```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Collections;

public class Assignment {
    // a) Metoder
    /**
     * Returns the sum of all the received numbers.
     */
    public int addThreeNumbers(int i, int j, int k) {
        return i+j+k;
    }


    // b) if else
    /**
     * Evaluates if the received number is
     * Small (less than 100).
     * Big (greater than 1000).
     * Medium (not small or big)
     */
    public String isNumberSmallMediumOrBig(int number){
        if (number < 100) return "Small";
        if (number > 1000) return "Big";
        return "Medium";
    }

    // c) switch
    /**
     * Prints course name for provided course code.
     * ADTS1600 -> Interaksjonsdesign og Prototyping
     * DAPE1400 -> Programmering
     * DATA1200 -> Webutvikling og inkluderende design
     * DATA1100 -> Teknologi og samfunn for programmerere
     * or "Unknown" if none of the above.
     */
    public void printCourseName(String courseCode){
 String name;
        switch (courseCode) {
            case "ADTS1600":
                name = "Interaksjonsdesign og Prototyping"; break;
            case "DAPE1400":
                name = "Programmering"; break;
            case "DATA1200":
                name = "Webutvikling og inkluderende design"; break;
            case "DATA1100":
                name = "Teknologi og samfunn for programmerere";
break;
            default:
                name = "Unknown";
        }
        System.out.println(name);
    }
    // d) Strings
    /**
     * Returns true if provided color is represented in the
Norwegian flag.
     * Color input is lowercase only.
     */
    public boolean isColorInNorwegianFlag(String color){
        return color.equals("rød") || color.equals("rod") ||
color.equals("red")
            || color.equals("hvit") || color.equals("white")
            || color.equals("blå") || color.equals("bla") ||
color.equals("blue");
    }

    /**
     * Returns the combined length of the provided Strings.
     */
    public int combinedLength(String s1, String s2){
      int len1 = 0;
      int len2 = 0;

    if (s1 != null) {
        len1 = s1.length();
    }

    if (s2 != null) {
        len2 = s2.length();
    }

    return len1 + len2;
    }


    /**
     * Return true if string is shorter than or equal to maxChar
characters and longer then or equal to minChar characters.
     * hint: https://www.w3schools.com/java/ref_string_length.asp
     */
    public Boolean checkIfStringIsWithinCorrectLength(String string,
int maxChar, int minChar){
        int len = string.length();
        if (len <= maxChar && len >= minChar) {
        return true;
        }

        return false;
    }


    // e) Arrays
    /**
     * Prints all Strings in received array using
System.out.println.
     * One String on each line.
     */
    public void printAllStrings(String[] strings){
        for (int i = 0; i < strings.length; i++) {
            System.out.println(strings[i]);
        }
    }


    /**
     * Returns the sum of all numbers in received array.
     */
    public int arraySum(int[] numbers){
        int sum = 0;
        for (int i = 0; i < numbers.length; i++) {
            sum += numbers[i];
        }
        return sum;

    }

    /**
     * Prints all Strings in received array using
System.out.println.
     * One String on each line.
     * But only if the String is not exactly "Corona".
     */
    public void printAllStringsNotCorona(String[] strings){
        for (int i = 0; i < strings.length; i++) {
            if (!strings[i].equals("Corona")) {
                System.out.println(strings[i]);
            }
        }
    }

    // f) Collections
    /**
     * Finds all integers lower than a given number and stores these
in an ArrayList
     */
    public ArrayList<Integer> findAllIntsBelowNumberInArray(int[]
integerArray, int number) {
        ArrayList<Integer> store = new ArrayList<>();
        for (int i = 0; i < integerArray.length; i++) {
        int lower = integerArray[i];
         store.add(lower);}

        // for (int lower : integerArray) {
        //     if (lower < number) store.add(lower);
        // }
        return store;
    }

    /**
     * Inputs two arrays and maps the elements in the keyArrays to
```

```
the elements in the valueArrau
     * in a hashmap and returns this HashMap
     */
    public HashMap<String, String> makeHashMapFromTwoArrays(String[]
keyArray, String[] valueArray) {
         HashMap<String,String> map = new HashMap<>();

        int number = Math.min(keyArray.length, valueArray.length);
        for (int i = 0; i < number; i++) {
            map.put(keyArray[i], valueArray[i]);
        }
        return map;
    }

    /**
     * Find the frequency of occurences of each element in the
parameter array (stringArray)
     * and store the frequency of each element in the array as a key
value pair in a HashMap
     * with the element as key and frequency as value
     */

    public HashMap<String, Integer>
findFrequencyOfElementsInArrayListOfStrings(ArrayList<String>
stringList) {
      HashMap<String,Integer> freq = new HashMap<>();
        for (int i = 0; i < stringList.size(); i++) {
        String current = stringList.get(i);

        if (freq.containsKey(current)) {
            int oldValue = freq.get(current);
            freq.put(current, oldValue + 1);
        } else {
            freq.put(current, 1);
        }
    }
    return freq;
    }


    // The following methods are Optional assignments:
    // additional optional assignments might be added later.


    /**
     * Returns the index of the first occurrence of char c in String
string.
     * Returns -1 if char is not found.
     * Tips: google er din venn
     */
    public int firstOccurrence(String string, char c){
        return 0;
    }
    /**
     * Returns the string with out starting spaces only a single
trailing space at the end
     * hint: https://www.w3schools.com/java/ref_string_trim.asp
     */
    public String ensureOnlySingleSpaceAtEndOfString(String string){
        return string;
    }

    /**
     * Return True if the string is valid under the following
conditions:
     * Only single trailing spaces
     * No starting spaces
     * Must be longer or equal to 6 characters
     * Must be shorter or equal to 60 characters
     * Hint: Maybe its possible to reuse previous methods for this
task?
     */
    public Boolean validateString(String string){
        return false;
    }


    /**
     * Prints the provided strings in upper case letters.
     * One String on each line.
     */
    public void printUpperCaseStrings(String[] strings){

    }

    /**
     * Print all characters until a char is "."
     * including the .
     * Do not print in separate lines.
     */
    public void printFirstSentence(char[] chars) {

    }

    /**
     * Prints all Strings in received array to standard output.
     * One String on each line.
     * But only if the String is not Corona (case insensitive).
     */
    public void printAllStringsNotCoronaCaseInsensitive(String[]
strings){

    }

    /**
     * Returns the sum of all the received numbers.
     * hint: this is called varargs
     */
    public int addNumbers(int... numbers){
        return 0;
    }
}
```