# Problem 2 - Peberholm notes

## The Grid

- Grid size: $n \times m$
- $n, m < 0$
- $n$ and $m$ values are found at `PeberholmCosntantsAndUtilities.H` and `W`
- Plants take up one grid space
- Plants can only be located in grid positions

## Plants

- Four kinds of plants:
  - trees
  - bushes
  - flowers
  - mosses
- Plants differ in:
  - how they look
  - how they spread seeds
  - $s$ and $r$ values
- $s$, $r$ and color values are stored in `PeberholmConstantsAndUtilities.java`
  - $s$ value for bush object: `PeberholmConstantsAndUtilities.BUSH_SEED_NO`
  - $r$ value for bush object: `PeberholmConstantsAndUtilities.BUSH_RANGE`
  - color value for bush object: `PeberholmConstantsAndUtilities.BUSH_COLOR`

## Making new plants

In one step, one plant generates $s$ new plants.

New plants are generated by adding random numbers between [-r;r] to the coordinates of the mother plant.

No plants can be generated in water. They die immediately.

The “seeds” that *do not* land in water, immediately become full-size $1 \times 1$ plants.

### `public Plant[] spreadSeeds()`

Generating new plants should be handled in the plant’s `spreadSeeds()` method.

`spreadSeeds()` is called from `PeberhomSimulation.java`

`spreadSeeds()` returns an array of new plant objects. The new plant objects are added to the `ArrayList newPlants` in `PeberhomSimulation.java`

### `PeberholmSimulation.java`

This class runs the simulation, including all the graphics.

`PeberholmSimulation.java` handles planting and drawing in steps.

This class cannot be edited.

### `PeberholmConstantsAndUtilities.java`

Constants:

- Grid ("island") size
- Total nr. of turns
- $s$, $r$, and color of plants
- `boolean SHOW`: controls whether graphics are on

Utilities (not v. useful stuff):

- A method to check weather a `Point` is on the "island".
- A method to check weather coordinates `int x, int y` are on the "island".
- A method to get a random position on the "island".
- A random int generator method.

This class cannot be edited.

### `PeberholmDriver.java`

Contains the main method, just starts the simulation.

Also contains some weird shit about a mysterious Checker class.

This class cannot be edited.

### `Plant.java`

Is the parent class to the different plant sub-classes.

This class cannot be edited.