

Panel 1

Task: Remove leading spaces from a string

Input: a string to check + remove leading spaces

Output: String with no leading space

"-- Bert Wachsmuth --"

Algorithm:

Let pos be the position of first non-space character

Then return substring from pos to the end.

1

Panel 2

Task: Remove leading spaces from a string

Input: string

Output: string

name clean

"BERT"

Header

```
public static String clean(String s)
```

Algorithm:

Let pos be the position of first non-space character.

Then return substring from pos to the end.

```
{
    int pos = posOfNonSpace(s);
    String answer = "";
    for (int i = pos; i < s.length(); i++)
        answer += s.charAt(i);
    return answer;
}
```

2

Panel 3

Task: Remove leading spaces from a string

```
public static String clean(String s)
{
    int pos = posOfNonSpace(s);
    return s.substring(pos);
}
```

built-in method to
return substring from
pos to end

Task: Find first non-space character " _ _ _ "

Algorithm:

Start looking
through characters
as long as they
are spaces.
Stop when non-
space and return
position

```
public static int posOfNonSpace(String s)
{
    → int pos = 0;
    while ((s.charAt(pos) == ' ') && pos < s.length())
    {
        pos++;
    }
    return pos;
}
```

3

Panel 4

```
public class StringCleaner
{
    // Returns position of first non-empty character in string s
    public static int posOfNonSpace(String s)
    {
        int pos = 0;
        while ((pos < s.length()) && (s.charAt(pos) == ' '))
        {
            pos++;
        }
        return pos;
    }

    // Removes all leading spaces from string s
    public static String clean(String s)
    {
        int pos = posOfNonSpace(s);
        return s.substring(pos);
    }

    // Tests the 'clean' method in various situations
    public static void main(String args[])
    {
        String s1 = "Bert Wachsmuth";
        String s2 = " Bert Wachsmuth ";
        String s3 = "";
        String s4 = " ";
        System.out.println("[ " + clean(s1) + " ]");
        System.out.println("[ " + clean(s2) + " ]");
        System.out.println("[ " + clean(s3) + " ]");
        System.out.println("[ " + clean(s4) + " ]");
    }
}
```

4

Panel 5

Task: Write a method that shifts a character up or down by a certain amount.

'z' + 1 = 'a'
 | c -

Input: char and amount

Output: char (shifted)

Algorithm:

- convert char to int
- add amount to int ✓
- if new int is too large subtract (high - low + 1)
- if new int is too small add (high - low + 1)
- convert back to char

```
public static shiftChar(char c, int amt)
{
    int code = (int)c;
    code += amt;
    if (code > high)
        code -= (high - low + 1);
    if (code < low)
        code += (high - low + 1);
    return (char)code;
}
```

field

5

Panel 6

```
public class Shiftter = 122;
{
    public static int high = (int)'z';
    public static int low = (int)'A';
    public static char shiftChar(char c, int amt)
    {
        // as before
    }
    public void main()
    {
        // one more method
        // shiftString
    }
}
```

6

Panel 7

Substitution Cypher

⇒ String 1: "A B C D E U V W X Y Z" (LOCK)

⇒ String 2: "Z Y X W V A B C D E F" (KEY)

to encode "DUDE" →

↓

WAWV

↓

DUDE

7

Panel 8

Subst. cypher method:

Input: String to code, String 1 and String 2

Output: String (coded)

Algorithm:

- take ^{ith} first char. -
- let pos be the pos of that char. in string 1 (lock)
- find char at pos in string 2 (key)
- append that char to output string

repeat, then return output string

```
public static String subst (String s,
                          String lock, String key)
{
    String answer = "";
    for (int i = 0; i < s.length(); i++)
    {
        char c = s.charAt(i);
        int pos = posChar(lock, c);
        char code = key.charAt(pos);
        answer += code;
    }
    return answer;
}
```

8 } return answer.