### Task 9.1 Adapter

In this exercise, we will standardise the various animal libraries to one interface.
From the lecture repository, rename and copy `exercise/pennaLV/src/zoo/adapter_skeleton.hpp` into the corresponding directory of your animal library and implement it such that it correctly forwards to your animals.
If your animals already follow this exact interface, the adapter becomes quite simple to write.

### Task 9.2 Concept Adaptation

Update the concept used by your simulation such that it works with the interface fixed by the adapter. Update your `main.cpp` as well (e.g. `sheep` → `xxx_to_sim<sheep>`).
From this point on, you will be able to exchange your animal library with any programmer who also implemented the task correctly.

### Task 9.3 Documentation

Write a full documentation of your PennaLV program.

- Install *Doxygen* on your operating system and set up the documentation in your project directory:

```
mkdir doc && cd doc
doxygen -g
```

  Instead of editing the newly generated `Doxyfile` manually (by at least setting the `PROJECT_NAME` and `INPUT` variables), you can install *doxygen-gui* (*doxywizard*).

- In each source file, format the header comment in a Doxygen-conforming (`/**`) way:

```
/**
 * \file
 * \brief    short description
 * \author   your name
 * \details  detailed description (delete this line if none)
 */
```

- Right above each struct/class, **public** method/member and function, write a documentation:

```
/**
 * \brief     short description
 * \params    input parameters (delete this line if none)
 * \return    return value (delete this line if none)
 * \details   detailed description, hints, examples (delete if none)
 */
```

  e.g.:

```
/** \brief Species properties holder */
static species_properties prop;
```

- Each time you want to update (by default) a html and latex documentation, just enter your `doc` directory and check `html/index.html` after running:

```
doxygen
```

### Task 9.4 Thunder-Struct Challenge (optional)

In this week's challenge you may or may not learn much about C++, but you will be having fun.
The task is to submit a `struct` with exactly a move and a name function. It will be placed on a hexagonal field together with another (enemy) `struct`, and they will be alternatingly allowed to move by one field. The goal is to beat your nemesis by moving onto its field. If this hasn't happened after a maximum number of iterations, the game ends in a draw. After battling all other `structs` one-on-one, the most successful `struct` gets the most chocolate.
We favor hunting over surviving: a win gets 3 points, a draw only 1 and a loss 0 points, which yields a normalised score $\in [0, 1000]$.
You `struct` can only see adjacent fields and must not (!) have a state (no static variables in methods / no access to global variables / no members / ...). Make your move decision only dependent on the arguments the move function receives. Do not abuse the random number engine to somehow store a state. If something is unclear, ask us on the mailing list. You can submit up to two bots per student. It's perfectly fine to test your bots in advance against your friends' bots.
gl hf :)
The submission deadline for this challenge is Wednesday 18.11.15 at 05:00 in the morning. Notify us via the mailing list (pt2_hs15_ta@lists.phys.ethz.ch) with a link to your uploaded solution on your PT2 repository on GitLab.