Login and Registration services

Backend will use `PasswordKeeper` to manage login credentials. Since this is a RESTful service, most of the logic will happen in implementation of the `PasswordKeeper`.

Communication with frontend

Send `POST` request to `services/login` or `services/register`. The data should be in JSON format.

Example of login:

```
POST /CodeReview/services/login HTTP/1.1
Content-Type: application/json
Content-Length: <length>

{"username": "Ashley", "password": "Massacre"}
```

Example of response to successful login

```
200 OK HTTP/1.1
Content-type: text/plain

<redirection-url>
```

On login failure status code 401 will be returned. Due to Jersey's implementation of `Response`, it won't be empty, but frontend is safe to ignore body of the response.

Example of register:

```
POST /CodeReview/services/register HTTP/1.1
Content-Type: application/json
Content-Length: <length>

{
    "username" : "Ashley"
    "password" : "Massacre"
    "age" : 23
    <other seconday key-value pairs to store in DB>
}
```

On **successful registration** status code 200 will be returned. Body of the response can be safely ignored.

On **failure to register**, status code 400 will be sent, and after that it would be great if frontend will not let users send again without modifying the credentials. Body of the response can be safely ignored.

AuthService

The class will mostly parse JSON and forward to `PasswordKeeper`. Depending on the answer from the keeper, it will build a response and send it.

UnencryptedPasswordKeeper

Basically the class is a plaintext version of the password keeper. It operates on a file.

- `constructor(String filename)`

  The constructor should initialize the `PasswordKeeper` with the contents of the file. Format of the file can be found below.

- `@OVerride boolean synchronized exists(String user, String password)`

  The method should check if the combination of `user` and `password` exists.

The method is `synchronized` because somebody might insert the `user + password` combination while the current method searches for it. Don't forget to put `@Override`.

- `@Override boolean synchronized register(String user, String password)`

  The method should add the `user + password` combination into `userToPassword` map, and immediately propagate changes into file on which it is operating, so the registered users won't be lost between server reboots. It is synchronized for reasons stated in `exists()`.

File format

The easiest format will be (in regex)

```
(\.+) : (\.+)\n
```

e.g. sequence of non-space characters followed by `" : "` (space, colon, space), followed by another sequence of non-space characters. One can use the regex to actually retrieve the `username` and `password` by using `.group(1)` for username and `.group(2)` for password.