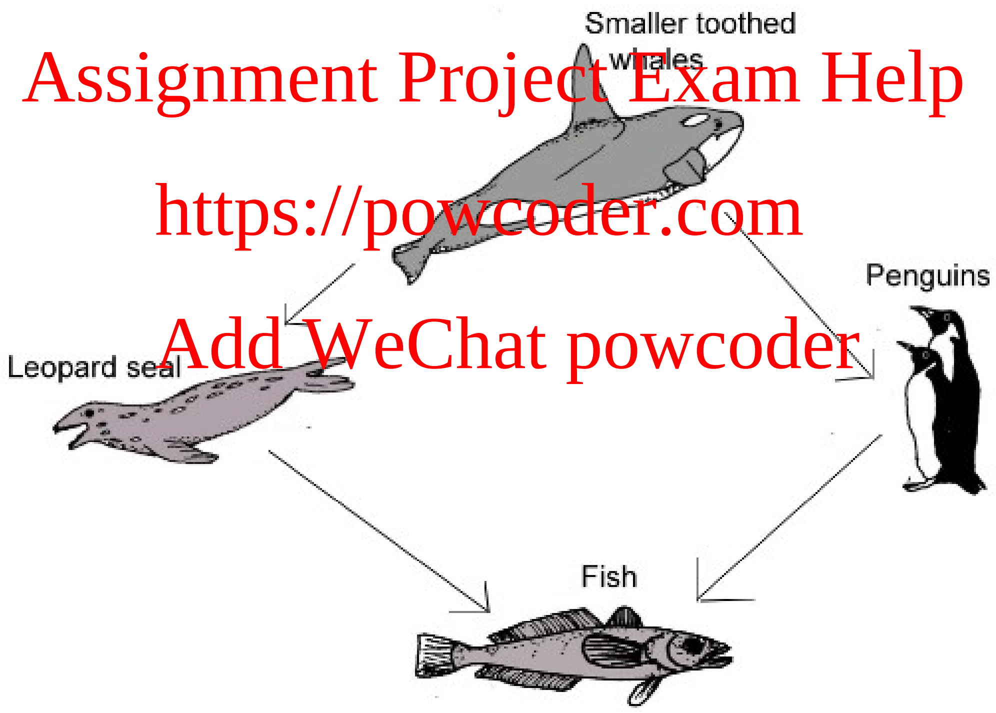
Penguin game in Antarctica

They already simulated the population of penguins at the beginning of the semester. Now you will go one step further and simulate a small (very simplified) section of the Antarctic ecosystem. We look at a square, tiled section of the Antarktis world with a fixed side length of 41. In Antarktis, the tiles are represented by a two-dimensional array of Animals, with a null value for an empty field. The world is cyclic, i.e. if an animal runs out of the world to the left, it is placed on the field furthest to the right of the corresponding line (and vice versa). The same applies to the top and bottom.
The coordinate system of the world behaves in such a way that tiles with y-value 0 are at the top, and higher y-values at the bottom.



Animals are instances of a subclass of the abstract class Animal. They have an alive attribute that ensures that they are drawn. Dead animals are not drawn.

For each animal a method public boolean canEat(Animal animal) is already implemented in the template, which returns whether this animal can eat any other animal, according to the hierarchy shown above (arrows indicate that this species eats another animal, e.g. the whale eats penguins). However, this method still requires implementations of the following methods in each of Animal's subclasses:

```
protected abstract boolean eatenBy(Penguin penguin);
protected abstract boolean eatenBy(PlayerPenguin
playerPenguin);
protected abstract boolean eatenBy(Whale whale);
protected abstract boolean eatenBy(LeopardSeal
leopardSeal);
protected abstract boolean eatenBy(Fish fish);
```

Whereby these methods do NOT check whether the animals are on neighbouring fields, but only whether it comes according to the hierarchy to the food.

Feeding behaviour

Movement of animals

Depending on the species, the animals have similar movement patterns. Basically, the animals have the following movement preferences:

- Fish: 1. up, 2. right, 3. down, 4. left
- All other species (Penguin, LeopardSeal, Whale): 1. left, 2. up, 3. right, 4. down

First the animals check the adjacent fields in the order indicated whether there is food there. If this is the case, they move to the first such field for which the move is allowed. If you cannot move to any field with food, move to the first allowed field (also in the order given).

Moves to a field with another animal that cannot be eaten by the moving animal are not allowed. Animals also never move to a field where a hunter borders in one of the four directions. If there is no allowed move for an animal, it does not move.

There is also a PlayerPenguin class, which is a subclass of Penguin, but controlled by the player.

Movements are made in turns, with the player always moving first and then in the following order:

1. Whale
2. LeopardSeal
3. Penguin

4. Fish

Animals of the same species are moved in the order of their occurrence in the respective array. Eaten animals lose their lives and disappear from the tile. Dead animals no longer move.

End of the game

The goal of the player is to move to the field of the other penguin. The player wins the moment he moves to the square where the other penguin is. The move of the other penguin or all other animals is then no longer performed.
The player loses if he moves to a field with a soul leopard or a whale. Furthermore, the player loses if the player penguin or the other penguin is eaten.

Implementation of the course of the game

1. Implement the method public void move() in the classes Animal and Fish.
   Movements (example)
2. Implement the method public boolean move(int newX, int newY) in the PlayerPenguin class. Your method sets the player to the corresponding position. It also returns true if the game is over at the moment you win or lose. There is no need to check whether the player is teleporting, i.e. whether the new position is really only one step away.
   Player Movement (Example 1) No results
   Player Movement (Example 2) No results
3. Implement the method public void moveAll() in the class Antarktis. The method should:
   - Execute the movements using the implemented move methods in the order described above.
   - The corresponding animals should be eaten during the movement.
4. Complete the game loop in the method public void gameLoop() in the class Antarktis. The loop should run as long as the game is not over. After drawing the playing field with draw(), the playerPenguin should move one step in the direction the user wishes. To do this, access the currentEvent attribute in the method by either NOTHING if no key was pressed or UP, DOWN, LEFT or RIGHT if the user pressed one of the arrow keys. If a key was pressed, the player penguin should be moved to the corresponding new position. If this is not the end of the game, the movements of the other animals are carried out after the player's move. Set the value of

currenEvent back to NOTHING at the end of each iteration. (This method must not be recursive under any circumstances, otherwise it can lead to stack overflows during very long games!)

Hints:

- The Antarctic class already contains some static fields and a setup method to help you test your implementation.
- You may extend the given classes by any (auxiliary) methods, but the given methods or methods specified in the task must be implemented accordingly.
- *The keyword instanceof of Java must not be used.*