UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

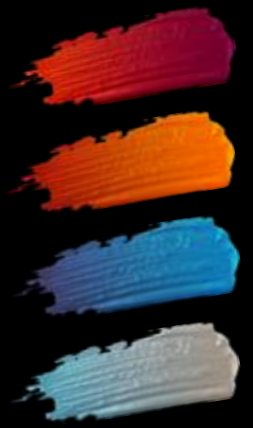
SERLAB
Software Engineering Research

Integrazione e Test di Sistemi Software

Left Pad exercise

Azzurra Ragone

Dipartimento di Informatica - Università degli Studi di Bari
Via Orabona, 4 - 70125 - Bari
Tel: +39.080.5443270 | Fax: +39.080.5442536
serlab.di.uniba.it



Specification based + structural testing: an example

REQS:

Left-pad a string with a specified string. Pad to a size of size.

- `str`—The string to pad out; may be null.
- `size`—The size to pad to.
- `padStr`—The string to pad with. Null or empty is treated as a single space.

The method returns a **left-padded string**, the original string if no padding is necessary, or null if a null string is input.

EX.

- `str = 'abc'`
- `size = 5`
- `padStr = 'X'`

RESULT: 'XXabc'

LeftPad(): implementation

```
public static String leftPad(final String str, final int size,
  String padStr) {

  if (str == null) {
    return null;
  }

  if (padStr==null || padStr.isEmpty()) {
    padStr = SPACE;
  }

  final int padLen = padStr.length();
  final int strLen = str.length();
  final int pads = size - strLen;

  if (pads <= 0) {
    // returns original String when possible
    return str;
  }

  if (pads == padLen) {
    return padStr.concat(str);
  } else if (pads < padLen) {
    return padStr.substring(0, pads).concat(str);
```

If the string to pad is null, we return null right away.

If the pad string is null or empty, we make it a space.

There is no need to pad this string.

If the number of characters to pad matches the size of the pad string, we concatenate it.

If we cannot fit the entire pad string, we add only the part that fits.

LeftPad(): implementation

```
public static String leftPad(final String str, final int size,
  String padStr) {

  if (str == null) {
    return null;
  }

  if (padStr==null || padStr.isEmpty()) {
    padStr = SPACE;
  }

  final int padLen = padStr.length();
  final int strLen = str.length();
  final int pads = size - strLen;

  if (pads <= 0) {
    // returns original String when possible
    return str;
  }

  if (pads == padLen) {
    return padStr.concat(str);
  } else if (pads < padLen) {
    return padStr.substring(0, pads).concat(str);
```

If the string to pad is null, we return null right away.

There is no need to pad this string.

If the number of characters to pad matches the size of the pad string, we concatenate it.

If we cannot fit the entire pad string, we add only the part that fits.

Ex:

```
str = 'abc'
size = 2
padStr = 'X'
pads = 2-3<= 0
RESULT = 'abc'
```

LeftPad(): implementation

```
public static String leftPad(final String str, final int size,
  String padStr) {

  if (str == null) {
    return null;
  }

  if (padStr==null || padStr.isEmpty()) {
    padStr = SPACE;
  }

  final int padLen = padStr.length();
  final int strLen = str.length();
  final int pads = size - strLen;

  if (pads <= 0) {
    // returns original String when po
    return str;
  }

  if (pads == padLen) {
    return padStr.concat(str);
  } else if (pads < padLen) {
    return padStr.substring(0, pads).concat(str);
```

If the string to pad is null, we return null right away.

If the pad string is

Ex:

```
str = 'abc'
size = 4
padStr = 'X'
pads = 4-3 = 1
padLen = 1
RESULT = 'Xabc'
```

s to
he
pad string, we concatenate it.

If we cannot fit the entire pad string, we add only the part that fits.

LeftPad(): implementation

```
public static String leftPad(final String str, final int size,
  String padStr) {

  if (str == null) {
    return null;
  }

  if (padStr==null || padStr.isEmpty()) {
    padStr = SPACE;
  }

  final int padLen = padStr.length();
  final int strLen = str.length();
  final int pads = size - strLen;

  if (pads <= 0) {
    // returns original String when p
    return str;
  }

  if (pads == padLen) {
    return padStr.concat(str);
  } else if (pads < padLen) {
    return padStr.substring(0, pads).concat(str);
```

If the string to pad is null, we return null right away.

If the pad string is null or empty, we make it a space.

Ex:

```
str = 'abc'
size = 4
padStr = 'XXX'
pads = 4-3 = 1
padLen = 3
RESULT = 'Xabc'
```

...ers to ...he ...te it.

If we cannot fit the entire pad string, we add only the part that fits.

LeftPad(): implementation

```
  } else {
    final char[] padding = new char[pads];
    final char[] padChars = padStr.toCharArray();

    for (int i = 0; i < pads; i++) {
      padding[i] = padChars[i % padLen];
    }

    return new String(padding).concat(str);
  }
}
```

We have to add the pad string more than once. We go character by character until the string is fully padded.

Ex: pads>padLen

```
str = 'abc'
size = 7
padStr = 'XY'
pads = 7-3 = 4
padLen = 2
RESULT = 'XYXYabc'
```


Let’s start with Specification-based testing

leftPad()

Three inputs:

```
str = ‘abc’
size = 5
padStr = ‘X’
```

One output: `String`

Identify the partitions and the test suite (20 mins)

Azzurra Ragone

Department of Computer Science- Floor VI – Room 616
Email: azzurra.ragone@uniba.it