                        {
                                messageCharArray[i] -= 26;
                        }
                }
                else if ( messageChar >= 'A' && messageChar <= 'Z')
                {
                        messageCharArray[i] += shiftAmount;
                        if ( messageCharArray[i] < 'A' )
                        {
                                messageCharArray[i] += 26;
                        }
                        else if (messageCharArray[i] > 'Z')
                        {
                                messageCharArray[i] -= 26;
                        }
                }

        }


    }

    return new String(messageCharArray)        ;
}
    // Print the cipher of args[1], args[2], etc. on their own lines
    // shifted by args[0].
    public static void main(String args[])
    {
            // Read shiftAmount from args[0]
            // for each arg in args[1], args[2], etc.:
            //   print caesarCipher(...)
//          if(args.length < 2)
//          {
//                  return;
//          }
            int shiftAmount=0;
            try
            {
                    shiftAmount = Integer.valueOf(args[0]);
            }
            catch (Exception e)
            {
                    return;
            }
            for(int i = 1 ; i < args.length ; i++)
            {
                    System.out.println(caesarCipher(args[i],shiftAmount));
            }
    }
}
```

7

CaesarCipher.java

```
package hw01;

/**
 *
 * @author Peter
 */
public class CaesarCipher {
    /**
     * It creates a Caesar Cipher String
     * @param message
     * @param shiftAmount
     * @return
     */
    public static String caesarCipher(String message, int shiftAmount) {
        shiftAmount = shiftAmount % 26;
        int length = message.length();                      //Get the length of the message
        char currentChar;                                   //We are going to store character by character in this variable
        String newMessage="";                               //Declare and initialize the new string that we are going to produce after cypher
        for (int i = 0; i < length; i++) {                  //loop character by chararter until length of the word
            currentChar = message.charAt(i);                    //get the current character at position
```

```
        if (currentChar >= 'a' && currentChar <= 'z') {            //if the current char is between a and z, it is lowercase letter
            currentChar = (char) (currentChar + shiftAmount);       //just add the shift and cast it to be sure it is a character
            if (currentChar > 'z') {                               //if it is greater than z
                currentChar = (char) (currentChar - 'z' + 'a' - 1); //we are going to rest the value of 'z' and sum that to 'a' and then rest 1 to consider also 'a', if not z will
turn into b shifting 1
            }
            if (currentChar < 'a') {                               //if it is less than a
                currentChar = (char) (currentChar - 'a' + 'z' + 1); //we are going to rest the value of 'a' and sum that to 'z' and then sum 1
            }
            newMessage += currentChar;                             //we are going to concatenate the new character to the word we are creating
        } else if (currentChar >= 'A' && currentChar <= 'Z') {     //it means if it is uppercase letter
            currentChar = (char) (currentChar + shiftAmount);       //current character plus shift is going to be the new letter

            if (currentChar > 'Z') {                               //if the new letter is greater than 'Z' (65 in ascii)
                currentChar = (char) (currentChar - 'Z' + 'A' - 1); //We are going to rets that 65 (or 'Z') and sum to A, and also rest 1 to include the 'A'
            }
            if (currentChar < 'A') {                               //if it is less than a
                currentChar = (char) (currentChar - 'A' + 'Z' + 1); //we are going to rest the value of 'a' and sum that to 'z' and then sum 1
            }
            newMessage += currentChar;                             //We concatenate
        } else {
            newMessage += currentChar;                             //if all was good in the same range, we simply concatenate
        }

    }
        return newMessage;
    }

    // Print the cipher of args[1], args[2], etc. on their own lines
    // shifted by args[0].
    public static void main(String args[]) {

        int length = args.length;                                  //Get the ammount of args
        if (length>1) {
            int shift = Integer.parseInt(args[0]);                 //Get the first arg that is going to be used as shift
            for (int i = 1; i < length; i++) {                     //Loop from 1 to lenght (it does not include 0, because that is the shift)
                System.out.println(caesarCipher(args[i], shift));  //Just print the new encrypted words
            }
        }
    }
}
```

8

CaesarCipher.java

```
package hw01;
public class CaesarCipher {

    // Return a String that is the same as message but where all the alphabetic
    // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
    // All non-alphabetic characters should remain unchanged. For example,
    // caesarCipher("abxy!",  2).equals("cdza!")
    // caesarCipher("cdza!", -2).equals("abxy!")\



public static String CaesarCipher(String message, int shiftAmount) {
    // for each character, c in message:
    //   if c >= 'a' and <= 'z'
    //     shift c by shiftAmount
    //   else if c >= 'A' and <= 'Z'
    //     shift c by shiftAmount
    // add c to output
    String asecretcode = "";
    for(int i = 0; i < message.length(); i++){
        char c = message.charAt(i);
        if(c >= 'a' && c <= 'z') {
            if(c + shiftAmount< 123){//not wrapping
                int ascii = ((int) c + shiftAmount);
                c = (char) ascii;

            }
            else {
                int ascii = ((int) c + shiftAmount) % 123 + 97; //wraping
```

```
            c = (char) ascii;
          }
        }
          else if (c >= 'A' && c <= 'Z') {
            if (c + shiftAmount < 91) {//not wrapping
              int ascii = ((int) c + shiftAmount);
              c = (char) ascii;
            } else {
              int ascii = ((int) c + shiftAmount) % 91 + 65; //wrapping
              c = (char) ascii;

            }
        }
        asecretcode += c;
          }



    return asecretcode;
}

// Print the cipher of args[1], args[2], etc. on their own lines
// shifted by args[0].
public static void main(String args[]) {
    // Read shiftAmount from args[0]
    // for each arg in args[1], args[2], etc.:
    //   print caesarCipher(...)
    int shiftAmount = Integer.valueOf(args[0]);
    for (int i = 1; i < args.length; i++) {

   String acoolcode = CaesarCipher(args[i], shiftAmount);
   System.out.println(acoolcode);

        }
    }
  }
```

9

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

  public static String caesarCipher(String message, int shiftAmount) {
    if (shiftAmount > 26) {
      shiftAmount = shiftAmount % 26;
    }
    else if (shiftAmount < 0) {
      shiftAmount = (shiftAmount % 26) + 26;
    }

    String t = "";
    int l = message.length();
    for (int i = 0; i < l; i++) {
      char ch = message.charAt(i);
      if (Character.isLetter(ch)) {
        if (Character.isLowerCase(ch)) {
          char c = (char) (ch + shiftAmount);
          if (c > 'z') {
            t += (char) (ch - (26 - shiftAmount));
          }
          else {
            t += c;
          }
        }
        else if (Character.isUpperCase(ch)) {
          char c = (char) (ch + shiftAmount);
          if (c > 'Z') {
            t += (char) (ch - (26 - shiftAmount));
          }
          else {
            t += c;
          }
        }
```

```
      }
      else {
        t += ch;
      }
    }
    return t;
  }

  public static void main(String args[]) {
    int s = Integer.parseInt(args[0]);
    for (int i = 1; i < args.length; i++) {
      System.out.println(caesarCipher(args[i], s));
    }
  }
}
```

10

CaesarCipher.java

```
package hw01;

public class CaesarCipher {
	public static void main(String args[]) {
		String shift = args[0];
		int m = Integer.valueOf(shift);
		String result = "";
		for(int k = 1; k < args.length; k++) {
			result += caesarCipher(args[k], m);
			result += "\n";
		}
		System.out.print(result);
}

	public static String caesarCipher(String message, int shift) {
		// StringBuffer object is mutable and its length can be changed
		StringBuffer result= new StringBuffer();
		String msg = message;
		for(int i = 0; i < msg.length(); i++)
		{
			int firstletter = msg.charAt(i);
			char ch = msg.charAt(i);
			// special chars like : should not be shifted
			// and should be printed as it is
			if (!Character.isUpperCase(firstletter) &&
				!Character.isLowerCase(firstletter)) {
				if (ch == '\\') {
					// \n => ch = \\ = msg.charAt(i)
					// 'n' = msg.charAt(i+1)
					// Look at the next character.
					result.append((char)firstletter);
					firstletter = msg.charAt(i + 1);
					i = i+1;
			}
			result.append((char)firstletter);
			continue;
	}
	int mya = 'a';
	if (Character.isUpperCase(firstletter))
		mya = 'A';
	int aftershift = 0;
	int finalresult = 0;
	if (shift > 0)
	{
		shift = ((shift % 26) + 26) % 26;
		aftershift = firstletter + shift;
		//System.out.println("firstletter: " + firstletter + ", shift: " + shift + ", aftershift: " + aftershift);
		finalresult = (aftershift % mya) % 26 + mya;
		//System.out.println("finalresult: " + finalresult);
	}
	//additional code to fix the problem
	else
	{
		int mul = (mya - (firstletter + shift))/26 + 1;
		aftershift = firstletter + shift + 26*mul;
		finalresult = (aftershift % mya) % 26 + mya;
```

```
                        }

                        char charfinal = (char)finalresult;
                        result.append(charfinal);
                }
                //System.out.print(result);
                return result.toString();
        }
}

/*
Testcases:
pass: -10 a b -> q r
pass: -36 a b -> q r
pass: -20 a b -> g h
pass: 10 x y -> h i
pass: 20 x y -> r s
pass: 36 x y -> h i
pass: -10 x y -> n o
pass: 10 a b -> k l
capital:
pass: -10 A B -> Q R
pass: -36 A B -> Q R
pass: -20 A B -> G H
pass: 10 A B -> K L
pass: 10 X Y -> H I
pass: 20 X Y -> R S
pass: 36 X Y -> H I
pass: -10 X Y -> N O

Special characters:
pass: -10 X: Y -> N: O
pass: -10 X\n Y -> N\n O
pass: -10 X\p Y -> N\p O
pass: -10 X\r Y -> N\r O
pass: -10 "X Y" "Y X" -> N O  O N
pass: -54321 "Some test string!" "and another TEST STRING!" -> Lhfx mxlm lmkbgz! tgw tghmaxk MXLM LMKBGZ!
pass: 2 "ab xy!\n",) -> cd za!\n,)
pass: 0 This is an example: abcxyz -> This is an example: abcxyz
pass: 1 This is an example: abcxyz -> Uijt jt bo fybnqmf: bcdyza
pass: 2 This is an example: abcxyz -> Vjku ku cp gzcorng: cdezab
pass: -2 This is an example: abcxyz -> Rfgq gq yl cvyknjc: yzavwx

pass: -270 a b -> q r
pass: -296 a b -> q r
pass: +270 q r -> a b
pass: +296 q r -> o p
pass: 10 q r -> a b
pass: 62 q r -> a b
pass: 88 q r -> a b

*/
```

11

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

        private static int ALPHABET_LENGTH = 'z' - 'a' + 1;

        public static char shift(char target, int shiftAmount, char firstCharacter) {
                return (char) ((target + ALPHABET_LENGTH + (shiftAmount % ALPHABET_LENGTH) - firstCharacter) % ALPHABET_LENGTH + firstCharacter);
        }

    public static String caesarCipher(String message, int shiftAmount) {
        char[] encodedChars = new char[message.length()];

        for (int i=0; i<message.length(); i++) {
                if ('a' <= message.charAt(i) && message.charAt(i) <= 'z') {
                        encodedChars[i] = shift(message.charAt(i), shiftAmount, 'a');
                } else if ('A' <= message.charAt(i) && message.charAt(i) <= 'Z') {
                        encodedChars[i] = shift(message.charAt(i), shiftAmount, 'A');
                } else {
```

```
                encodedChars[i] = message.charAt(i);
            }
        }

        return new String(encodedChars);
    }


    public static void main(String args[]) {
        for (int i=1; i<args.length; i++) {
            System.out.println(caesarCipher(args[i], Integer.parseInt(args[0])));
        }
    }
}
```

12

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

    /**
     * Return a String that is the same as message but where all the alphabetic
    * characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
    * All non-alphabetic characters should remain unchanged. For example,
    * caesarCipher("abxy!",  2).equals("cdza!")
    * caesarCipher("cdza!", -2).equals("abxy!")
    */

        public static int shiftIntegerInRange(int myInt, int start, int end, int shift) {
                /** Take an integer belonging in the interval [start, end] and
                 *  adding the integer shift to it, possibly wrapping around the beginning
                 *  of the interval.
                 */
                int normalizedInt;
                int encodedInt;
                int unnormalizedInt;
                int rangeLength = end - start + 1;

                // Shift myInt so that it lies in [0, rangeLength]
                normalizedInt = myInt - start;
                // Encode using the modulus operator, using a representative int in
                // the range [0, rangeLength]
                int shiftedInt = normalizedInt + shift;
                while (shiftedInt < 0) {
                        shiftedInt += rangeLength;
                }
                encodedInt = shiftedInt % rangeLength;
                // Shift the encoded integer back into the range [start, end]
                unnormalizedInt = encodedInt + start;
                return unnormalizedInt;
        }

        public static char encodeChar(char myChar, int shiftAmount) {
                int upperA = 65;
                int upperZ = 90;
                int lowerA = 97;
                int lowerZ = 122;
                char encodedChar;

                if (Character.isLowerCase(myChar)) {
                        encodedChar = (char) shiftIntegerInRange((int) myChar, lowerA, lowerZ, shiftAmount);
                } else if (Character.isUpperCase(myChar) ){
                        encodedChar = (char) shiftIntegerInRange((int) myChar, upperA, upperZ, shiftAmount);
                } else {
                        encodedChar = myChar;
                }
                return encodedChar;
        }

public static String encodedMessage(String message, int shiftAmount) {
    // for each character, c in message:
    //   if c >= 'a' and <= 'z'
    //     shift c by shiftAmount
```

```
    //   else if c >= 'A' and <= 'Z'
    //     shift c by shiftAmount
    // add c to output

        StringBuilder encoded = new StringBuilder();

        for (int i=0; i < message.length(); i++) {
                char myChar = message.charAt(i);
                char encodedChar = encodeChar(myChar, shiftAmount);
                encoded.append(encodedChar);
        }
        return encoded.toString();
}

public static void testShiftIntegerInRange() {
        int start = 3;
        int end = 9;
        int shift = -2;

        for (int i=start; i<=end; i++) {
        System.out.println(i + " -> " + shiftIntegerInRange(i, start, end, shift));
        }

}
```

```
    // Print the cipher of args[1], args[2], etc. on their own lines
    // shifted by args[0].
    public static void main(String args[]) {

        int shiftAmount = Integer.parseInt(args[0]);

        for (int i=1; i < args.length; i++) {
                String toEncode = args[i];
                System.out.println(encodedMessage(toEncode, shiftAmount));
        }


        //testShiftIntegerInRange();

    }
}
```

13

CaesarCipher.java

```
package hw01;
public class CaesarCipher {
        // Return a String that is the same as message but where all the alphabetic
        // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
        // All non-alphabetic characters should remain unchanged. For example,
// caesarCipher("ab xy!\n",  2).equals("cd za!\n")
// caesarCipher("cd za!\n", -2).equals("ab xy!\n")
public static int convertShift(int shiftAmount){
        if(shiftAmount > 26 || shiftAmount < -26){
                if(shiftAmount > 26){
                        shiftAmount -= (shiftAmount / 26) * 26;
                }
                else{
                        shiftAmount -= (shiftAmount / 26) * 26;
                }
        }
        return shiftAmount;
}
public static String caesarCipher(String message, int shiftAmount) {
        char[] messageChar= message.toCharArray();
        for(int i = 0; i < messageChar.length; i++){
                char c = messageChar[i];
                //if lower case
                if(c >= 'a' && c <= 'z'){
                        if((c + shiftAmount) < 'a'){
                                messageChar[i] = (char)(c + shiftAmount + 26);
                        }
                        else if((c + shiftAmount) > 'z'){
                                messageChar[i] = (char)(c + shiftAmount - 26);
                        }
```

```
                else{
                        messageChar[i] =  (char)(c + shiftAmount);
                }
            }
            //if higher case
            else if(c >= 'A' && c <= 'Z'){
                if((c + shiftAmount) < 'A'){
                        messageChar[i] = (char)(c + shiftAmount + 26);
                }
                else if((c + shiftAmount) > 'Z'){
                        messageChar[i] = (char)(c + shiftAmount - 26);
                }
                else{
                        messageChar[i] =  (char)(c + shiftAmount);
                }
            }
            //if not alphabet
            else{
                messageChar[i] = c;
            }
        }
        String output = new String(messageChar);
        return output;
}
public static void main(String args[]) {
            // Read shiftAmount from args[0]
            int shiftAmount = convertShift(Integer.valueOf(args[0]));
//          System.out.println(shiftAmount);
            for(int i = 1; i < args.length; i++){
                System.out.println(caesarCipher(args[i],shiftAmount));
            }
    }
}
```

14

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

  // Return a String that is the same as message but where all the alphabetic
  // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
  // All non-alphabetic characters should remain unchanged. For example,
  // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
  // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
  public static String caesarCipher(String message, int shiftAmount) {
    StringBuilder builder = new StringBuilder();
    if (shiftAmount < 0) {
      shiftAmount = shiftAmount % 26 + 26;
    }
    for (int i = 0; i < message.length(); i++) {
      char c = message.charAt(i);
      if (c >= 'a' && c <= 'z') {
        c = (char)(((int)c - 'a' + shiftAmount) % 26 + 'a');
      } else if (c >= 'A' && c <= 'Z') {
        c = (char)(((int)c - 'A' + shiftAmount) % 26 + 'A');
      }
      builder.append(c);
    }
    return builder.toString();
  }

  // Print the cipher of args[1], args[2], etc. on their own lines
  // shifted by args[0].
  public static void main(String args[]) {
    int shiftAmount = Integer.valueOf(args[0]);
    for (int i = 1; i < args.length; i++) {
      System.out.println(caesarCipher(args[i], shiftAmount));
    }
  }
}
```

15

CaesarCipher.java

```
//Yanyan Chen 20332382 CS01B Assignment1
package hw01;


public class CaesarCipher {
    // Return a String that is the same as message but where all the alphabetic
    // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
    // All non-alphabetic characters should remain unchanged. For example,
    // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
    // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
    public static String caesarCipher(String message, int shiftAmount) {
        String result = "";
        for (int i = 0; i < message.length(); i++) {
            char c = message.charAt(i);

            //if c is lower case
            if (c - 'a' < 26 && c - 'a' >= 0) {
                c = (char) (c + shiftAmount % 26);

                // if out of bound
                if (c < 'a') {
                    c += 26;
                } else if (c > 'z') {
                    c -= 26;
                }
            //if c is upper case
            } else if (c - 'A' < 26 && c - 'A' >= 0) {
                c = (char) (c + shiftAmount % 26);

                // if out of bound
                if (c < 'A') {
                    c += 26;
                } else if (c > 'Z') {
                    c -= 26;
                }
            }
            result += c;
        }
        return result;
    }

    // Print the cipher of args[1], args[2], etc. on their own lines
    // shifted by args[0].
    public static void main(String args[]) {
        // Read shiftAmount from args[0]
        // for each arg in args[1], args[2], etc.:
        //   print caesarCipher(...)
        int shiftAmount = Integer.parseInt(args[0]);
        for (int i = 1; i < args.length; i++) {
            System.out.println(caesarCipher(args[i], shiftAmount));
        }
    }
}
```

16

CaesarCipher.java

```
package hw01;
public class CaesarCipher{
        // Return a String that is the same as message but where all the alphabetic
    // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
    // All non-alphabetic characters should remain unchanged. For example,
    // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
    // caesarCipher("cd za!\n", -2).equals("ab xy!\n")

    public static String caesarCipher (String message, int shiftAmount) {
        // for each character, c in message:
        //   if c >= 'a' and <= 'z'
        //     shift c by shiftAmount
        //   else if c >= 'A' and <= 'Z'
        //     shift c by shiftAmount
        // add c to output
            String newMessage ="";
```

```
for (int i=0;i < message.length(); i++) {
            char c=message.charAt(i);
            if (c>='a'&&c<='z') {
                        c=(char)((c+shiftAmount-97)%26+97);
            if (c<97){
                        c=(char)((c-97)+123);
                        }
            else if (c>123) {
                        c=(char)((c-123)+97);
            }
                        newMessage+=c;
            }
            else if (c>='A'&&c<='Z') {
                        c=(char)((c+shiftAmount-65)%26+65);
                        if (c<65){
                        c=(char)((c-65)+91);
                        }
            else if (c>123) {
                        c=(char)((c-91)+65);
            }
                        newMessage+=c;
            }
            else {
                        newMessage+=c;
            }
            }
            return  newMessage;
    }


    // Print the cipher of args[1], args[2], etc. on their own lines
    // shifted by args[0].
    public static void main(String args[]) {
        // Read shiftAmount from args[0]
        // for each arg in args[1], args[2], etc.:
        //   print caesarCipher(...)

    for(int i=1;i<=args.length;i++) {
            int shiftAmount = Integer.valueOf(args[0]);
            String message =args[i];

            String newMessage= caesarCipher(message,shiftAmount);
            System.out.println(newMessage);
    }
    }
}
```

17

CaesarCipher.java

```
package hw01;

public class CaesarCipher {
        // Return a String that is the same as message but where all the alphabetic
        // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
        // All non-alphabetic characters should remain unchanged. For example,
        // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
        // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
        public static String caesarCipher(String message, int shiftAmount) {
            // for each character, c in message:
            //   if c >= 'a' and <= 'z'
            //     shift c by shiftAmount
            //   else if c >= 'A' and <= 'Z'
            //     shift c by shiftAmount
            // add c to output
            char[] New_message = message.toCharArray();
            for (int i = 0; i < message.length(); i++){
                if(New_message[i] >= 'a' && New_message[i] <= 'z'){
                    int Convert_message = (int)New_message[i] + (shiftAmount % 26);
                    if (Convert_message < 'a'){
                        Convert_message = Convert_message +  26;
                    }
                    else if (Convert_message > 'z'){
                        Convert_message = Convert_message - 26;
                    }
```

```
            New_message[i] = (char) Convert_message;
        }
        else if(New_message[i] >= 'A' && New_message[i] <= 'Z') {
            int Convert_message = (int)New_message[i] + (shiftAmount % 26);
            if (Convert_message < 'A'){
                Convert_message = Convert_message +  26;
            }
            else if (Convert_message > 'Z'){
                Convert_message = Convert_message - 26;
            }
            New_message[i] = (char) Convert_message;
        }
    }
    return String.valueOf(New_message);
}

// Print the cipher of args[1], args[2], etc. on their own lines
// shifted by args[0].
public static void main(String args[]) {
    // Read shiftAmount from args[0]
    // for each arg in args[1], args[2], etc.:
    //   print caesarCipher(...)
    Integer shiftAmount = Integer.valueOf(args[0]);
    String message = "";
    for (int i = 1; i < args.length; i++){
        message = message + args[i] + "\n";
    }
    caesarCipher(message,shiftAmount);
    System.out.print(caesarCipher(message,shiftAmount));

  }
}
```

18

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

        // Return a String that is the same as message but where all the alphabetic
        // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
        // All non-alphabetic characters should remain unchanged. For example,
        // caesarCipher("ab xy!\n", 2).equals("cd za!\n")
        // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
        public static String caesarCipher(String message, int shiftAmount) {
                String Message = "";
                for (int i = 0; i < message.length(); i++) {
                        char c = message.charAt(i);
                        if (c >= 'a' && c <= 'z') {
                                c = (char) ((c - 97) + (shiftAmount % 26) + 97);
                                if (c < 97) {
                                        c = (char) ((c - 97) + 123);
                                }
                                else if(c >= 123){
                                        c = (char) ((c - 123) + 97);
                                }
                                Message += c;
                        }
                        else if (c >= 'A' && c <= 'Z') {
                                c = (char) ((c - 65) + (shiftAmount % 26) + 65);
                                if (c < 65) {
                                        c = (char) ((c - 65) + 91);
                                }
                                else if ( c >= 91){
                                        c = (char) ((c - 91) + 65);
                                }
                                Message += c;
                        }
                        else{
                                c = c;
                                Message += c;
                        }
}
// for each character, c in message:
```

```
        // if c >= 'a' and <= 'z'
        // shift c by shiftAmount
        // else if c >= 'A' and <= 'Z'
        // shift c by shiftAmount
        // add c to output
        return Message;
}

// Print the cipher of args[1], args[2], etc. on their own lines
// shifted by args[0].
public static void main(String args[]) {
        // Read shiftAmount from args[0]
        // for each arg in args[1], args[2], etc.:
        // print caesarCipher(...)
        // String message = "This is an example: abcxyz";
        // int shiftAmount = 1;
        // System.out.println(caesarCipher(message, shiftAmount));
        int shiftAmount = Integer.valueOf(args[0]);
        for (int i = 1; i < args.length; i++) {
                String message = args[i];
                String Message = caesarCipher(message, shiftAmount);
                // System.out.println(Message);
                System.out.println(Message);
        }
}
```

19

CaesarCipher.java

```
package hw01;
public class CaesarCipher{
   public static String caesarCipher(String message, int shiftAmount) {
      StringBuilder newMessage = new StringBuilder(message);
      for (int i = 0; i < message.length(); i++) {
         int currentAscii = message.charAt(i);
         if(currentAscii >= 'a' && currentAscii <= 'z') {
            int modShift = (shiftAmount % 26);
            int newAscii = currentAscii + modShift;
            if (newAscii >'z'){newAscii-=26;}
            else if (newAscii <'a'){newAscii += 26;}
            newMessage.setCharAt(i, (char) newAscii);
         }
         else if(currentAscii >='A' && currentAscii <='Z'){
            int modShift = (shiftAmount % 26);
            int newAscii = currentAscii + modShift;
            if (newAscii >'Z'){newAscii -= 26;}
            else if (newAscii <'A'){newAscii += 26;}
            newMessage.setCharAt(i, (char) newAscii);
         }
      }
      return newMessage.toString();
   }
   public static void main(String[] args) {
         int shiftAmount = Integer.parseInt(args[0]);
         String message="";
         for (int i=1; i<args.length; ++i){
            message+= args[i]+"\n";
         }
         System.out.print(caesarCipher(message,shiftAmount));
      }
   }
```

20

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

   public static String caesarCipher(String message, int shiftAmount) {
                  if( Math.abs(shiftAmount)  >= 26)
                           shiftAmount = shiftAmount % 26;
```

```
StringBuilder sb = new StringBuilder();

char[] arr = message.toCharArray();

for(int i=0; i< arr.length; i++ ) {

        int _charCode = arr[i];

//65 - 90 A-Z
//97 - 122 a-z
        int start = 97;
        int end = 122;
        if( (_charCode <= 90) &&  ( _charCode >= 65) ) {
                start = 65;
                end = 90;
        }
        else if ( !((_charCode >= 97) &&  ( _charCode <= 122))) {
                //foreign character
                sb.append(arr[i]);
                continue;
        }

        _charCode+= shiftAmount;

        if(_charCode > end) {
                _charCode = start + (_charCode - (end+1));
        }  else if (_charCode < start) {
                _charCode = end - ( (start-1) - _charCode);
        }

        sb.append((char) _charCode);
    }

    return sb.toString();

  }

  public static void main(String args[]) {

    int shift = Integer.parseInt(args[0]);

    for(int i=1;i<args.length;i++) {
        System.out.println(caesarCipher(args[i], shift));
    }
  }

}
```

21

CaesarCipher.java

```
/* HW01 of CS1B. Caesar Chiper. Oct. 2018. Pam Kung

to compile in the command line:

$ javac hw01/CaesarCipher.java

Usage:

1. inp. arguments: shiftAmount str1 str2 ...

2. shiftAmount : integer to shift the letters.

3. Non-alphabets are not shifted.

3. shiftAmount is circular (modulo) in 26 (= # of letters). i.e., it
wraps around if it's > 26 || < -26.  e.g. all integer multiples of 26,
+/-26, +/-52, etc => 0 (no shift), -27 & -1 => 25, -28, -2 => 24

examples of running on the command line:

java hw01.CaesarCipher 1 This is an example: abcxyz

Uijt
jt
```

bo
fybnqmf
bcdyza

java hw01.CaesarCipher 2 This is an example: abcxyz

Vjku
ku
cp
gzcorng:
cdezab

java hw01/CaesarCipher -2 This is an example: abcxyz

Rfgq
gq
yl
cvyknjc:
yzavwx
*/

```
package hw01;

public class CaesarCipher
{
// Return a String that all alphabetic characters ('a' to 'z' and
// 'A' to 'Z') shifted by shiftAmount.  All non-alphabetic
// characters remain unchanged. For example,
// caesarCipher("abxy!", 2).equals("cdza!")  caesarCipher("cdza!",
// -2).equals("abxy!")
public static String caesarCipher(String message, int shiftAmount)
{
  shiftAmount %= 26; // -> [-25, 25]
  if (shiftAmount == 0) return message;

  // since char. is *unsigned*, have to do this below for negative
  shiftAmount += 26; // [-25, 25] -> [1, 51]
  shiftAmount %= 26; // [1, 51] -> [0, 25]

  // use a char array which allows changing in place (mutable)
  // (alternative can also start w/ an empty String then
  // concatenate)
   char[] msgOut = message.toCharArray();

  for (int j = 0; j < msgOut.length; j++)
  {
    // skip non-alphabets
    if (!(msgOut[j] >= 'a' && msgOut[j] <= 'z') && !(msgOut[j] >= 'A' && msgOut[j] <= 'Z'))
      continue;

    // 'a'-'z': ASCII 97 - 122, 'A'-'Z': 95 - 122, so only need
    // to test if it's is < 'a'
    boolean isUpper = (msgOut[j] < 'a');
    if (isUpper)
      msgOut[j] -= 'A';
    else
      msgOut[j] -= 'a';

    msgOut[j] += shiftAmount;
    msgOut[j] %= 26;

    if (isUpper)
      msgOut[j] += 'A';
    else
      msgOut[j] += 'a';
  }
  return new String(msgOut);
} // end of caesarCipher

// Print the cipher of args[1], args[2], etc. on their own lines
// shifted by args[0].
public static void main(String[] args)
{
  if (args.length == 0) // no input arguments
  {
    System.out.println("Usage:\nshiftAmount str1 str2...");
```

```
      System.exit(0);
    }

    int key = Integer.parseInt(args[0]);

    // for (int i = 1; i < args.length; i++)
    //  System.out.println(args[i]);

    // System.out.println("\nShifted by " + key + ":");

    for (int i = 1; i < args.length; i++)
      System.out.println(caesarCipher(args[i], key));
  }
}
```

22

CaesarCipher.java

```
package hw01;

import static java.lang.Character.*;

public class CaesarCipher {
    public static int calculateShift(int shift, int base){
        // Calculates the shift between 0 and base
        int newShift;
        newShift = (((shift % base) + base) % base);
        return newShift;
    }

    // Return a String that is the same as message but where all the alphabetic
    // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
    // All non-alphabetic characters should remain unchanged. For example,
    // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
    // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
    public static String caesarCipher(String message, int shift) {
        // for each character, c in message:
        //   if c >= 'a' and <= 'z'
        //     shift c by shiftAmount
        //   else if c >= 'A' and <= 'Z'
        //     shift c by shiftAmount
        // add c to output
        final int UPPERCASE_A = 65;
        final int LOWERCASE_A = 97;
        final int NUM_OF_LTRS = 26;

        char c;
        int intC;
        String output = "";
        shift = shift % NUM_OF_LTRS;
        shift = calculateShift(shift, NUM_OF_LTRS);
        for (int i = 0; i < message.length(); i++) {
            c = message.charAt(i);
            intC = (int) c;
            if (isLowerCase(c)) {
                intC = intC - LOWERCASE_A;
                c = (char)((intC + shift) % NUM_OF_LTRS + LOWERCASE_A);
            } else if (isUpperCase(c)) {
                intC = intC - UPPERCASE_A;
                c = (char)((intC + shift) % NUM_OF_LTRS + UPPERCASE_A);
            }
            output = output + c;
        }
        return output;
    }

    // Print the cipher of args[1], args[2], etc. on their own lines
    // shifted by args[0].
    public static void main(String args[]) {
        // Read shiftAmount from args[0]
        // for each arg in args[1], args[2], etc.:
        //   print caesarCipher(...)
        int shiftValue = Integer.parseInt(args[0]);
```

```
        for (int i = 1; i < args.length; i++){
            //System.out.println(args[i]);
            System.out.println(caesarCipher(args[i], shiftValue));
            //System.out.println("---------");
        }
    }
}
```

23

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

        //notes/reminders
                //check modulo in rotator methods



        //instructions{
                // Return a String that is the same as message but where all the alphabetic
                // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
                // All non-alphabetic characters should remain unchanged. For example,
                // caesarCipher("abxy!",  2).equals("cdza!")
                // caesarCipher("cdza!", -2).equals("abxy!")
//}
//rotates the UpperCase Chars
  public static char RotateAscii97To122(long x) {
     long a = x+7;
     long b = 26;
     return (char)((a % b) + 97);
    }

//rotates the LowerCase Chars
  public static char RotateAscii65To90(long x) {
           long a = x-13;
           long b = 26;
        return (char)((a % b) + 65 );
    }

          //Shifts the given char depending on what it is i.e: being a uppercase,lowercase, or other
         public static char caesarShifter(char a, long b) {
          //char that will be returned
          char postShift;

          //shift amount
          final long shifty = b;

boolean UpperCheck = Character.isUpperCase(a);

boolean LowerCheck = Character.isLowerCase(a);

char charPlusShift = (char)(a + shifty) ;

if(UpperCheck) {
          //rotate34To59
  postShift = RotateAscii65To90(charPlusShift);
}
 else if (LowerCheck) {
 //rotate66To91
    postShift = RotateAscii97To122(charPlusShift);
```

```
    }
                    else {
  //do nothing
     postShift = a;
    }
            return postShift ;
     }

         //instructionForCaesarCipherMethod{
     // for each character, c in message:
     //   if c >= 'a' and <= 'z'
     //     shift c by shiftAmount
     //   else if c >= 'A' and <= 'Z'
     //     shift c by shiftAmount
     // add c to output
public static String caesarCipher(String message, long shiftAmount) {
//shift amount
final long m = shiftAmount;

//how long the given char string is
 int k = message.length();

     //Ciphered encryption that will be printed
     String encryptedString = "";

 //for loop to encrypt the given chars until the string is done
     for (int j = 0 ; j < k ; j++) {

          //ascii # of the char at a given point is a unicode value
          //this char will go into caesarShifter (char a,???null???) **^method above^**
          char x  = (message.charAt(j));

          // casting for char after shiftAmount
          char c = caesarShifter(x, m);

          //adds on each char until there is no chars left to add
          encryptedString = encryptedString + c;
     }
     return encryptedString ;
}

//outcome{
    // Print the cipher of args[1], args[2], etc. on their own lines

    // shifted by args[0].
//}
    public static void main(String args[]) {
        // o Read shiftAmount from args[0]
         // for each arg in args[1], args[2], etc.:

         int shiftA = Integer.valueOf(args[0]);

        //   print caesarCipher(...)
         for (int i = 1 ; i < args.length ; i++) {

           String encrypt = args[i].toString();

           String cipher =  caesarCipher(encrypt, shiftA);
```

```
                System.out.println(cipher);
            }
        }
    }
```

24

CaesarCipher.java

```
package hw01;

public class CaesarCipher {
    public static String caesarCipher(String message, int shiftAmount) {
            String output = "";

            for(int i = 0; i < message.length(); i++) {
                // Convert each letter into ascii. Establish remainder based on ascii table.
                int ascii = (int) message.toLowerCase().charAt(i);
                int updated_shift = shiftAmount % 255;

                // Narrow down the range of letters targeted from ascii table.
                if(ascii > 122 || ascii < 97) {
                    String unmodified = Character.toString((char)ascii);
                    output = output + unmodified;
                    continue;
                } else {
                    // Restrict ascii to affect only A-Z alphabet
                    if(ascii > 26) {
                        updated_shift = updated_shift % 26;
                    }

                    // Shift values and output result
                    int update_ASCII = ascii + updated_shift;
                    if(update_ASCII > 122) {
                        update_ASCII -= 26;
                    } else if(update_ASCII < 97) {
                        update_ASCII += 26;
                    }
                    String update_letter = Character.toString((char)update_ASCII);

                    output = output + update_letter;
                }
            }
            return output;
}

public static void main(String args[]) {
    if(args.length > 1) {
            int shiftAmount = Integer.valueOf(args[0]);
            for(int i = 1; i < args.length; i++) {
                System.out.println(caesarCipher(args[i], shiftAmount));
            }
        }
        System.out.println(caesarCipher("ab xy!\n", -127657345));
    }
}
```

25

CaesarCipher.java

```
package hw01;

public class CaesarCipher
{
   // Return a String that is the same as message but where all the alphabetic
   // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
   // All non-alphabetic characters should remain unchanged. For example,
   // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
   // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
        public static String caesarCipher(String message, int shiftAmount)
        {
            String output;
     int key;
            //Implemented StringBuilder because it is faster and more efficient.
```

```
StringBuilder encryptedMsg = new StringBuilder();

//Splits up message into Char array and loops through each Char.
            for (char i : message.toCharArray())
{
                    key = shiftAmount % 26 + 26; //Sets key according to shift.
   //Only changes Char if it is a letter.
   if (Character.isLetter(i))
   {
      if (Character.isUpperCase(i))
      {
         encryptedMsg.append((char) (((i - 65 + key) % 26) + 65));
      }
                        else
      {
         encryptedMsg.append((char) (((i - 97 + key) % 26) + 97));
      }
   }
                    else
      {
         encryptedMsg.append(i);
      }
}
            output = encryptedMsg.toString();
return (output);
   }

   public static void main(String args[])
      {
            int i;

            i = 1;
            while (i < args.length)
            {
                  System.out.println(caesarCipher(args[i], Integer.parseInt(args[0])));
                  i += 1;
            }
      }
}
```

26

CaesarCipher.java

```
package hw01;

public class CaesarCipher
{
public static void main(String args[]) {
// When you run a program on the command line, like// $ java ArgsPrinter "This is" a test// then you will see output like this:// args.length == 3// args[0] == "This is"// args[1] == "a"// args[2] == "test"System.out.println("args.length == " + args.length);for (int i = 0; i < args.length; i++) {System.out.println("args[" + i + "] == \"" + args[i] + "\"");}}}
      int shiftAmount = Integer.valueOf(args[0]);

      //StringBuilder sb = new StringBuilder(args().length());
      for (int i =1; i < args.length; i++) {
            String plaintext = args[i];
            String ciphertext  = caesarCiper(plaintext, shiftAmount);


      // System.out.println("args[" + i + "] == \"" + args[i] + "\"");
       System.out.println(ciphertext);
 }
}

public static String caesarCiper(String s, int shiftAmount) {
      String str ="";
      for (int i = 0; i <s.length(); i++) {
      char c = s.charAt(i);
      if (c >= 'a' && c <= 'z') {
            c = (char) ((c-97)+ (shiftAmount % 26) + 97);
            if ( c< 97) {
                  c = (char) ((c-97) + 123);
            }
            else if (c >= 123) {
                  c = (char) ((c -123) + 97);
```

```
                }
                str += c;
                }
        else if ( c >= 'A' && c <= 'Z') {
                c = (char) ((c - 65) + (shiftAmount% 26) + 65);
                if (c < 65) {
                        c = (char) (( c- 65) + 91);

                }
                else if (c>=91) {
                        c = (char) ((c-91) + 65);
                }
                str += c;
        }
        else {
                str += c;
        }
        }

        return str;
}

}
```

27

CaesarCipher.java

```
package hw01;
import java.lang.Math.*;

public class CaesarCipher {
   // Return a String that is the same as message but where all the alphabetic
   // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
   // All non-alphabetic characters should remain unchanged. For example,
   // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
   // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
   public static String caesarCipher(String message, int shiftAmount) {
      // for each character, c in message:
      //   if c >= 'a' and <= 'z'
      //     shift c by shiftAmount
      //   else if c >= 'A' and <= 'Z'
      //     shift c by shiftAmount
      // add c to output
      String s = "";
      for(int i = 0; i < message.length(); i ++){

         char c = message.charAt(i);
         if (c >= 'a' && c <= 'z'){
            int shiftTest = message.charAt(i) - 97 + (shiftAmount % 26);
      if (shiftTest < 0){
         s += (char)(((shiftTest) % 26) + 123);
      }
      else{
         s += (char)(((shiftTest) % 26) + 97);
      }

   }
   else if (c >= 'A' && c <= 'Z' ){
      int shiftTest = message.charAt(i) - 65 + (shiftAmount % 26);
      if(shiftTest < 0){
         s += (char)(((shiftTest) % 26) + 91);
      }
      else{
         s += (char)(((shiftTest) % 26) + 65);
      }
   }
   else{
      s += message.charAt(i);
   }



   //char c = (char)(message.charAt(i) + shiftAmount);
```

```
        //if (c > 'z')
        //s += (char)(message.charAt(i) - (26-shiftAmount));
        //else
        //s += (char)(message.charAt(i) + shiftAmount);
    }
    return s;

  }

  // Print the cipher of args[1], args[2], etc. on their own lines
  // shifted by args[0].
  public static void main(String args[]) {
    // Read shiftAmount from args[0]
    // for each arg in args[1], args[2], etc.:
    //   print caesarCipher(...)
    String[] expression = args;

    for(int i = 0; i < expression.length - 1; i ++) {
      int s = Integer.valueOf(expression[0]);
      int j = i + 1;
      System.out.println(caesarCipher(expression[j], s));
    }

  }
}
```

28

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

	public static String caesarCipher(int shiftAmount, String message) {

		String output = "";
		int fixedShift = shiftAmount % 26;

		for (int i = 0; i < message.length(); i++) {

			char c = message.charAt(i);

			if (Character.isLetter(c)) {

				c += fixedShift;

				if (Character.isLowerCase(message.charAt(i)) && c > 'z'

						|| (Character.isUpperCase(message.charAt(i)) && c > 'Z')) {

					c -= 26;
				}
				else if (Character.isLowerCase(message.charAt(i)) && c < 'a'

						|| (Character.isUpperCase(message.charAt(i)) && c < 'A')) {

					c += 26;
			}


			}

			output += c;
		}

		return output;
	}

	public static void main(String[] args) {

if (args.length < 2) {
	System.out.println("Usage: java hw01/CaesarCipher shiftAmt message");

	return;
}
```

```
        int shiftAmt = Integer.parseInt(args[0]);

        for (int i = 1; i < args.length; i++) {
            caesarCipher(shiftAmt, args[i]);

            System.out.println(caesarCipher(shiftAmt, args[i]));

//                          System.out.println(args[i]);

                    }

//      int shiftAmt = Integer.parseInt(args[0]);
//      caesarCipher(shiftAmt, args[1]);



//      caesarCipher(shiftAmt, "This is an example: abcxyz");
//                  System.out.println("------------------------");
//                  caesarCipher(1, "This is an example: abcxyz");
//                  caesarCipher(26, "This is an example: abcxyz");


        }
}
```

29

CaesarCipher.java

```
package hw01;

public class CaesarCipher {

        public static String caesarCipher(String message, int shiftAmount) {
                String out = "";
                for(int i = 0; i < message.length();i++) {
                        // process each char
                        char ch = message.charAt(i);
                        if (ch >= 'a' & ch <= 'z') {
                                ch += shiftAmount;
                                // less than a, wraps from z
                                if (ch < 'a') {
                                        ch = (char) ('z'+1 - ('a' - ch));
                                }
                                if (ch > 'z') {
                                        ch = (char) ('a'-1 + ch - 'z');
                                }
                        } else if (ch >= 'A' & ch <= 'Z') {
                                ch += shiftAmount;
                                if (ch < 'A' ) {
                                        ch = (char) ('Z'+1-('A'-ch));
                                }
                                if (ch > 'Z') {
                                        ch = (char) ('A'-1 + ch - 'Z');
                                }
                        }
                        out += ch;
                }
                return out;
        }

        public static void main(String[] args) {

                int shiftNum = Integer.parseInt(args[0]);
                if (shiftNum > 26)
                        shiftNum = shiftNum % 26;
                else if (shiftNum < -25) {
                        // mod and add -ve sign
                        shiftNum = Math.abs(shiftNum);
                        shiftNum = shiftNum % 26;
                        shiftNum = -shiftNum;
                }

                for(int i = 1; i < (args.length); i++) {
```

```
                System.out.println(caesarCipher(args[i], shiftNum));
            }

        }

}
```

30

CaesarCipher.java

```
package hw01;

public class CaesarCipher {
        // Return a String that is the same as message but where all the alphabetic
        // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
        // All non-alphabetic characters should remain unchanged. For example,
        // caesarCipher("ab xy!\n", 2).equals("cd za!\n")
        // caesarCipher("cd za!\n", -2).equals("ab xy!\n")
        public static String caesarCipher(String message, int shiftAmount) {
                // for each character, c in message:
                // if c >= 'a' and <= 'z'
                // shift c by shiftAmount
                // else if c >= 'A' and <= 'Z'
                // shift c by shiftAmount
                // add c to output

                StringBuilder result = new StringBuilder();
                char c;
                int remainder = shiftAmount % 26;

                for (int i = 0; i < message.length(); i++) {
                        c = message.charAt(i);

                        if ((Character.isLowerCase(message.charAt(i))) == true) {
                                if (c >= 'a' && c <= 'z') {
                                        c = (char) (((int) message.charAt(i) + shiftAmount - 97) % 26 + 97);
                                } else {
                                        c = (char) (((int) message.charAt(i) + remainder));
                                }
                        } else if ((Character.isUpperCase(message.charAt(i))) == true) {
                                if (c >= 'A' && c <= 'Z') {
                                        c = (char) (((int) message.charAt(i) + shiftAmount - 65) % 26 + 65);
                                } else {
                                        c = (char) (((int) message.charAt(i) + remainder));
                                }
                        }

                        result.append(c);

                }
                return result.toString();
        }

        // Print the cipher of args[1], args[2], etc. on their own lines
        // shifted by args[0].

        public static void main(String args[]) {
                // Read shiftAmount from args[0]
                // for each arg in args[1], args[2], etc.:
                // print caesarCipher(...)

                //String character = new String();
                //character = "This is an example: abcxyz";

                if (args.length > 1)
                {
                        int shiftAmount = Integer.parseInt(args[0]);

                        for (int i = 1; i < args.length; i++)
                        {
                                System.out.println(caesarCipher(args[i], shiftAmount));
                        }
                }
```

```
        }
}
```

31

CaesarCipher.java

```
package hw01;

public class CaesarCipher
{

  public static String caesarCipher(String message, int shiftAmount)
  {
    char   myChar;
    char   cipherChar;
    String resStr = "";

    for (int i = 0; i < message.length(); i++)
    {
      myChar = message.charAt(i);
      if (myChar >= 'A' && myChar <= 'Z')
      {
        cipherChar = (char) ((((int) myChar +
                     shiftAmount - 65) % 26 + 26) % 26 + 65);
      }
      else if (myChar >= 'a' && myChar <= 'z')
      {
        cipherChar = (char) ((((int) myChar +
                     shiftAmount - 97) % 26 + 26) % 26 + 97);
      }
      else
      {
        cipherChar = myChar;
      }
      resStr += cipherChar;
    }
    return resStr;
}

public static void main(String[] args)
{

  if (args.length > 1)
  {
    for (int i = 1; i < args.length; i++)
    {
      System.out.println(caesarCipher(args[i], Integer.parseInt
          (args[0])));
      }
    }
    else
      System.out.println("No command line " +
                         "arguments found.");
  }
}
```

32

CaesarCipher.java

```
package hw01;


public class CaesarCipher {

   // Return a String that is the same as message but where all the alphabetic
   // characters ('a' to 'z' and 'A' to 'Z') have been shifted by shiftAmount.
   // All non-alphabetic characters should remain unchanged. For example,
   // caesarCipher("ab xy!\n",  2).equals("cd za!\n")
   // caesarCipher("cd za!\n", -2).equals("ab xy!\n")

        public static String caesarCipher(String message, int shiftAmount) {
      String newName = "";
```

```
String addOn =  "";
if(shiftAmount<0) {
		shiftAmount = (shiftAmount % 26) +26;
}
		for (int i = 0; i < message.length(); i++) {
			char letter = message.charAt(i);
			if('a'<= letter && letter <='z') {
				char newLetter = (char) ((letter-'a'+shiftAmount) % 26);
				char tyName = (char)('a' + newLetter);
				addOn = String.valueOf(tyName);


			}
			else	if('A'<= letter && letter <='Z') {
				char newLetter = (char) ((letter-'A'+shiftAmount) % 26);
				char tyName = (char)('A' + newLetter);
				addOn = String.valueOf(tyName);

			}
			else {
				addOn = String.valueOf(letter);
			}
			newName += addOn;
		}
		return newName;
	}
```

```
   // Print the cipher of args[1], args[2], etc. on their own lines
   // shifted by args[0].
   public static void main(String[] args) {
			for(int i = 1; i<args.length; i++) {
				System.out.println(caesarCipher(args[i], Integer.parseInt(args[0])));
			}
		// Read shiftAmount from args[0]
      // for each arg in args[1], args[2], etc.:
      //   print caesarCipher(...)
   }
}
```

33

CaesarCipher.java

```
package hw01;

public class CaesarCipher {
   // Shift alphabetical characters by shift amount
   public static String caesarCipher(String message, int shiftAmount) {
      String out = "";

      if (shiftAmount < 0)
         shiftAmount = 26 + shiftAmount % 26;

      for (int i = 0; i < message.length(); i++) {
         char glyph = message.charAt(i);
         if (64 < glyph && glyph < 91)
            out += (char)((glyph - 65 + shiftAmount) % 26 + 65);
         else if (96 < glyph && glyph < 123)
            out += (char)((glyph - 97 + shiftAmount) % 26 + 97);
         else
            out += glyph;
      }

      return out;
   }

   // Decode Caesar cipher messages
   public static void main(String args[]) {
      int shiftAmount = Integer.valueOf(args[0]);

      for (int i = 1; i < args.length; i++)
         System.out.println(caesarCipher(args[i], shiftAmount));
```

```
    }
}
```