AARHUS UNIVERSITET

Software Engineering and Architecture

”Rye bread Algorithms”

AARHUS UNIVERSITET

Motivation

- SWEA is an **architecture and engineering** course
  - Functionality = “The required work done by the program”
  - *Functionality can be made with any number of architectures!*

- *Thus this implies that*
  - *SWEA* evaluation is less focused on correct functionality

- **But…**
  - ***Embarrassing if you code computes 2+2 to be 5, right?***

AARHUS UNIVERSITET

Motivation

- In previous years, I found that too many students writes very long, cumbersome code which … **computes incorrectly!**

- Even for trivial algorithms like those in HotCiv
  - *Increase treasury in all cities when round ends*
  - *If treasury > cost(unit) then produce a unit, in all cities*
  - *Place produced unit on first empty tile around the city*
  - *Reset move counter in all units in the world when round ends*
  - *Grow every city in the world (EtaCiv)*

AARHUS UNIVERSITET

”Algorithms”

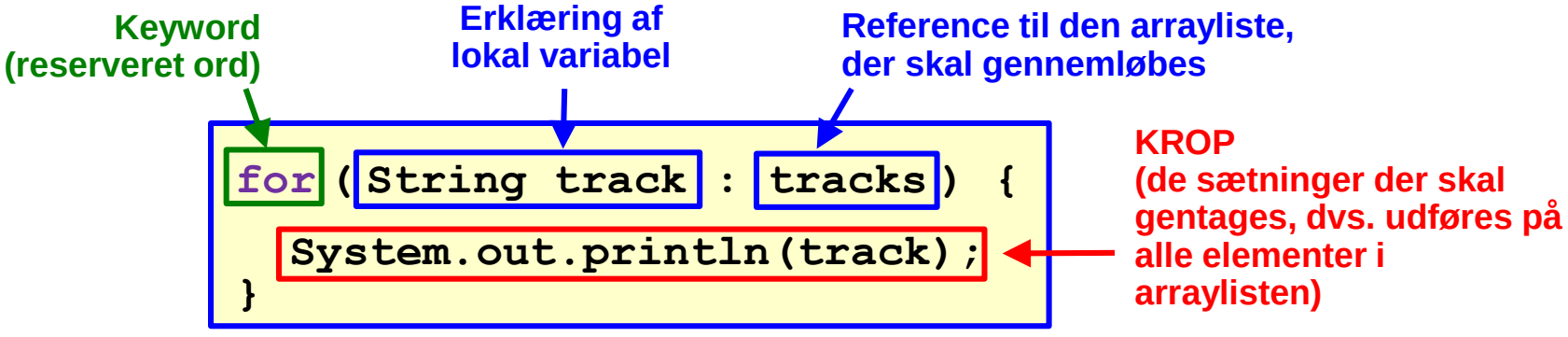
- Talking to Kasper/Gerth, they will not even call this ‘algorithms’ ☺
  - But it is. And be prepared – 90% of all industry algorithms is of this variant ☺
    - [Note: figure used here is ”qualified guess work” on my part ☺]

- [Barnes/Kölling 6th Ed, §4.9.1] **Iterations**
  - **forEach(element in collection) { doSomething(element); }**

AARHUS UNIVERSITET

From IntProg

- [Kurt Jensen: Slides-Uge3-Mandag / E2017]
  - https://users-cs.au.dk/dintprog/e17/uge_3a/



Sweep

- [Barnes&Kölling §5.3.1]

```
for each element, e, in collection:
  process e;
end.
```

```
for each element, e, in collection:
  if (e fullfills criteria c) { process e;}
end.
```

AARHUS UNIVERSITET

The Pattern / Template

- The **Sweep** template is universal

```
for each element, e, in collection:
  process e;
end.
```

```
for each element, e, in collection:
  if (e fullfills criteria c) { process e;}
end.
```

- But at the code level, differs pending on collection type

Exercise

- *Increase treasury in all cities when round ends…*

```
for each element, e, in collection:
  process e;
end.
```

```
for each element, e, in collection:
  if (e fullfills criteria c) { process e;}
end.
```

- Which one? What is e? What is collection?

Exercise

- *If treasury > cost(unit) then produce a unit, in all cities*

```
for each element, e, in collection:
  process e;
end.
```

```
for each element, e, in collection:
  if (e fullfills criteria c) { process e;}
end.
```

- Which one? What is e? What is collection? What is c?

AARHUS UNIVERSITET

Exercise

- *Place produced unit on first empty tile around the city*

```
for each element, e, in collection:
  process e;
end.
```

```
for each element, e, in collection:
  if (e fullfills criteria c) { process e;}
end.
```

- Which one? What is e? What is collection? What is c?

AARHUS UNIVERSITET

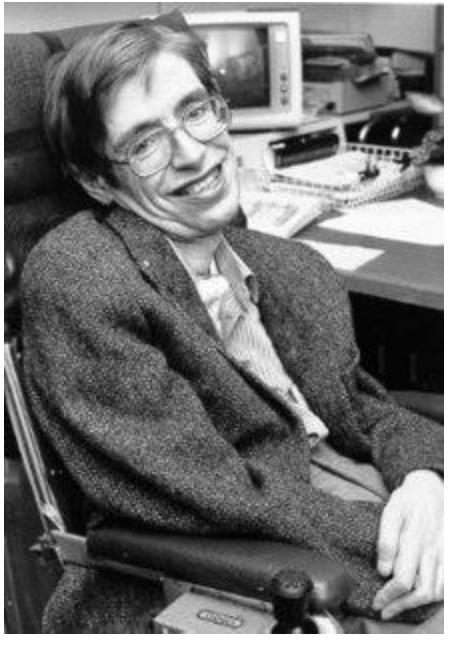
Hit the Metal

AARHUS UNIVERSITET

SWEA in a Nutshell

- Templates, like sweep, are *mental models* we use as designers!
- **But it does not help if we cannot express it in our programming language**

- Think Stephen Hawking *without* the speech-generating device



AARHUS UNIVERSITET

If we use a Matrix:

- Design decision: *City objects are stored in a matrix*
  - *Matrix[4][1] contains city object in world position (4,1)*

```
// Demonstration of the sweep template on a matrix
public void sweepMatrix() {
  System.out.println("-- Sweeping a matrix --");
  City[][] matrix = new City[WORLDSIZE][WORLDSIZE];
  matrix[1][1] = new City(); // The red city
  matrix[4][1] = new City(); // The blue city
  // Sweeping a matrix
  for (int row = 0; row < WORLDSIZE; row++) {
    for (int column = 0; column < WORLDSIZE; column++) {
      City element = matrix[row][column];
      if (element != null) {
        // process element
        System.out.println("Processing "+element);
      }
    }
  }
}
```

AARHUS UNIVERSITET

If we use a List(64):

- Design decision: ”unfold matrix to a one-dim List”
  - list.get(row*16+col) contains city at (row,col)

```
// Demonstration of the sweep template on a List
public void sweepList() {
  System.out.println("-- Sweeping a List --");
  // We create a List<City> of size 16x16,
  // and then position (3,4) is index (3*16+4)
  List<City> list = new ArrayList<City>(WORLDSIZE * WORLDSIZE);
  // Though the capacity is 64, the size is still 0 so I need to
  for (int i = 0; i < WORLDSIZE * WORLDSIZE; i++) list.add(null);
  int index;
  index = computeIndex(1,1); list.set(index, new City()); // The red city
  index = computeIndex(4,1); list.set(index, new City()); // The blue city

  // Sweeping the List
  for (City element: list) {
    if (element != null) {
      // process element
      System.out.println("Processing "+element);
    }
  }
}
```

If we use a Map<Pos, City> (1):

- Design decision: *City objects are stored in a HashMap*
  - *map.get(new Position(4,1)) contains city object in world position (4,1)*

```
// Demonstration of the sweep template on a Map
// using the Java 7 / classic for iteration
public void sweepMapClassic() {
  System.out.println("-- Sweeping a map / Classic --");
  Map<Position, City> map = new HashMap<>();
  map.put(new Position(1,1), new City()); // The red city
  map.put(new Position(4,1), new City()); // The blue city
  // Sweeping a map
  for (Position p: map.keySet()) {
    City element = map.get(p);
    if (element != null) {
      // process element
      System.out.println("Processing "+element);
    }
  }
  // Note: the 'if' is not necessary as the map only
  // contains the two cities.
}
```

If we use a Map<Pos, City> (2):

- Design decision: *City objects are stored in a HashMap*
  - *map.get(new Position(4,1)) contains city object in world position (4,1)*

```
// Demonstration of the sweep template on a Map
// using the Java 8 / stream api
public void sweepMapStream() {
  System.out.println("-- Sweeping a map / Stream --");
  Map<Position, City> map = new HashMap<>();
  map.put(new Position(1,1), new City()); // The red city
  map.put(new Position(4,1), new City()); // The blue city
  // Sweeping a map
  map.keySet()
     .stream()
     .filter(p -> map.get(p) != null)
     .forEach(p -> {
         City element = map.get(p);
         System.out.println("Processing "+element);
           });
  // Note: the 'filter' is actually not necessary here,
  // as the map only contains the two cities
}
```

Note

- You will probably never hear the term ‘sweep’ again ☺ but it important to have a term to denote a specific recurring structure…

- ‘for loop’ and ‘iteration’ are more often heard