DAT101 Notes

String magic

Create an array of a string

```
StringBuilder tekstarr = new StringBuilder(tekst);
```

Set a char at index i in array

```
tekstarr.setCharAt(i, 'y');
```

Return array as string

```
return tekstarr.toString();
```

Testing

Setup function, creating from original GUI

```
@Before
public void setUp(){
    gui = new GUI();
}
```

Test function, use assertEquals to test if a given value returns the correct value

```
@Test
public void testAlgoritme2() {
    // Checks if gui.algoritme2 returns åøe when given øæy
    assertEquals("åøæ", gui.algoritme2("øæy"));
}
```

For numbers, pass a margin of error (delta) as the third argument to assertEquals

```
@Test
public void testKonverter2() throws Exception {
    assertEquals(319.34, Oblig16.konverter(5000, 2, 0), 0.01);
}
```

Proxy

A kind of memory buffer, saves bandwidth by saving things in a cache.

HashMap is a list of unique elements

```
    private HashMap<String,Image> p = new HashMap<String,Image>();
```

Downloading an image from the Internet

```
    public void downloadImage(String url) {
    // Use a try/catch in case the URL is invalid
        try {
            realurl = new URL(url);

            // Read an image from the URL
            Image img = ImageIO.read(realurl);

            // Put the image in the list
            p.put(url, img);

            getImage(url);
        } catch (MalformedURLException e) {
            // Runs if the URL is invalid
            e.printStackTrace();
        } catch (IOException e) {
            // Runs on an I/O error (error connecting to the Internet)
            e.printStackTrace();
        }
    }
```

Use a buffer to save bandwith, check if the image is already in the HashMap

```
    public Image getImage(String url){

        // Check if the image is already in the HashMap, download and add if not
        if(p.containsKey(url) == false) {
            downloadImage(url);
        }

        // Get image from HashMap, return it
        Image img = this.p.get(url);
        return img;
    }
```

Kakeoppskrift

For-each loop

```java
for (Object s : noe) {
    m.addElement(s);
}
```

Collections

ArrayList - Like an array but dynamic size, add/remove elements at will

```java
ArrayList<Integer> a = new ArrayList<Integer>();

// Add/remove elements
a.add(14)
a.remove(0) // Use with index
```

Sets are lists that only have unique elements

HashSet
Elements are unique

```java
HashSet<Integer> s = new HashSet<Integer>();
```

TreeSet
Elements are unique and sorted automatically

```java
TreeSet<Integer> s = new TreeSet<Integer>();
```

LinkedHashSet
Elements are unique and sorted in the same order they were added

```java
LinkedHashSet<Integer> s = new LinkedHashSet<Integer>();
```

HashMap
A list that has a key and a corresponding value (like a dictionary in Python)

```java
HashMap<String, String> h = new HashMap<String, String>();

// Add element using put
h.put("key", "value");
h.remove("key");
```

Update GUI with lists

```
// Pass the JList that is used in the GUI and the ArrayList you want to make a
// list from
public void updateGUI(JList l, Object[] a) {
    DefaultListModel m = new DefaultListModel();
    for (Object s : a) {
        // Copy contents of ArrayList into the list model
        m.addElement(s)
    }

    // Update the JList with our new model
    l.setModel(m);
}
```