Algorithmic Thinking and Structured Programming (in Greenfoot)

Teachers:
Renske Smetsers-Weeda
Sjaak Smetsers

Today’s Lesson plan (6)

- Looking back
  - Retrospective last lesson

- Blocks of theory and exercises
  - Variables and Operators
  - Tracing code
  - Unplugged: sorting

What we will learn today:

- Variables
- Operators:
  - Assignment: =, +=, …
  - Arithmetic: +,-,*, ++, …
  - Comparisons: <, ==, …
- Tracing code

Objects *know* stuff, too

- An object *knows*/*remembers things* (properties or state)

homeHill

Ant
homeHill carryingFood
act( ) haveFood() headHome() smellPheromone()

Variables

- When executed, programs need to store information.
  - Examples: user input, calculated values, object states, etc.
- This information can vary: we use the term variable to describe an element of a program which stores information.
- *Variables* contain data such as numbers, booleans, letters, texts, …
  - Think of them as places to store data.
  - They are implemented as memory locations.
- The data stored by a variable is called its *value.*
  - The value is stored in the memory location.

```
int nrEggsFound = 0;
```

A variable of type **`int`** with name `nrEggsFound`

Variables (2)

Pronounced as ‘becomes’

- Its value can be changed.
  - This done in an assignment statement:

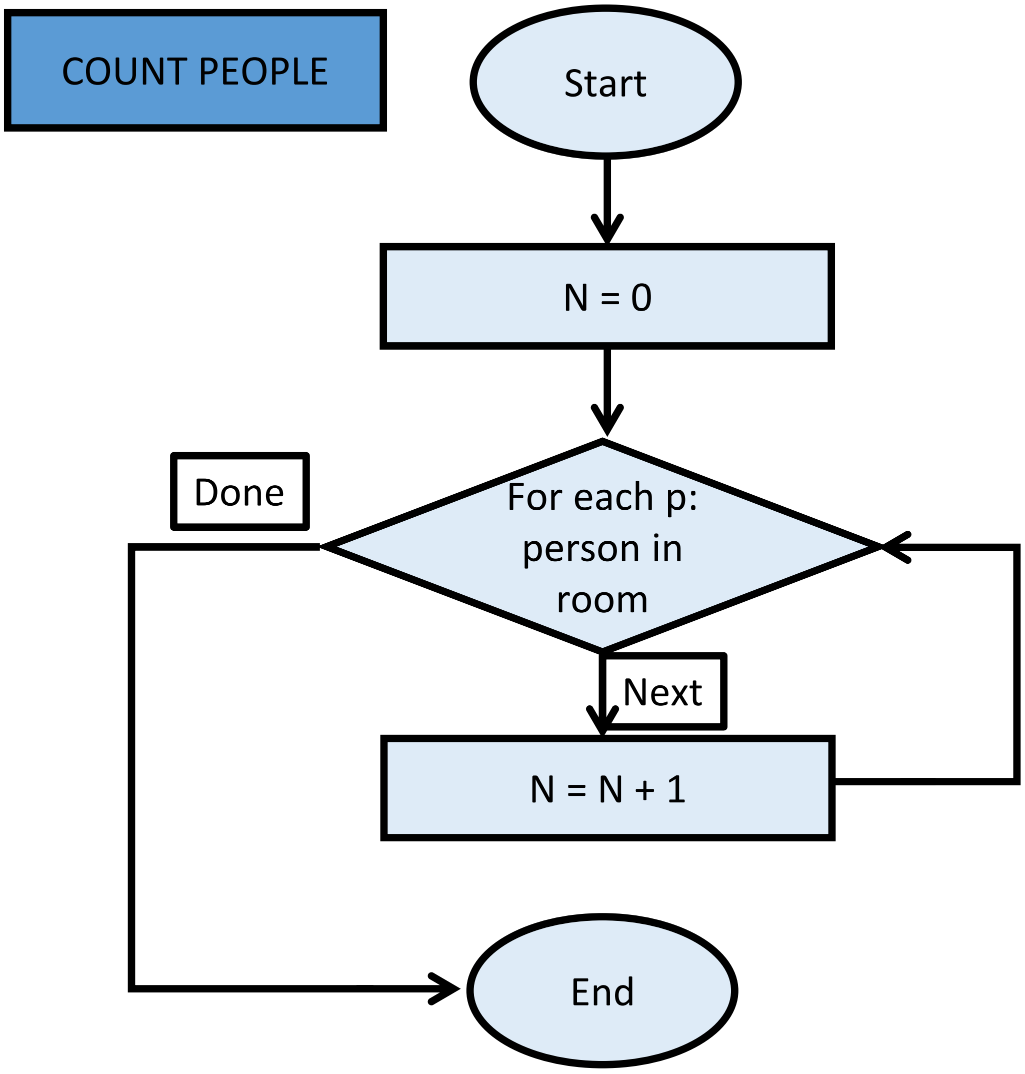
```
nrEggsFound = 15;
```

- Two kinds of variables:
  1. Local variables
  2. Instance variables

Variables (3)

Counting using a variable
For-loop

Film (20:00-25:00)



Naming and Declaring Variables

indicate, announce

- Choose names that are helpful such as **`count`** or **`speed`**, but not **`c`** or **`s`**.
- When you declare a variable, you provide its name and type.

```
int numberOfBaskets;
int eggsPerBasket;
```

- A variable's type determines what kinds of values it can hold (**`int`**, **`double`**, **`char`**, etc.).
- Any variable must be declared before it is used.

Examples

- Examples

```
int numberOfEggs, nrOfStepsTaken;
double average;
char pressedKey;
```

Film (until 1:30)

Assigning and Changing a Value

We can change the value of a variable as often as we wish. To assign a value, use:

```
variableName = some expression;
```

variable *assign to* *expression*

Memory

```
wormsEaten = 0;
wormsEaten = wormsEaten + 1;
```

0
1

Variables and Values

- Variables

```
int numberOfBaskets
int eggsPerBasket
int totalEggs
```

- Assigning values

```
eggsPerBasket = 6;
totalEggs = eggsPerBasket + 3;
eggsPerBasket = eggsPerBasket - 2;
eggsPerBasket++; //increment by 1
```

Operators

- Operators:
  - Assignment: =, +=, …
  - Arithmetic: +,-,*, ++, …
  - Comparisons: <, ==, …

Tracing code (ex 5.1.1)

Instructions ex 5.1.1:

- FIRST think!! And write down what you expect
- THEN check using Greenfoot
- DISCUSS together if different than expected!

- Example, what does nrOfEggsFound become?

```
int nrOfEggsFound = 3;

if ( nrOfEggsFound >=3 ){
    nrOfEggsFound --;
} else {
    nrOfEggsFound ++;
}
```

Tracing code (ex 5.1.1)

```
int nrOfEggsFound = 3;

if ( nrOfEggsFound >=3 ){
    nrOfEggsFound --;
} else {
    nrOfEggsFound ++;
}
```

CODE	VALUE OF nrOfEggsFound
Initialization: `int nrOfEggsFound = 3;`	3
If- branch `nrOfEggsFound --;`	2
Final situation	2

Values are overwritten

- Variable values are copied and overwritten

```
int a = 12;
int b = 4;


b = a;
```

Values are overwritten

```
int a = 12;
int b = 4;
b = a;
```

CODE	VALUE OF a	VALUE OF b
Initialization: `int a = 12;` `int b = 4;`	12	4
Assign value: `b = a;`	12	12

Quiz (discuss)

Swapping

- Computer can only do one thing at a time
- Variable values are copied and overwritten

So, how to swap the contents of 2 variables?

SITUATION	VALUE OF a	VALUE OF b
Initial situation	4	12
Final situation	12	4

Swapping

Imagine 2 glasses in front of you, one filled with cola, the other with fanta.

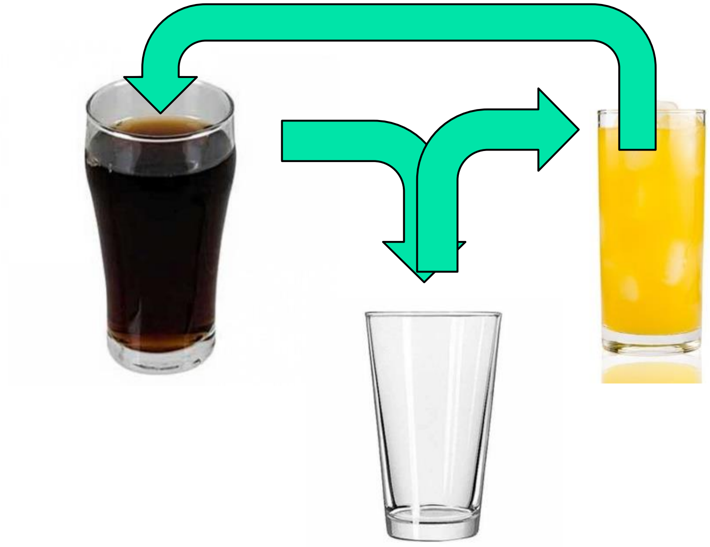
- How do you swap their contents?

Swapping

A computer can only perform 1 action at a time:

- You only have one hand
- A hand can pick up one thing at a time
- Keep in mind: when a variable is assigned a new value, the old value is replaced and cannot be accessed later. (the previous method will result in 2 copies of the same value.)
- How do you swap them?

- A temporary (empty) glass is needed.
- One of the full glasses could be poured into the temporary glass;
- the second glass could be poured into the emptied glass;
- finally the contents of the temporary glass can now be poured into its final destination.



Swapping strategy

- Variable values are copied and overwritten
- To swap, you need an additional ‘temp’ variable

```
int a = 12;
int b = 4;

int temp = a;   // temp becomes 12
a = b;          // a becomes 4
b = temp;       // b becomes 12
```

Variable Swapping strategy

```
int a = 12;
int b = 4;

int temp = a;
a = b;
b = temp;
```

CODE	VALUE OF a	VALUE OF b	VALUE of temp
`int a = 12;` `int b = 4;` `int temp = a;`	12	4	12
`a = b;`	4	4	12
`b = temp;`	4	12	12

isEven

Write a method **boolean isEven (int inputValue)**

Which

- receives an integer inputValue
- returns True or False accordingly

You may not use %

- Tip: you may use a while

isEven (for positive values)

```
public boolean isEven( int inputValue ){
        while ( inputValue > 0 ){
                inputValue = inputValue - 2;
        }

        if ( inputValue == 0 ){
                return true;
        } else {
                return false;
        }

}
```

Swapping strategy (tracing)

CODE	LOOP NR	VALUE OF inputValue	Return VALUE

`while (inputValue > 0){`
`    inputValue = inputValue - 2;`
`}`
	0	4	

	1	2	
	2	0	

`if ( inputValue == 0 ){`
`    return true;`
`} else {`
`    return false;`
`}`
		0	true

Testing cases

- For which values of inputValue must you test?

Unplugged

- Sorting algorithms and efficiency

Sort cards: Bogo Sort

Sort algorithms (in pairs, 5 minutes)

- Goal: Sort cards
  - order: lowest to highest value (2 < 3 < … < 10 < J < … < A)
  - student 1 selects 2 cards (without seeing their value)
  - student 2 compares the cards and tells which one has the highest value.
  - nr of steps?
- Describe an algorithm (with a flowchart) using basic instructions which a 4-year-old should be able to follow:
    - getCard (thirdCard)
    - determineHighestCard (thirdCard , seventhCard)

Sort algorithms: efficiency (2 minutes)

- Efficiency: Write down how many steps if you have:
  - 10 cards
  - 20 cards
  - 100 cards

Sort algorithms

- Share:
  - What did you come up with?
  - Efficiency

Quick sort: divide and conquer

1) Select a card at random

2) Divide collection into two groups:

   A) larger than selected card

   B) smaller than selected card

3) Give each pile of cards to another team

   & sit back and relax

4) Other teams repeat steps 1-3

When are we done?

Quick sort: divide and conquer

0) If you have 0 or 1 card, then STOP

1) Select a card at random

2) Divide collection into two groups:

    A) larger than selected card

    B) smaller than selected card

3) Give each pile of cards to another team

    Other teams repeat steps 1-3

Result: cards sorted from smallest to largest

Method: divide and conquer (recursive algorithm)

Quick sort summary

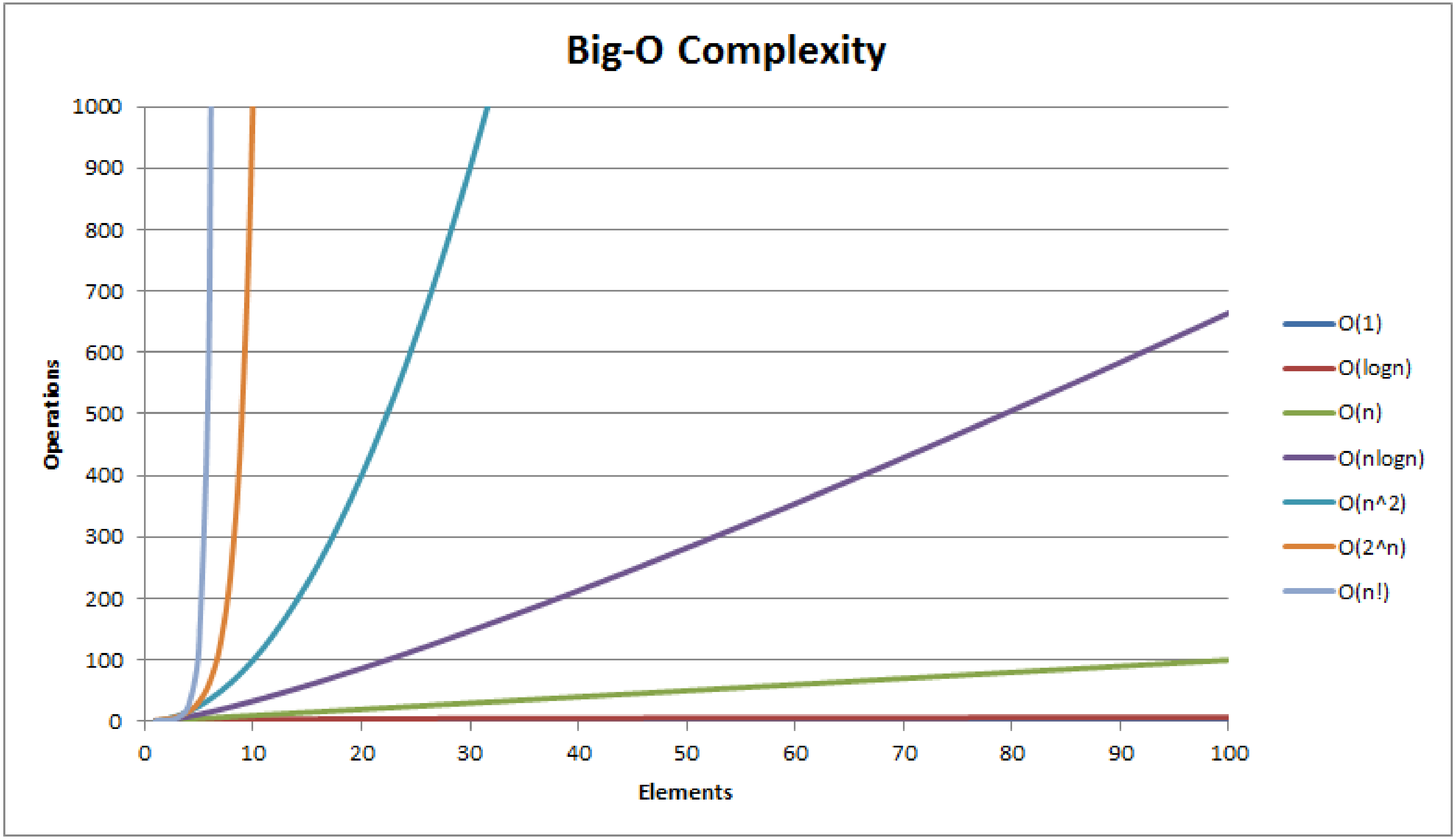
- Divide and conquer: Recursive programming
- Simple instructions
- Complexity n*log(n)

Growth Rates Compared:

	n=1	n=2	n=4	n=8	n=16	n=32
1	1	1	1	1	1	1
logn	0	1	2	3	4	5
n	1	2	4	8	16	32
nlogn	0	2	8	24	64	160
n^2	1	4	16	64	256	1024
n^3	1	8	64	512	4096	32768
2^n	2	4	16	256	65536	4294967296
n!	1	2	24	40320	20.9T	Don't ask!

Quick sort summary

- Complexity O(n*log(n)): purple curve



How much better is QuickSort?

https://www.youtube.com/watch?v=aXXWXz5rF64

Wrapping up

Homework for Wednesday 8:30 April 20th:

- Assignment 5:
  - **Finish assignment 5**
  - **Hand via email to sjaaksm@live.com**