Algorithmic Thinking and Structured Programming (in Greenfoot)

Teachers:

Renske Smetsers-Weeda

Sjaak Smetsers

Ana Tanase

Today's Lesson plan (4)

- 20 min Quiz
- 10 min Looking back
  - What did we learn last week?
  - Discuss problems / homework (and handing-in)

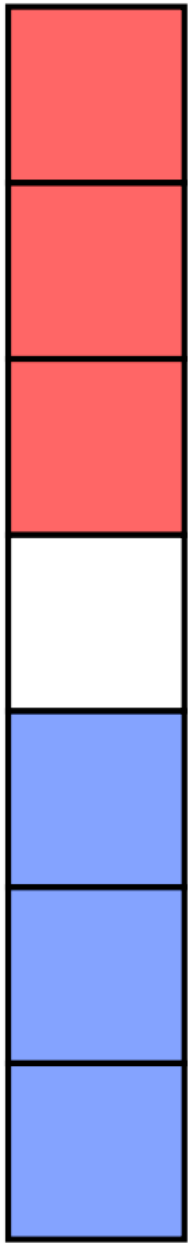
- Blocks of theory and exercises / unplugged

- 10 min Wrapping up
- Next class: Fri Jan 8th (NOT next week)

Swap Puzzle level 3

Describe a strategy

- In Java: **square1 = square2;**
- Which means: square 1 getsPieceFrom square2



Swap Puzzle level 3

The level 3 puzzle can be solved in 15 moves as follows:

Step 1: **Square 3** GETS THE PIECE FROM **Square 2**

Step 2: **Square 2** GETS THE PIECE FROM **Square 4**

Step 3: **Square 4** GETS THE PIECE FROM **Square 5**

Step 4: **Square 5** GETS THE PIECE FROM **Square 3**

Step 5: **Square 3** GETS THE PIECE FROM **Square 1**

Step 6: **Square 1** GETS THE PIECE FROM **Square 0**

Step 7: **Square 0** GETS THE PIECE FROM **Square 2**

Step 8: **Square 2** GETS THE PIECE FROM **Square 4**

Step 9: **Square 4** GETS THE PIECE FROM **Square 6**

Step 10: **Square 6** GETS THE PIECE FROM **Square 5**

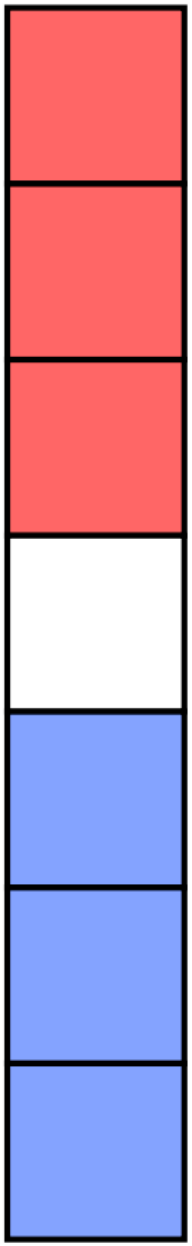
Step 11: **Square 5** GETS THE PIECE FROM **Square 3**

Step 12: **Square 3** GETS THE PIECE FROM **Square 1**

Step 13: **Square 1** GETS THE PIECE FROM **Square 2**

Step 14: **Square 2** GETS THE PIECE FROM **Square 4**

Step 15: **Square 4** GETS THE PIECE FROM **Square 3**



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

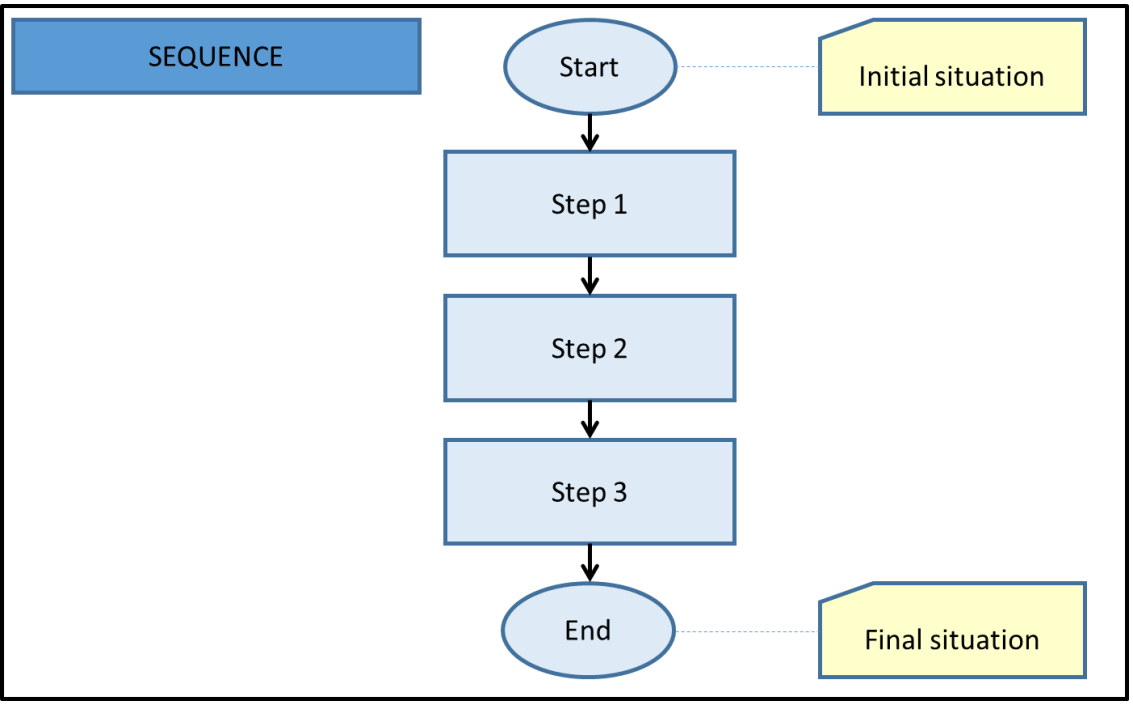
Discuss problems / homework

- Only hand in (via email):
  - MyDodo.java
  - Document with answers to **only** (IN) questions

- Any problems? Please email!

Sequence

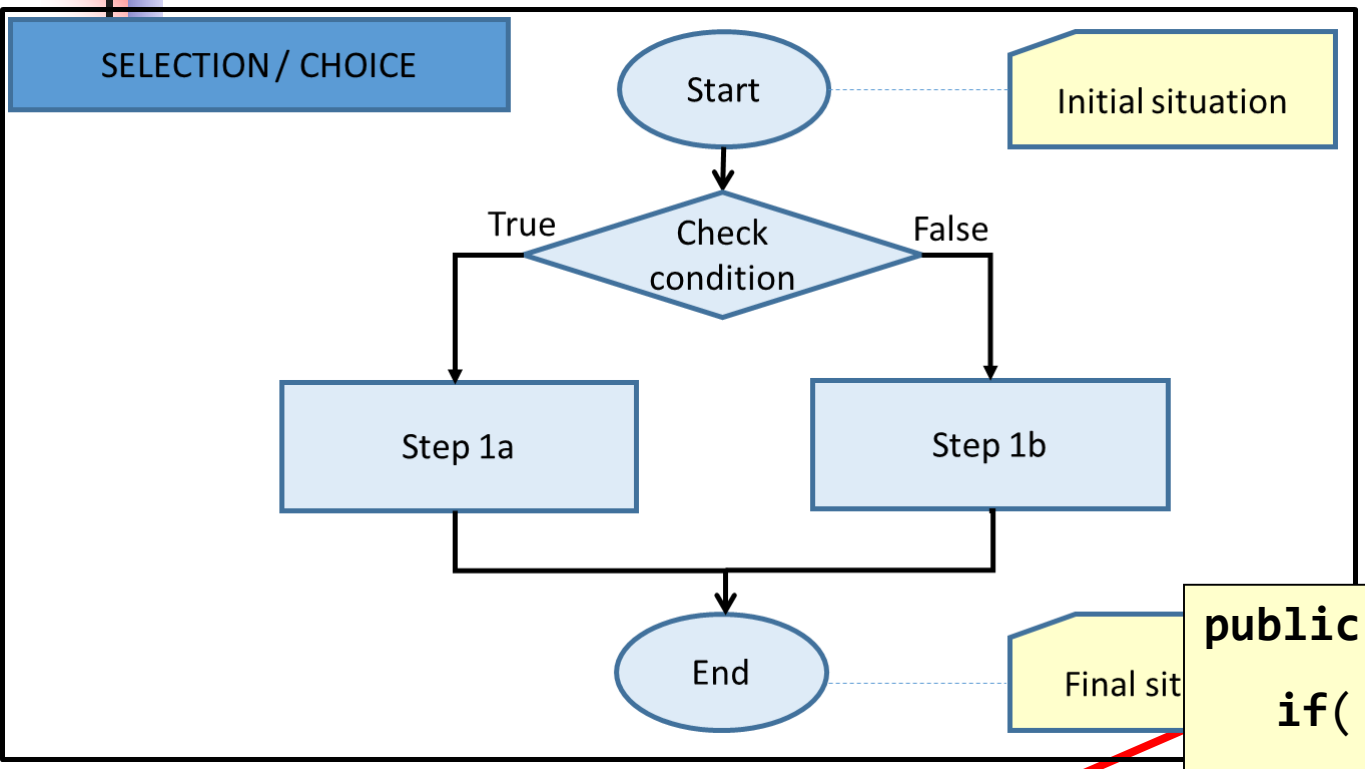
flowchart



code

```
public … methodName( … ) {
   step1();
   step2();
   step3();
}
```

Selection (choice, if..then..else)

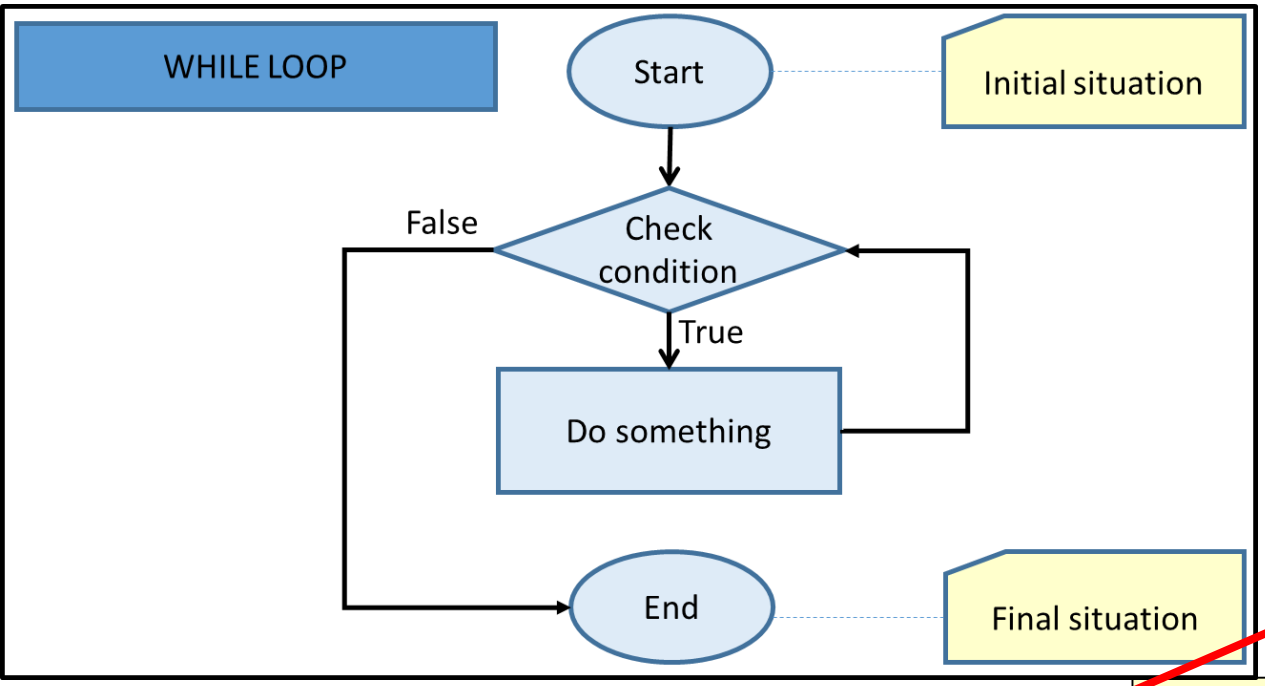


flowchart

code

```
public … methodName( … ) {
   if( check () ) {
      step1a();
   }else{
      step1b();
   }
}
```

Repetition (iteration, loop)



flowchart

code

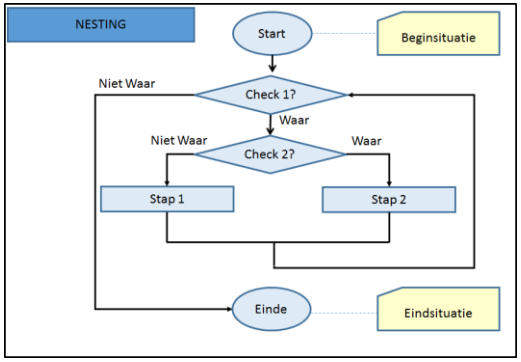
```
public … methodName( … ) {
   while ( check () ) {
      doSomething ();
   }
}
```

Challenge & problem

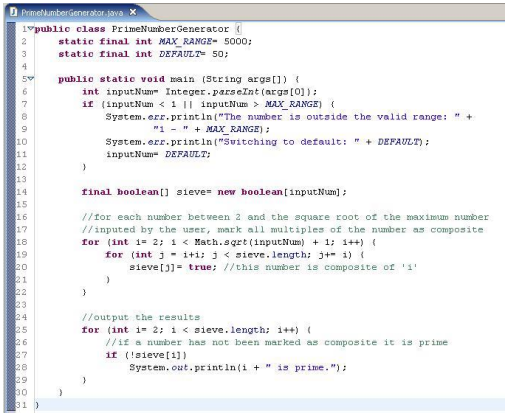
You must perform two aspects well:

1) Create a *problem-solving algorithm* (a disciplined and creative process)

We use a systematic approach



2) *Formulate* that algorithm *in terms of a programming language* (a disciplined and very precise process)



```
PrimeNumberGenerator.java
public class PrimeNumberGenerator {
    static final int MAX_RANGE= 5000;
    static final int DEFAULT= 50;

    public static void main (String args[]) {
        int inputNum= Integer.parseInt(args[0]);
        if (inputNum < 1 || inputNum > MAX_RANGE) {
            System.err.println("The number is outside the valid range: " +
                    "1 - " + MAX_RANGE);
            System.err.println("Switching to default: " + DEFAULT);
            inputNum= DEFAULT;
        }

        final boolean[] sieve= new boolean[inputNum];

        //for each number between 2 and the square root of the maximum number
        //inputed by the user, mark all multiples of the number as composite
        for (int i= 2; i < Math.sqrt(inputNum) + 1; i++) {
            for (int j = i+i; j < sieve.length; j+= i) {
                sieve[j]= true; //this number is composite of 'i'
            }
        }

        //output the results
        for (int i= 2; i < sieve.length; i++) {
            //if a number has not been marked as composite it is prime
            if (!sieve[i])
                System.out.println(i + " is prime.");
        }
    }
}
```

Always check that your algorithm is correct by running/testing the implementation!

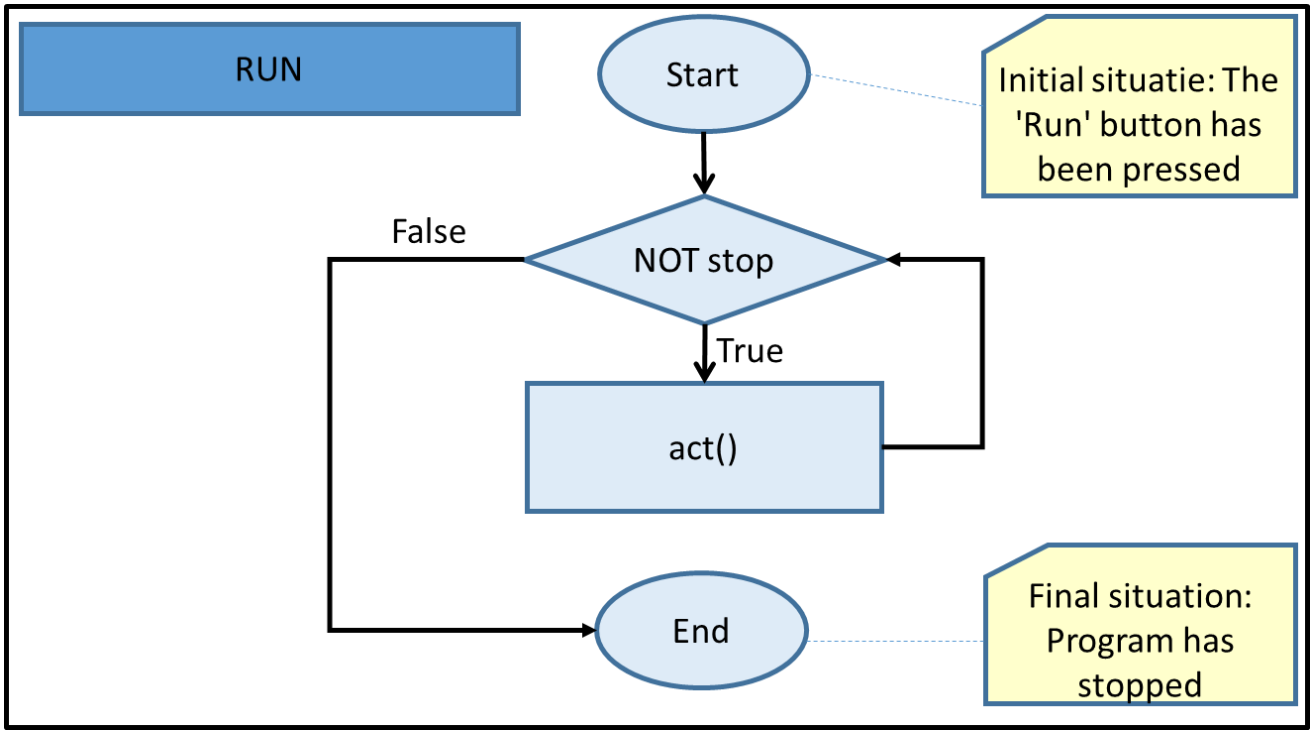
Today:

- Greenfoot Run: ‘Act’ in a while loop
  - Greenfoot.stop()
- Parameters
- Submethods: a method call in a method
- Boolean expressions (NOT, OR, AND)

Greenfoot Run

- Run is a special Greenfoot feature
- Run: Act called repeatedly
  - Act in a while loop

>Run: built –in iteration



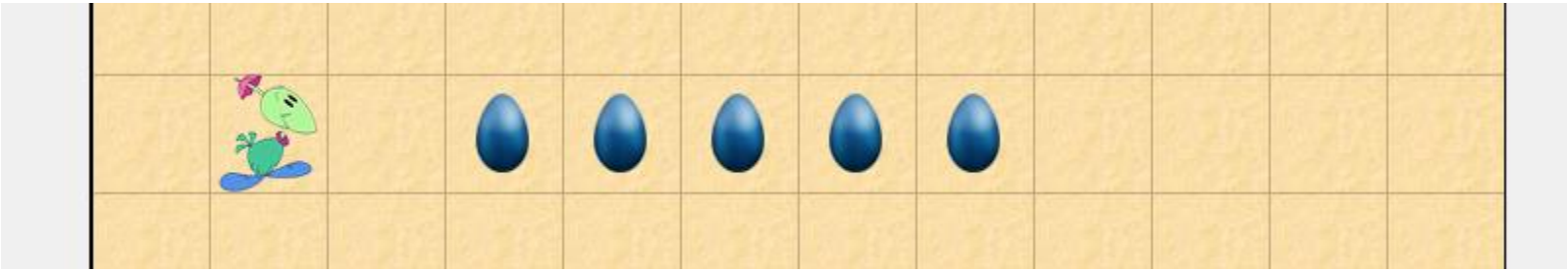
Can only be interrupted by:

- Pressing ‘Pause’
- Calling Greenfoot.stop()

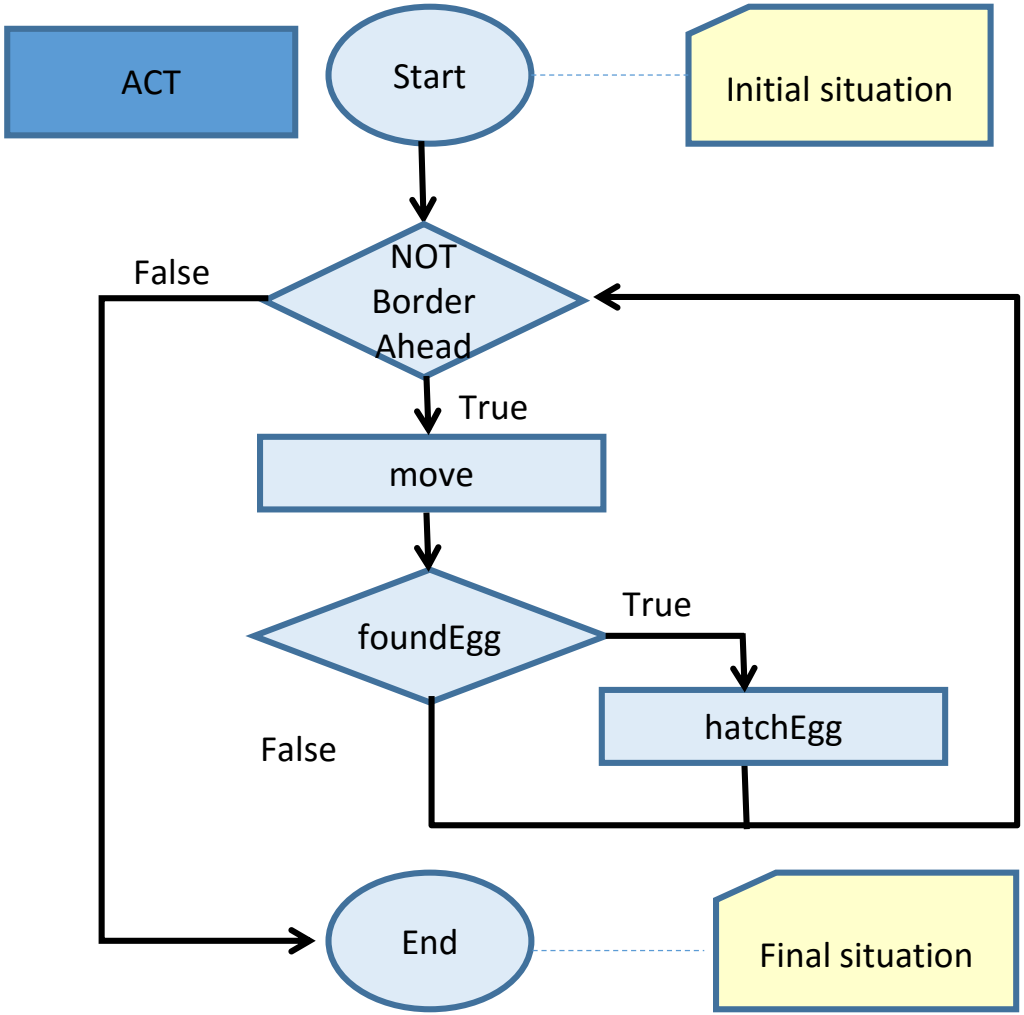
>Run: calling `act` repeatedly

What if your **`void`** `act()` contains a while-loop?

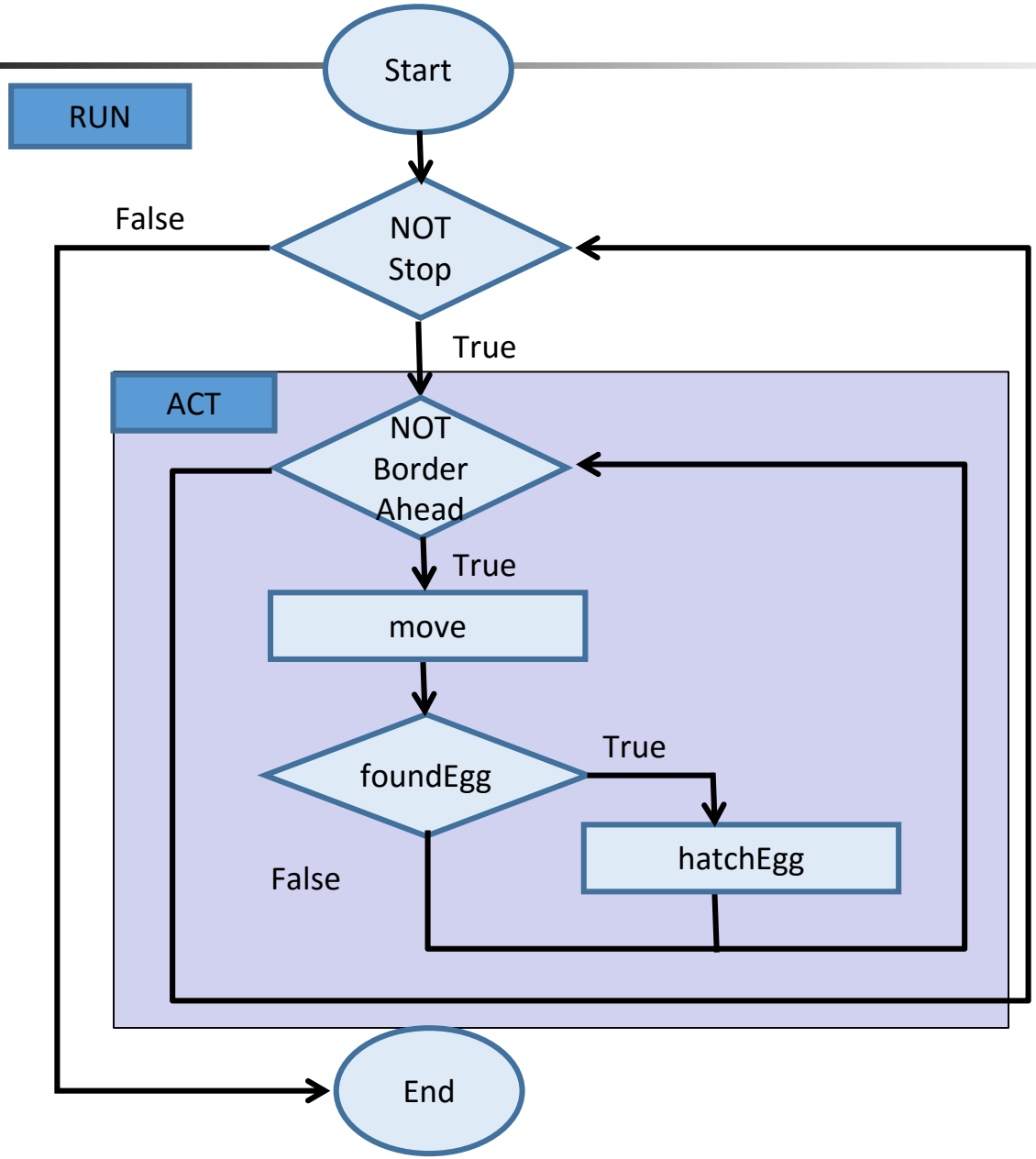
Example: hatching a row of eggs



Iteration in act

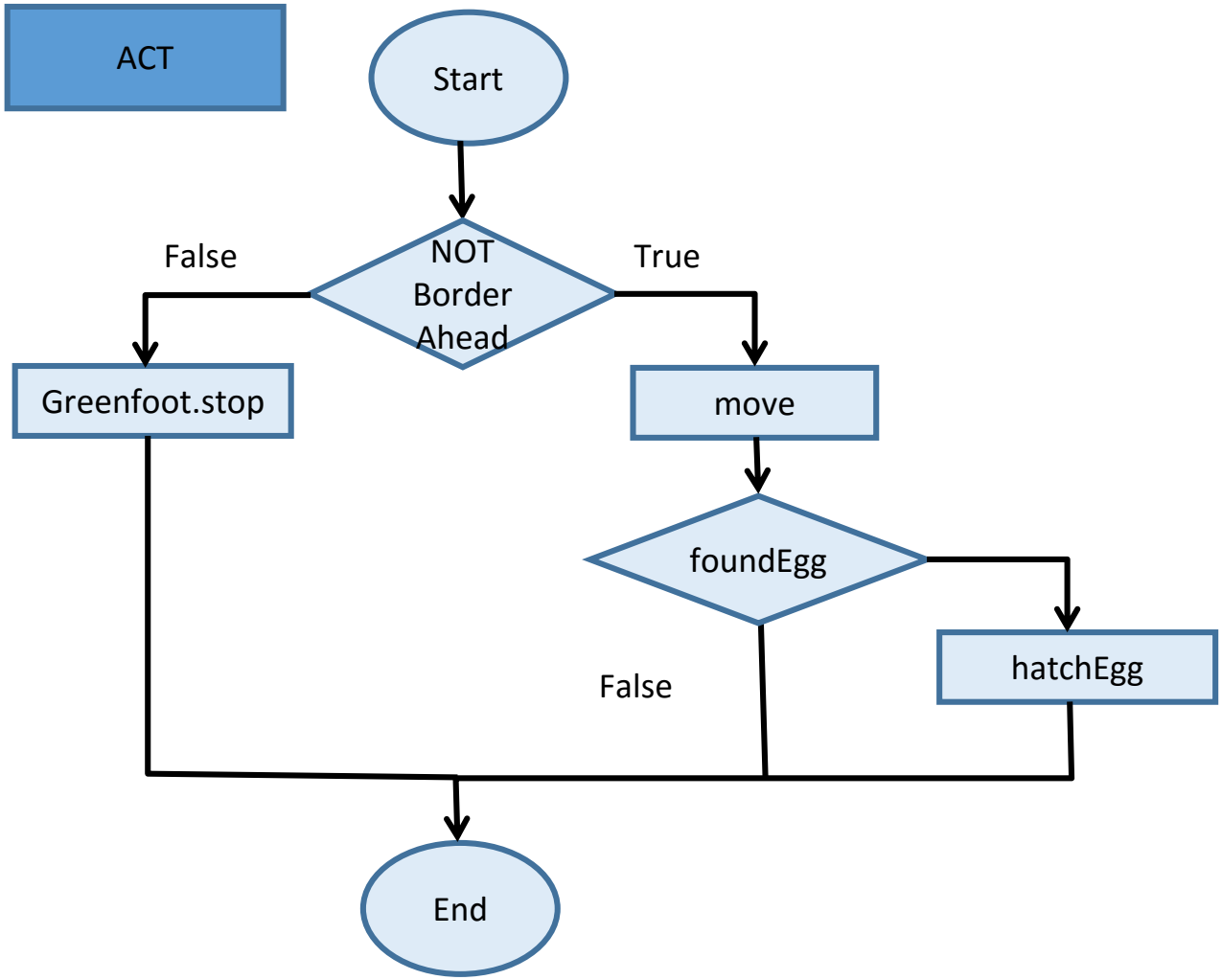


Act in Run



- One loop is superfluous
- Try to eliminate the act loop: Keep your act as simple/small as possible.

No iteration in act



The golden-promise:

- Don’t put too much work in the act method.
  - Avoid time-consuming while-loops or while-loops with ‘visible effects’.

JAVA: Printing to screen(console)

- Printing text to the screen (console) with:

  System.out.println (“Hello“);

- Print a variable studentName to the screen with:

  System.out.println (studentName);

- Print a combination of text and variable:

  System.out.println (“Hello ” + studentName);

- Example code:

  studentName = “Jack”;

  System.out.println (“Hello ” + studentName);

Outputs to screen: “Hello Jack”

Unplugged Songwriting

- Parameters
- Submethods

Songwriting: Parameters & Submethods

Old MACDONALD had a farm
E-I-E-I-O
And on his farm he had a cow
E-I-E-I-O
With a moo moo here
And a moo moo there
Here a moo, there a moo
Everywhere a moo moo
Old MacDonald had a farm
E-I-E-I-O

….

Song goes on for (just about) ever

More generic: Finding parameters

Old MACDONALD had a farm
E-I-E-I-O
And on his farm he had a **cow**
E-I-E-I-O
With a **moo moo** here
And a **moo moo** there
Here a **moo**, there a **moo**
Everywhere a **moo moo**
Old MacDonald had a farm
E-I-E-I-O

Old MACDONALD had a farm
E-I-E-I-O
And on his farm he had a **<ANIMAL>**
E-I-E-I-O
With a **<SOUND> <SOUND>** here
And a **<SOUND> <SOUND>** there
Here a **<SOUND>** , there a **<SOUND>**
Everywhere a **<SOUND> <SOUND>**
Old MacDonald had a farm
E-I-E-I-O

More generic: Using parameters

Old MACDONALD had a farm
E-I-E-I-O
And on his farm he had a **<ANIMAL>**
E-I-E-I-O
With a **<SOUND> <SOUND>** here
And a **<SOUND> <SOUND>** there
..
Old MacDonald had a farm
E-I-E-I-O

```
System.out.println("Old MACDONALD had a farm");
System.out.println("E-I-E-I-O");
System.out.println ("And on his farm he had a " + animal );
System.out.println("E-I-E-I-O");
System.out.println("With a " + sound + " " + sound+ "here" );
System.out.println("And a " + sound + " " + sound + "there" )
…
System.out.println("Old MACDONALD had a farm");
System.out.println("E-I-E-I-O");
```

Introducing parameters

```
public void singOldMcDonaldChorusWithParameters ( String animal, String sound ) {
    System.out.println( "Old MACDONALD had a farm" );
    System.out.println( "E-I-E-I-O" );
    System.out.println( "And on his farm he had a " + animal );
    System.out.println( "E-I-E-I-O" );
    System.out.println( "With a " + sound + " " + sound+ " here" );
    System.out.println( "And a " + sound + " " + sound + " there" );
    System.out.println( "Here a " + sound + ", there a " + sound );
    System.out.println( "Everywhere a " + sound + " " + sound );
    System.out.println( "Old MACDONALD had a farm" );
    System.out.println( "E-I-E-I-O" );
}
```

Generic: Using parameters

```
public void singOldMcDonaldSongWithParameters ( ) {
    singOldMcDonaldChorusWithParameters ( "cow", "moo" );
    singOldMcDonaldChorusWithParameters ( "pig", "oink" );
    singOldMcDonaldChorusWithParameters ( "duck", "quack" );
    singOldMcDonaldChorusWithParameters ( "lam", "baa" );
}
```

More generic: finding repetition

Old MACDONALD had a farm
E-I-E-I-O
And on his farm he had a cow
E-I-E-I-O
With a moo moo here
And a moo moo there
Here a moo, there a moo
Everywhere a moo moo
Old MacDonald had a farm
E-I-E-I-O

Defining submethods [1]

```
public void printOldMcHadFarm ( ) {
        System.out.println("Old MACDONALD had a farm");
}

public void singOldMcDonaldChorus ( String animal, String sound ) {
        printOldMcHadFarm ( );
        System.out.println( "E-I-E-I-O" );
         System.out.println( "And on his farm he had a " + animal );
        System.out.println( "E-I-E-I-O" );
        …
}
```

Why submethods [1]: easy to change

- Change in 1 place
- From:

```
public void printOldMcHadFarm ( ) {
        System.out.println("Old MACDONALD had a farm");
}
```

- Into:

```
public void printOldMcHadFarm ( ) {
        System.out.println("Old McDonald had a farm");
}
```

Defining submethod with arguments

```
public void printHadAnimal ( String animal ) {
        System.out.println("And on his farm he had a " + animal );
}

public void singOldMcDonaldChoruss ( String animal, String sound ) {
        printOldMcHadFarm();
        printEIEIO();
        printHadAnimal ( animal );
        printEIEIO ();
        …
         printOldMcHadFarm();
         printEIEIO ();
}
```

Why submethods [2]: easy to read

```
public void singOldMcDonald ( String animal, String sound ) {
        printOldMcHadFarm();
        printEIEIO();
        printHadAnimal ( animal );
        printEIEIO ();
        printWithSound (sound );
        printAndSound (sound );
        printOldMcHadFarm();
        printEIEIO ();
}
```

Why submethods and arguments

- More generic:
  - Less code
    - Less mistakes
    - Easier to read / understand
  - Code can be used for more (… animals)
  - Easier to change
  - Easier to reuse

Your turn!

The wheels on the bus go round and round,
round and round,
round and round.
The wheels on the bus go round and round,
all through the town.

- *The doors on the bus go open and shut.*
- *The wipers on the bus go Swish, swish, swish*

On paper:

- Find parameters and replace text
- Find and use submethods
- Write method to print song using parameters & submethods

Boolean quiz

- Answer questions on paper (incl your name)
- Hand-in
- Papers will be shuffled
- Teacher chooses paper and reads last statement
- If this is you…. DON’T SAY A THING
- Everyone stands
- Teachers reads statements:
  - If True about you: stay standing
  - If False about you: sit down

Answer the following

1. What is your favorite number?
2. What is the color of your bicycle?
3. What is your favorite color?
4. What month were you born?
5. Do you have siblings?
6. What is the last digit of your phone number?
7. What is something about you that people here don’t know and can’t tell by looking at you?

Boolean Quiz

Boolean statements

What it's about:

- In English, an
  - "or" is often an "exclusive or"
  - such as "You can have chicken or fish."
  - In English, you only get to pick one

- But with **Boolean** logic you could have
  - chicken, fish, **or both**!!
  - A **||** B means: (A **or** B) **or (A and B)**

True or False?

- Booleans can be true or false
- Boolean statements can be made very complex using combinations:
  - NOT: !
  - AND: &&
  - OR: ||

  For example: (A || B) && ! ((A && B) || C)
- Careful: Often a source of errors!

Modifying code

- After each MINOR adjustment
  - Compile
  - Test if it still works

- If you do too much at once, and then get an error…
  - … you're doomed to get frustrated!

- Remember, from our first lesson:
  - Expect to make mistakes! Learn from them.

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

Homework for Wednesday 8:30 Jan.6th 2016:

- Assignment 3: finish
- Assignment 4 up to and incl 5.2.2

Hand in:

- **Via email** : MyDodo.java and 'IN'

  to **Renske.weeda@gmail.com**

(Flowcharts: op paper in **pigeonhole** or photo/scan and paste into document)