Algorithmic Thinking
and
Structured Programming
(in Greenfoot)

Teachers:

Renske Smetsers-Weeda

Sjaak Smetsers

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

Today's Lesson plan (2)

- 15 min Pre-test: what DID you already know?
- Blocks of theory and exercises
  - Finish assignment 1
  - Begin assignment 2
- 10 min Wrapping up
  - Saving work
  - Handing-in
  - Plenary reflection

Last weeks homework:

❑ Methods:
- Mutator methods such as void move()
- Accessor methods such as boolean canMove()

❑ Result types such as int, boolean, void

❑ Java is an Object Oriented Programming (OOP) language

❑ In OOP, objects (such as MyDodo) have:
- Methods (*what it can do*)
- States (*what it knows / is*)

Objects and classes

- Every object belongs to a class.
  - a class provides the blueprint for objects
 - Mimi is an instance (or object) of the MyDodo class

Class diagram ➡️

Actor
Egg
  BlueEgg
  GoldenEgg
Message
  Compliment
  Alert
Scoreboard
Dodo
  MyDodo
Fence
Nest

inherited from Actor
inherited from Dodo

void act()
boolean canMove()
int getNrOfEggsHatched()
void hatchEgg()

void act()   [redefined in MyDodo]
GreenfootImage getImage()
int getRotation()
World getWorld()
int getX()
int getY()
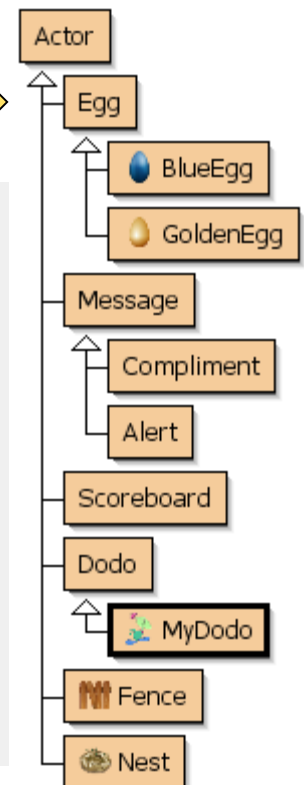
Inheritance

- Class diagram
- Mimi is a MyDodo, so:
  - Mimi can perform MyDodo methods, such as:
    - void move()
- But a MyDodo is a Dodo
  - Mimi can also perform all Dodo methods too!
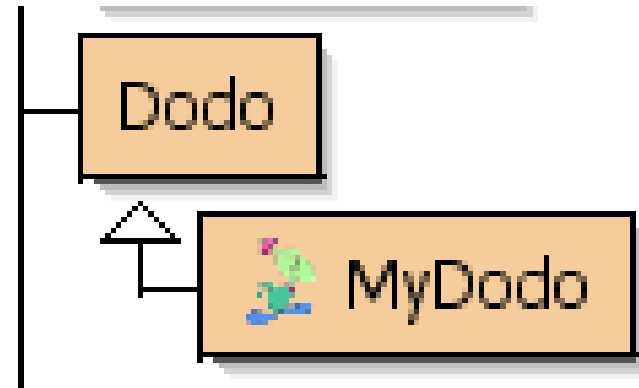   - void layEgg()

- MyDodo is a subclass of Dodo
- Dodo is a superclass of MyDodo
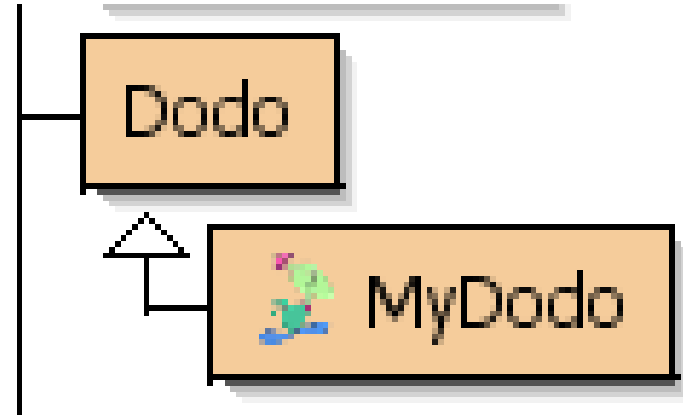
Exercise: inheritance

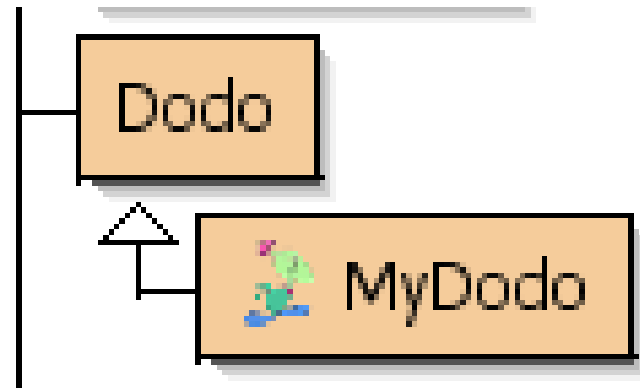Imagine a new Dodo species: IntelligentDodo

- ❑ Sketch the class diagram
- ❑ Which is subclass? Which is superclass?

Exercise: inheritance

Imagine a new Dodo species: IntelligentDodo

❑ What methods does IntelligentDodo get for free from her superclass?

❑ If we write a new IntelligentDodo method: public void readBook()

Which class(es) can use that method (more than one answer is possible):

a) IntelligentDodo
b) MyDodo
c) Dodo
d) Actor
e) Mimi

State of an object

blueegg2 : BlueEgg

private int myValue	1
int x	1
int y	1
int rotation	0
World world	
private GreenfootImage image	

Show static fields

Inspe
Get

Clo

- Every object has a state: its data

- 3 Egg objects
    - have the **same methods** (can do the same things)
    - but are different objects (or instances)
- Each can have an **own** state
    - for example: different coordinates

0 or 1 Results

- A method can return **one** result:
  - Example: public int getNrOfEggsHatched()
  - Returns an int (whole number) as a result

- A method can have **NO** results
  - public void move()
  - Returns void (nothing) as a result

- A method **cannot return more than one** result

0 or more Parameters

A method can have:

- zero parameters:
    - Example: public void move( )
    - Has **no** parameters
- one parameter:
    - Example: public void jump (int distance)
    - Has **one** parameter: distance
    - Type of the parameter: int
- more than one parameter:
    - Example:

        public void jumpToLocation(int xCoord, int yCoord)
    - Has **two** parameters: xCoord and yCoord

Generic methods and parameters

- Method with parameters can be used for more things
  - jump(1): Mimi jumps 1 place forward
  - jump(2): Mimi jumps 2 places forward
  - jump(100): Mimi jumps 100 places forward

- Method without parameters can only be used for 1 thing:
  - move( ): Mimi moves 1 place forward

- Generic: method with parameters is more generic, because it can be used in more situations.
- We **LIKE** generic methods! They're **SMART**.

Types

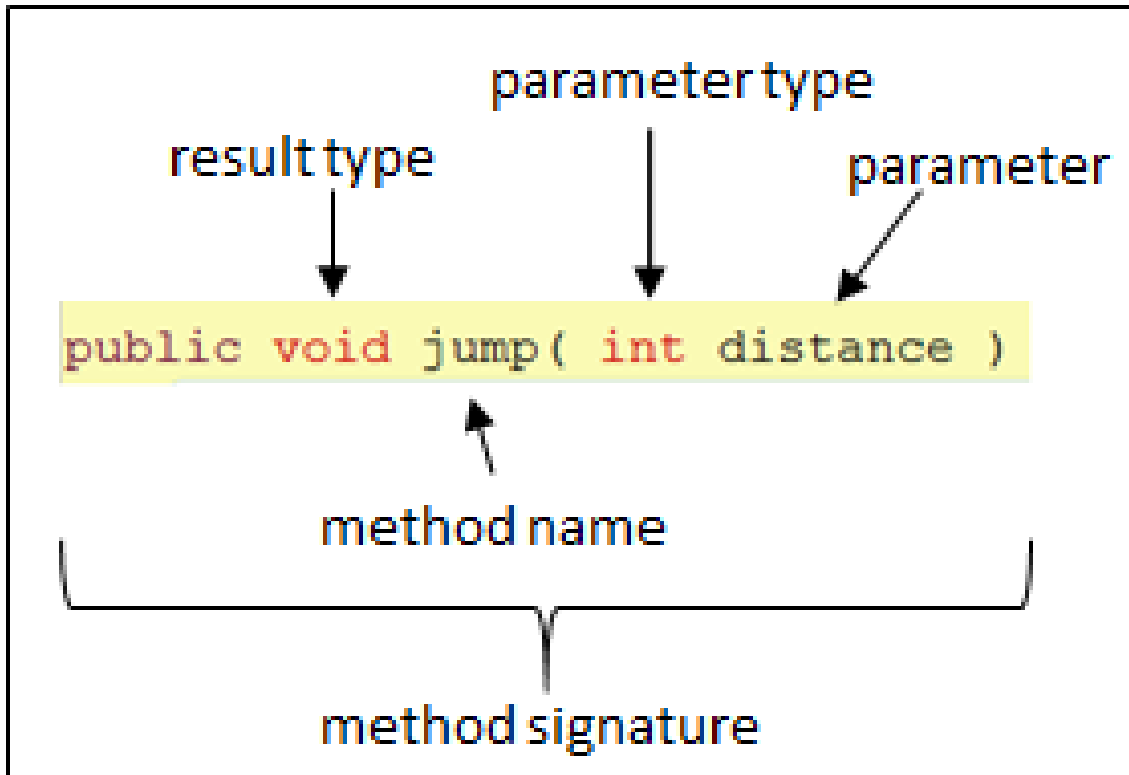- Results and parameters have **types**
- Examples:

Type	Meaning	Example
int	whole number	2
boolean	'true' or 'false'	true
String	text	"I lost my pen."
List	list	[1,2,3]

Signature

Intermezzo: Assignment

❑ Download and unzip the scenario 2 at
http://www.cs.ru.nl/~S.Smetsers/Greenfoot/Dominicus/

❑ Class will continue in 15 minutes

Algorithm

- Algorithm: precise set of instructions
  - For a certain problem (initial situation)
  - Always leads to exact same outcome (final situation)

  - Like a recipe, but more precise

- Program code: algorithm written specifically for a computer

Language ambiguous

"Time flies like an arrow"

What could this mean?

Language ambiguous

"Time flies like an arrow"

- Time moves fast, like an arrow moves fast
- Measure the speed of flies which resemble arrows
- Measure the speed of flies in the same way you would measure the speed of an arrow
- Insects of a type known as 'time flies' are fond of arrows
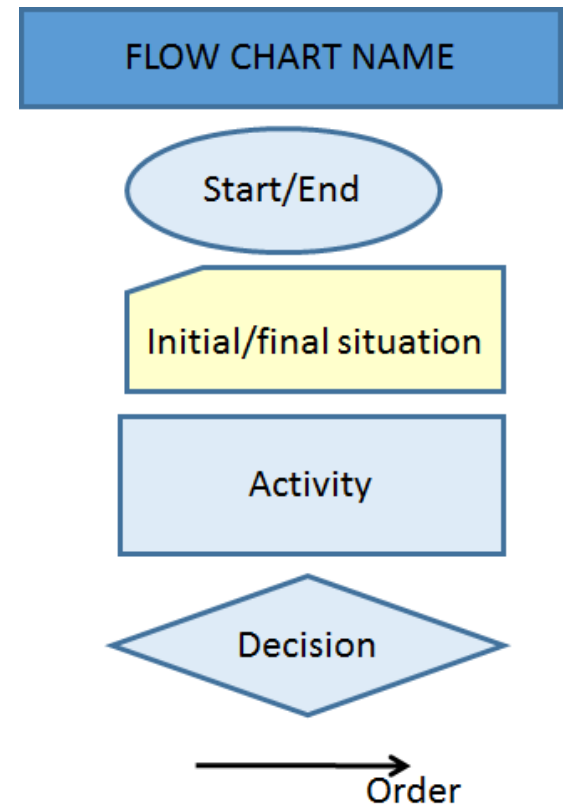- "Flies like an arrow" is the name of an American Indian. Time him
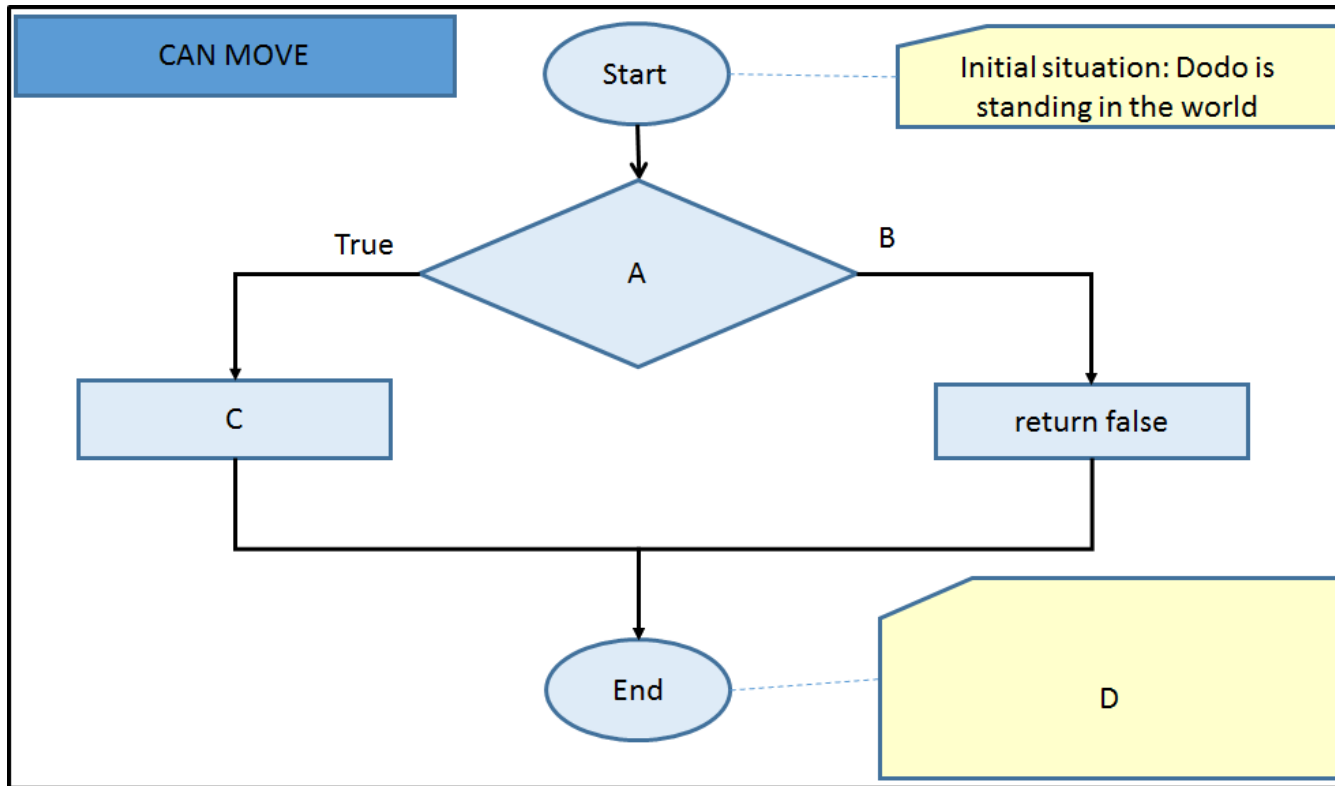
Flowchart: visualize algorithm

Flowchart:
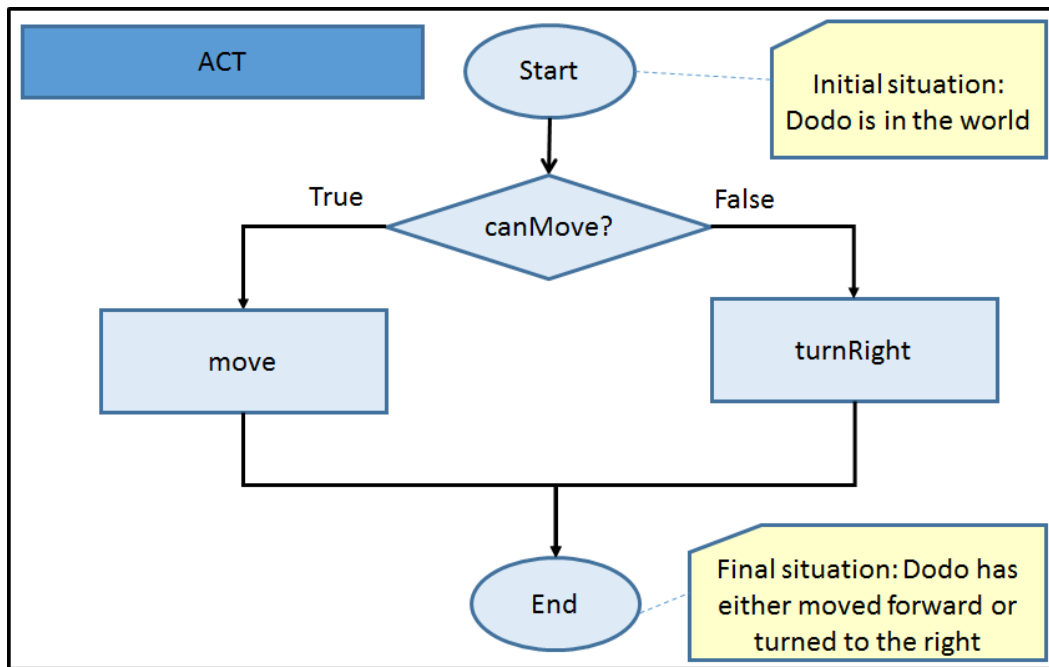
Key:

Exercise: Flowchart

❑ Flowchart visualizing the algorithm boolean canMove():

❑ What are A, B, C and D?

Compiling, running & testing code

❑ Need compiling:

❑ Compile:
- ■ Fix error messages

❑ Test:
- ■ Right-click on object & choose method to test
- ■ Check if works as expected
  - ■ i.e. Compare initial and final situation with flowchart
- ■ Not OK? Check if code is same as flowchart
- ■ Still not OK? Check if flowchart same as algorithm

Wrapping up

Save your work! Discuss how/when to finish off and who will turn it in.

Homework:

- Download scenario 2 at http://www.cs.ru.nl/~S.Smetsers/Greenfoot/Dominicus/
- Finish Assignment 1:
  - 5.5 until and incl 5.10
  - Diagnostic test 6.1 and check own answers 6.2
- Assignment 2: Until and incl 5.2 (lots of reading)
- **Hand via email to sjaaksm@live.com**
- **ALSO**: 5.1.2 ex 4 (or photo or scan)