Getting started guide
Importing framework (IntelliJ guide)
- Create new Java IntelliJ project.
- Then select File -> Project Structure
- Select Modules,
- Select Dependencies
- Click the + symbol  or  press (alt + insert) -> JARs or Directories
-Select the Jar file

## Using the Framework

The framework builds on the concept of having a series of objects that represents HTML tags. Some tags depends on each other tags, and we will use everything to create a page/HTML file, and using the builder pattern to create CSS brackets that are added together in CSS to create a CSS file

HEADER
To use the framework to create a HTML page, the first thing is to create a new header object using the generate method in this class. This needs to contain a title, optional is a navigation list in the form of a nav object and or just an optional paragraph. The generator will detect and create the object based on what we give

Example of just title, and with a paragraph

Examples:
```
Header header1= Header.generate("This is title");
Header header2= Header.generate("This is title", genericParagraph);
```

Example with just nav and nav and paragraph, we will show how to generate nav and paragraph under:

```
Header header3= Header.generate("This is title", nav);
Header header4= Header.generate("This is title",  genericParagraph, nav);
```

NAVIGATION
If we want with a navigation we first need to create a nav object, to do this we need 2 arrays of strings, one with the link URL and the other for the link text:
```
private static String[] linksList = new String[]{"www.google.com",
"www.amazon.com", "www.pottermore.com"};

private static String[] linksTextList = new String[]{"google", "amazon",
"pottermore"};
```

We then generate a navigation with the nav.generate method, including the link list, the link text list,  what the ID identifier and the class identifier will be

```
Nav nav = Nav.generate(linksList, linksTextList, "mainNav",
       "navClass");
```

MAINTAG

After we have created a header we need to create a main tag object. This can either be created from a string, an array of strings or a section, or an array of sections. We will once again use the generate method on this class. Example of how to create section will be below

Example of array of string

```
String[] arrayString = new String[]{"<div> <p>manuall insertion </p>
</div>",
       "<div> <p>lorem ipsum insertion </p> </div>" };
```

Example of array of Sections

```
Section[] sectionsArray = new Section[]{section1, section2};
```

Examples of main tag:

```
MainTag mainTag1 = MainTag.generate("<div> <p>manuall insertion </p>
</div>");
MainTag mainTag2 = MainTag.generate(arrayString);
MainTag mainTag3 = MainTag.generate(section);
MainTag mainTag4 = MainTag.generate(sectionsArray);
```

ARTICLES

To create a section we need an array of articles, so we will be looking at creating an article first.

We have have different kinds of articles to create they contain, examples on forms will be further down

Title paragraphString, article ID, group identifier
Title Paragraph, ArtcileID, group identifier
Title  paragraph, form, article ID, class identifier
title, paragraph string, form, article ID, group identifier

Example of making the articles

```
Article article1 = Article.generate("Article", "<p> stuff </p>",
       "article1",  "articleclass");

Article article2 = Article.generate("Article", genericParagraph,
       "article2",  "articleclass");

Article article3 = Article.generate("Article", genericParagraph, form,
       "article3",  "articleclass");

Article article4 = Article.generate("Article", "<p> stuff </p>", form,
```

```
        "article4",  "articleclass");
```

Back to section.

We take the articles we generated and make them into an array

```
Article[] articlesArray = new Article[] {article1, article2, article3,
article4};
```

then we use the generate method in section

```
Section section1 = Section.generate( articlesArray, "section",
"sectionClass");
Section section2 = Section.generate( articlesArray, "section",
"sectionClass");
```

FOOTER
We then need a footer
The kinds of footer we can create is footer with
Email
email paragraph
email paragraphString
email paragraph1, paragraph2
email paragraphString1, paragraphString2

examples

```
Footer footer1 = Footer.generate("anderscg@hiof.no");
Footer footer2 = Footer.generate("anderscg@hiof.no","<p>Hello</p>");
Footer footer3 = Footer.generate("anderscg@hiof.no", genericParagraph);
Footer footer4 = Footer.generate("anderscg@hiof.no", "<p>Hello</p>",
"<p>Goodbye</p>");
Footer f5 = Footer.generate("anderscg@hiof.no", genericParagraph,
genericParagraph);
```

Now that we have our main tag we can create a page with the header, the main tag and the footer

```
Page newPage = Page.generate(header1, mainTag2, footer1 );
```

Once we have generated it we can generate a file with the file name  and the HTML file will be generated

```
Page.generateFile("HTMLFile");
```

OTHER STUFF
FORM
To generate a form we need an array of form options so we will look at how we make a form option.

FORM OPTION

To generate a form we use the generate method with Description, type of input, form option ID and form option class

```
FormOption formOption1 = FormOption.generate("Birthday", "date",
       "q1", "FormOptionclass");
FormOption formOption2 = FormOption.generate("mothers name", "text",
       "q2", "FormOptionclass");
FormOption formOption3 = FormOption.generate("Phone number ", "number",
       "q3", "FormOptionclass");
```

We then use these in an array

```
FormOption[] formOptionArray = new FormOption[]{formOption1, formOption2,
formOption3};
```

We then generate a form using the generate method

```
Form form = Form.generate("Form", "FormID", formOptionArray);
```

IMAGE

To generate an image you  give the image URL/path a description for the picture, id and class

```
Image image =
       Image.generate("*Image url*", "picuter description", "img1", "img");
```

PARAGRAPH

If we want a Paragraph we first need to create a paragraph with the Paragraph.generate method. We have to write what the paragraph will contain, and an ID and class identifier

```
Paragraph genericParagraph =
       Paragraph.generate
               ("This is an generic paragraph", "genPar", "paragraphs");
```

CSS

To use CSS we need to have an array brackets

CSS bracket to use a css bracket you need to use CSSBracket.Builder.NewInstance() with the target name, it can be a class, an ID or a HTML tag, the you have to specify what type it is (you can use.getId/getClass from the objects . After you add the methods that generate the the values you want, then add .build

```
CSSBracket holder1 =
       CSSBracket.Builder.newInstance(nav.getID, "ID")
       .addMargin(5f, "px")
       .addUnderline()
       .addBackgroundColor("Red")
       .addTextColor("blue")
       .build();
```

we then generate an array of CSSbrackets then we generate the CSS with it

```
CSSBracket[] cssBracketsArray= new CSSBracket[]{bracket1,bracket3};
CSS css1 = CSS.generate(cssBracketsArray);
```