



Algorithmic Thinking and Structured Programming (in Greenfoot)

Teachers:

Renske Smetsers-Weeda

Sjaak Smetsers

Ana Tanase



Today's Lesson plan (6)

- Looking back
 - Retrospective last lesson

- Blocks of theory and exercises
 - Variables and Operators
 - Tracing code
 - Quiz demo



Retrospective: assignment 4

- Conditionals:
 - boolean methods
 - logical operators: ||, &&, !
- Nested if-then-else
- Return statements
- Modularization: Breaking problem down, solving subproblems (using existing solutions), and combining to solve the whole problem
 - Method calls (from within other methods)
 - Simplifies testing



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

Ants are **smart**.
We remember
where home is.



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)

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

```
nrEggsFound = 15;
```

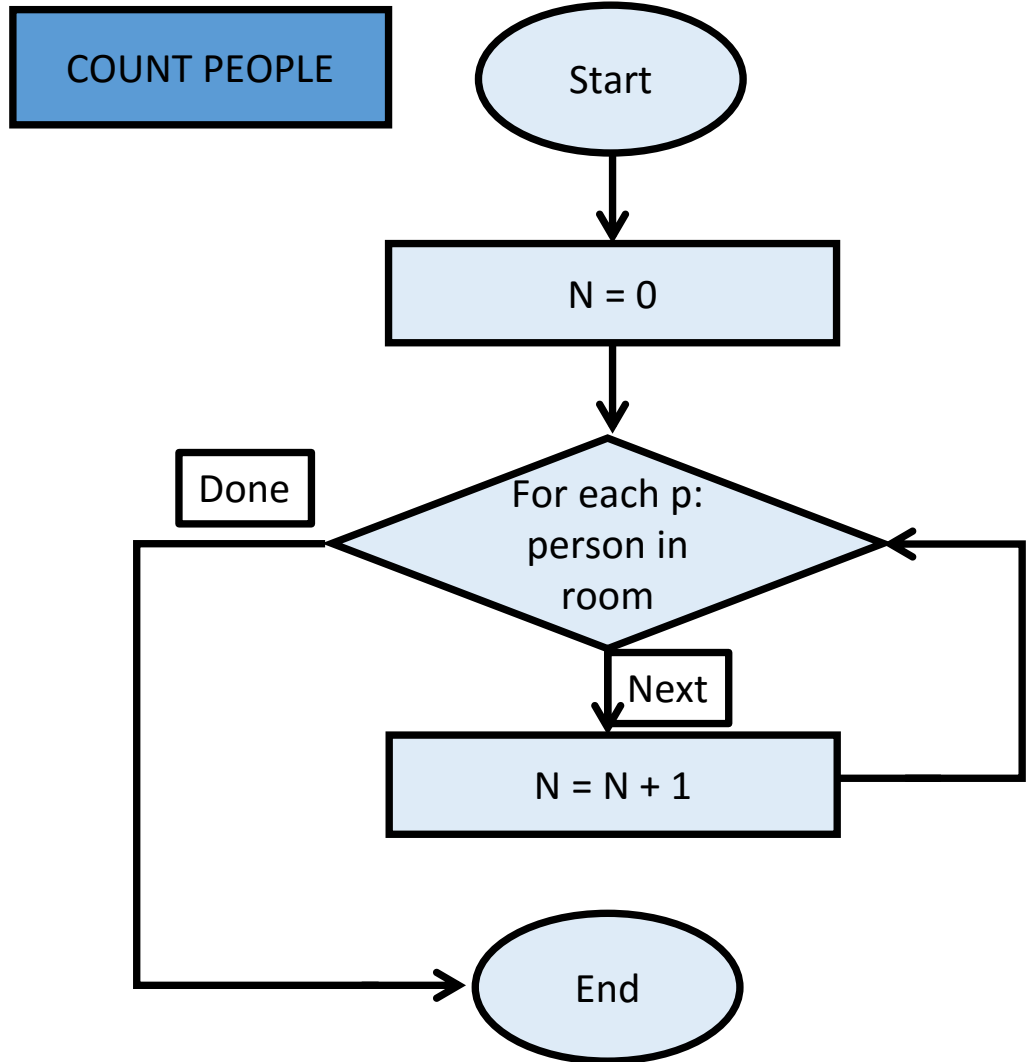
- Two kinds of variables:
 1. **Local variables**
 2. **Instance variables**

Pronounced as 'becomes'

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*

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

Memory

0

1



Variables and Values

- Variables

```
int numberOfBaskets
```

```
int eggsPerBasket
```

```
int totalEggs
```


- Assigning values

```
eggsPerBasket = 6;
```

```
totalEggs = eggsPerBasket + 3;
```

```
eggsPerBasket = eggsPerBasket - 2;
```

```
eggsPerBasket = eggsPerBasket ++; //inc 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: <code>int nrOfEggsFound = 3;</code>	3
If- branch <code>nrOfEggsFound --;</code>	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: <pre>int a = 12; int b = 4;</pre>	12	4
Assign value: <pre>b = a;</pre>	12	12




Code Tracing: why bother?

- Research shows:
 - Many students make mistakes understanding and using variables;
 - Just a few types of bugs account for the majority of students' mistakes;
 - Learning debugging strategies saves a lot of time finding bugs;
 - Debugging helps learn about code constructs.



Get started on ex 5.1.1



Quiz demo



Quiz (discuss)



Swapping

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

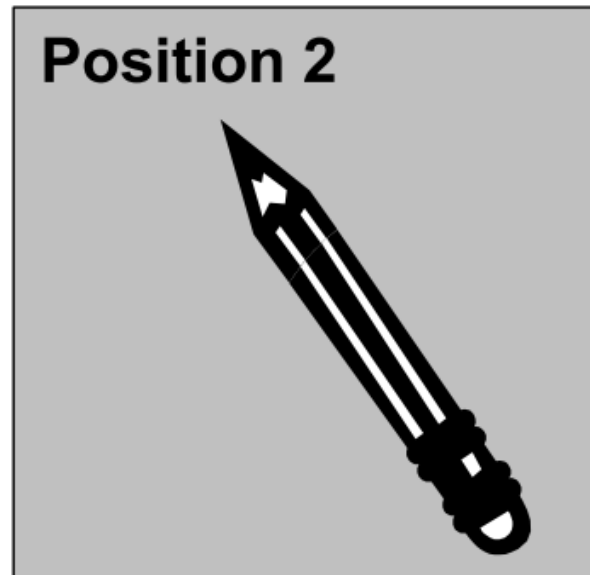
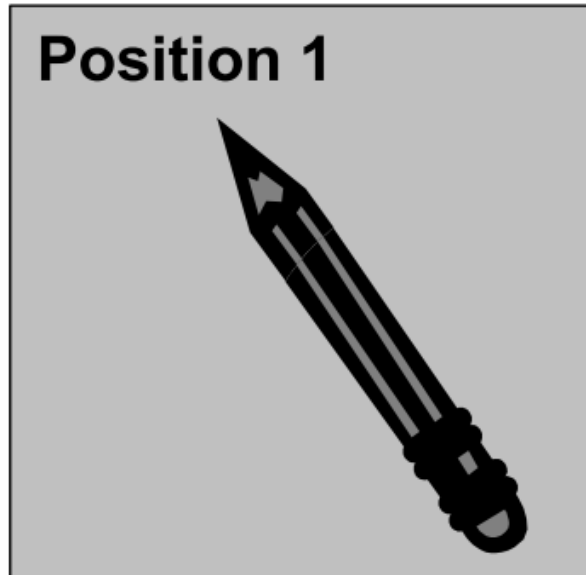
So, how to swap the contents of 2 variables?

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

Swapping

Imagine 2 pencils in front of you.

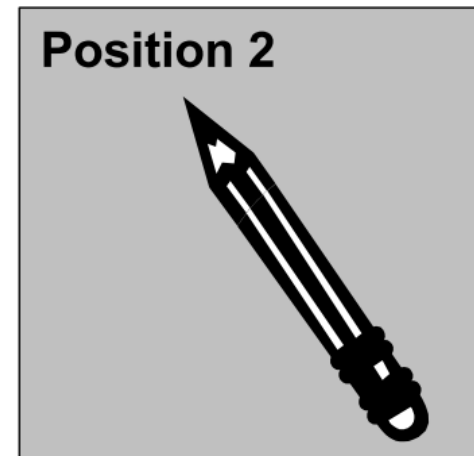
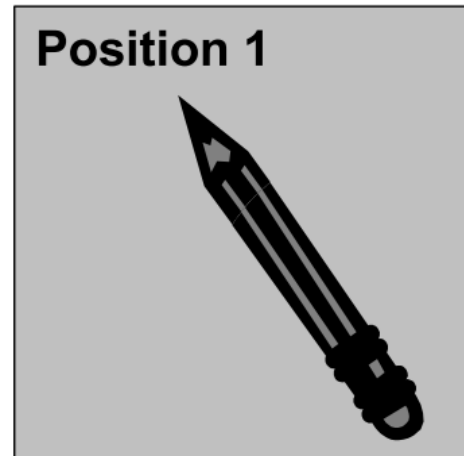
- How do you swap them?



Swapping

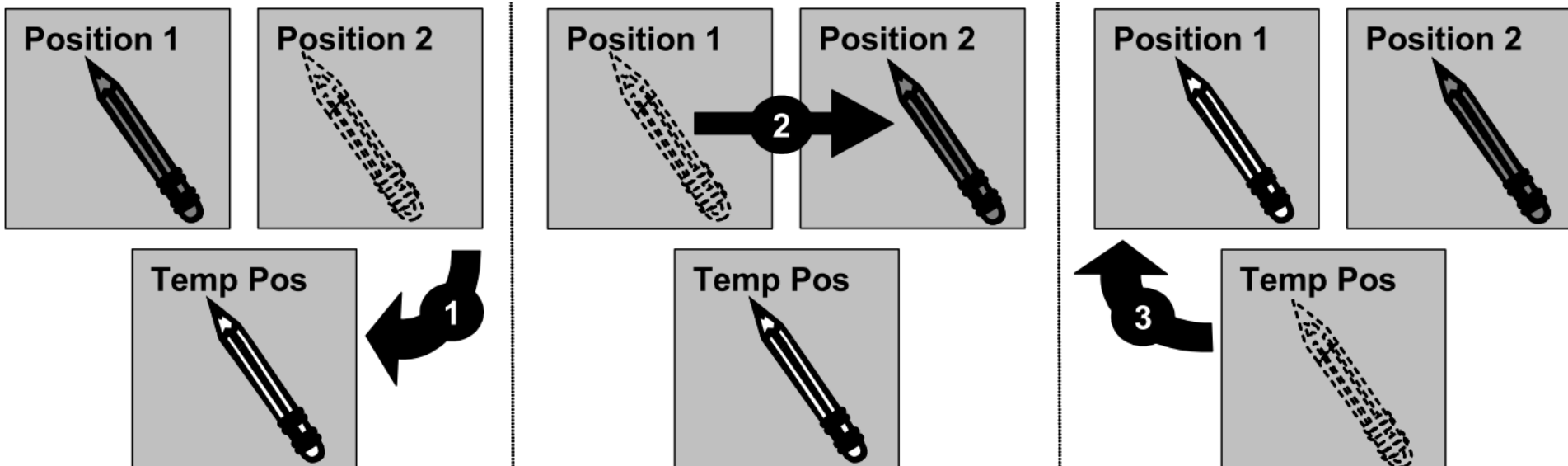
A computer can only perform 1 action at a time:

- ❑ 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 position is needed.
- One of the pencils could be moved to the temporary position;
- the second pencil could be moved to its new location;
- finally the first pencil could be moved from the temporary position to its new position.





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
```



Swapping strategy

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

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

CODE	VALUES
<pre>int a = 12; int b = 4; int temp = a;</pre>	<pre>a == 12 b == 4 temp == 12</pre>
<pre>a = b;</pre>	<pre>a == 4 b == 4 temp == 12</pre>
<pre>b = temp;</pre>	<pre>a == 4 b == 12 temp == 12</pre>



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
<pre>int a = 12; int b = 4; int temp = a;</pre>	12	4	12
<pre>a = b;</pre>	4	4	12
<pre>b = temp;</pre>	4	12	12



Variable Scope (lifetime)

```
public int walkAndCountSteps () {  
    int stepsTaken=0;  
    while ( canMove () ) {  
        stepsTaken++;  
        move ();  
    }  
    return stepsTaken;  
}
```



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
<pre>while (inputValue > 0){ inputValue = inputValue - 2; }</pre>	0	4	
	1	2	
	2	0	
<pre>if (inputValue == 0){ return true; } else { return false; }</pre>		0	true



Testing cases

- For which values of inputValue must you test?



Continue with the assignments

Homework for Wednesday 8:30 January 27th:

□ Assignment 5:

■ **UNTIL AND INCL 5.1.5**



Computational thinking

- **Working in a structured manner:**
 - Breaking problems down into subproblems
 - Design, solve and test solutions to subproblems
 - Combing these (sub)solutions to solve problem
- **Analyzing** the quality of a solution
- **Reflecting** about the solution chosen and proces
- **Generalizing** and re-use of existing solutions



Questions?



Wrapping up

Quiz on Feb 5th

No class next Friday (January 22nd)

Homework for Wednesday 8:30 January 27th:

□ Assignment 5:

- **UNTIL AND INCL 5.1.5**
- ZIP code and 'IN' and **email** to **Renske.weeda@gmail.com**