



# Algorithmic Thinking and Structured Programming (in Greenfoot)

---

Teachers:

Renske Smetsers-Weeda

Sjaak Smetsers



# Today's Lesson plan (5)

---

Jan 8<sup>th</sup>

- 10 min Looking back
  - Quiz: graded, will be discussed next week
  - What did we learn before/during vacation?
  
- Theory for assignment 4
- Work on assignment 4
  
  
- 10 min Wrapping up



# Retrospective assignment 3

---

- ❑ Nesting
- ❑ Optimization
- ❑ Submethods
- ❑ Run as an 'Act' loop
- ❑ (Greenfoot.stop)
- ❑ Generic solutions

# Retrospective: Optimization

- Redundancy: why do we care?

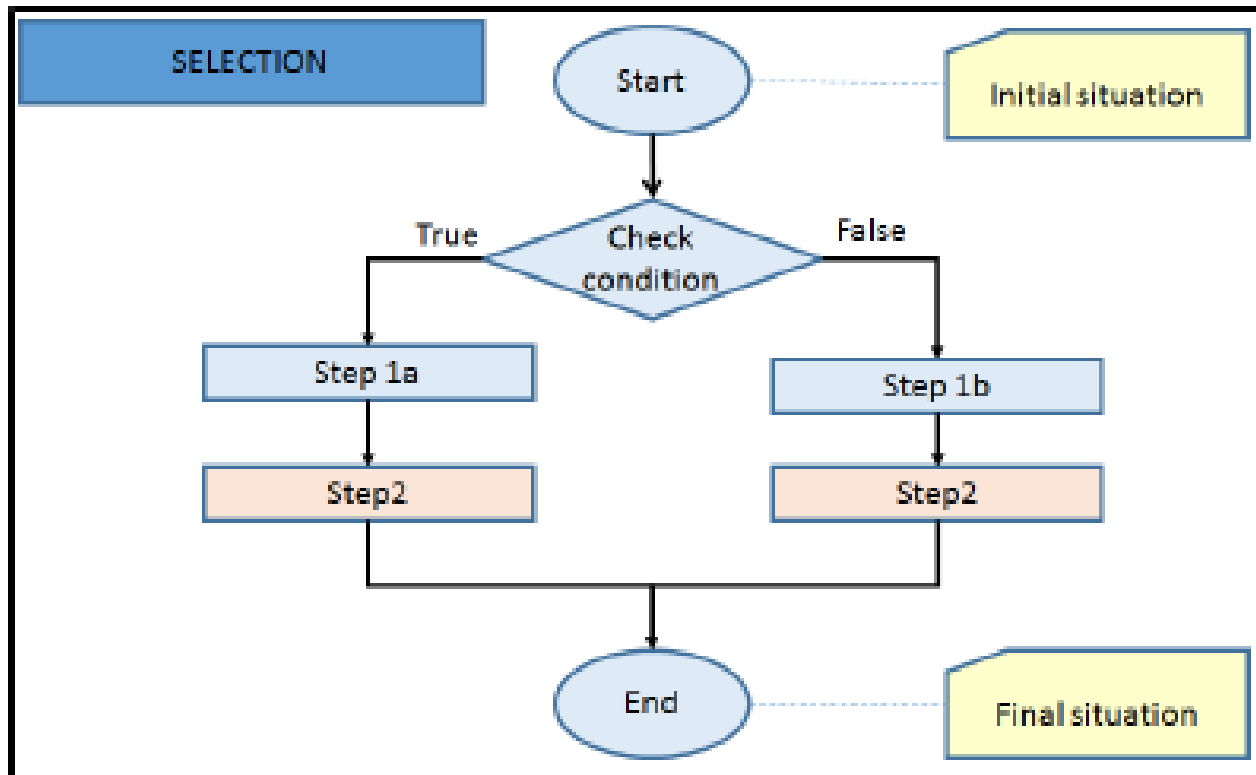
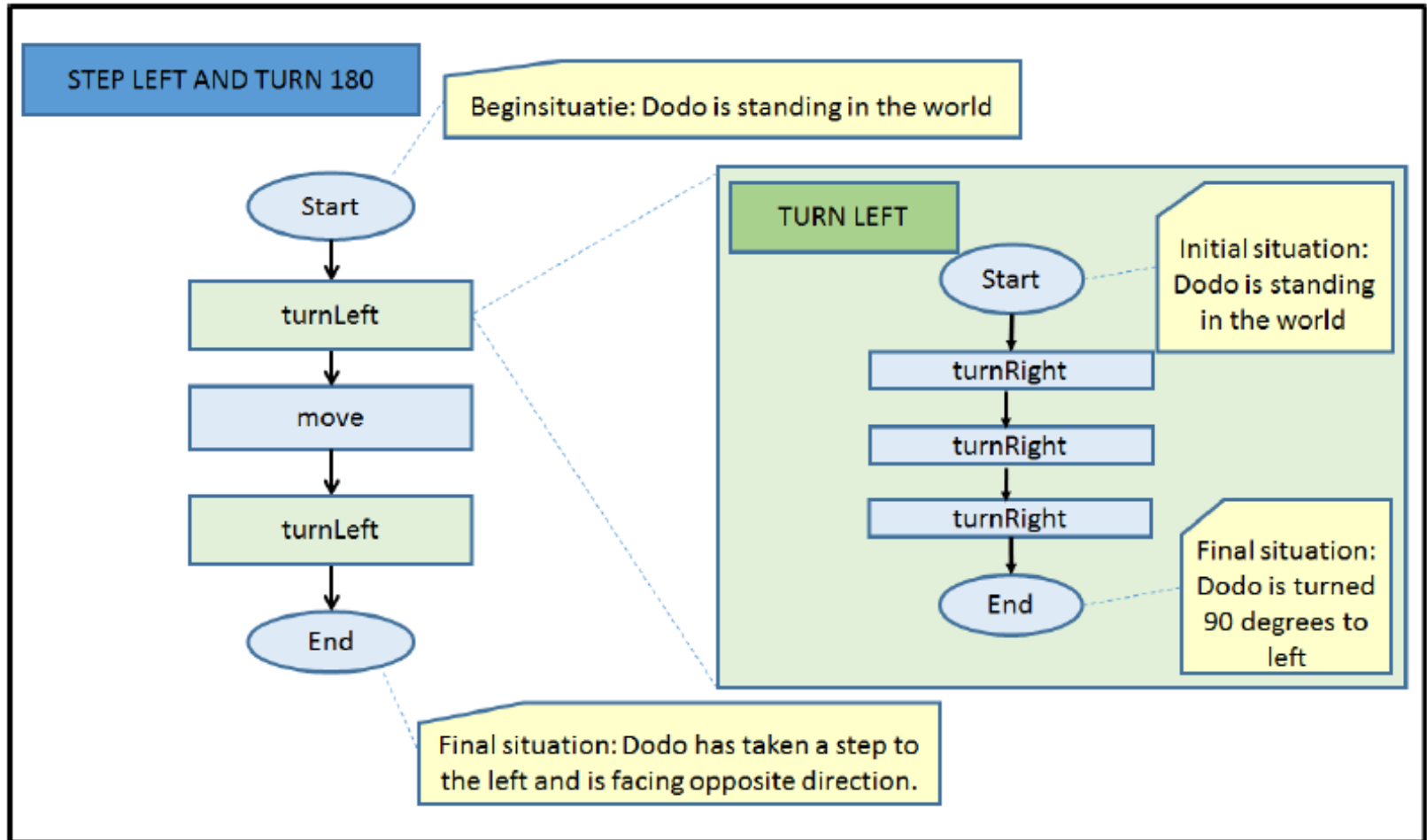


Figure 2: Flowchart with a redundant activity

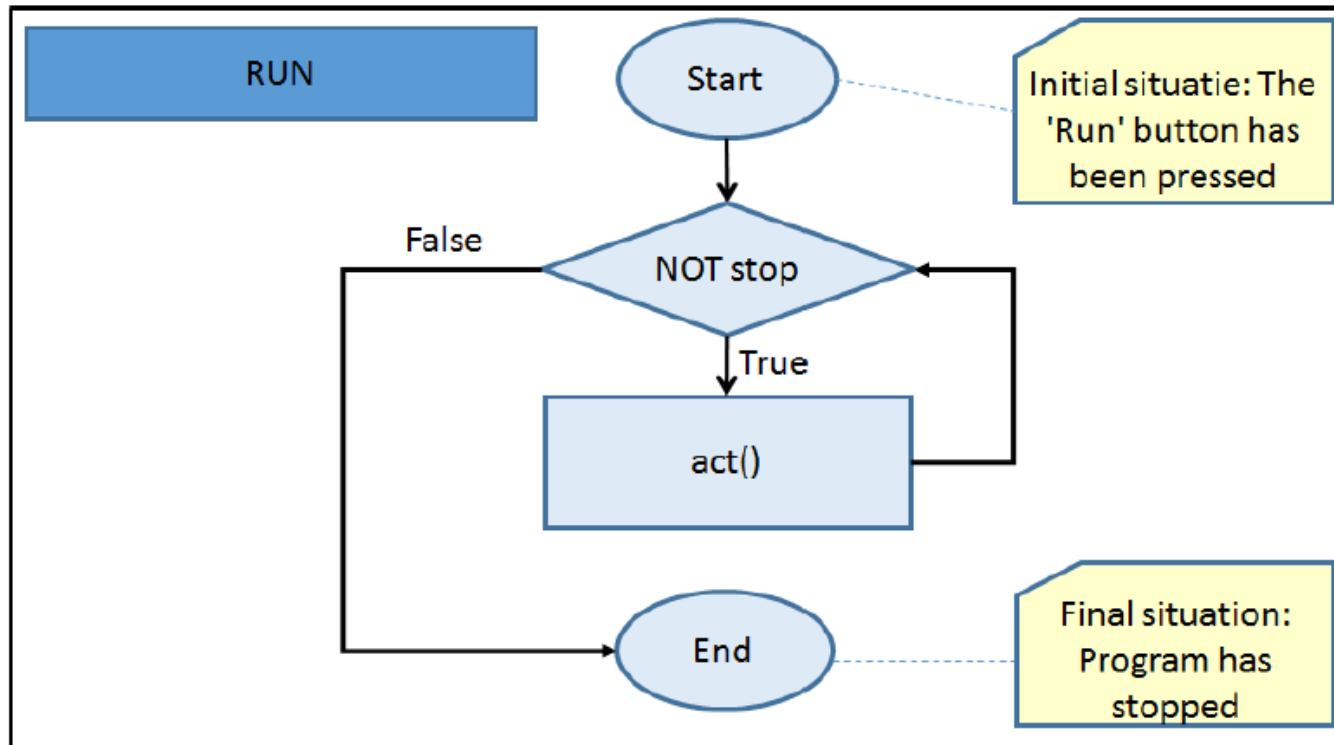
# Retrospective: submethods

- Submethods: why do we bother?




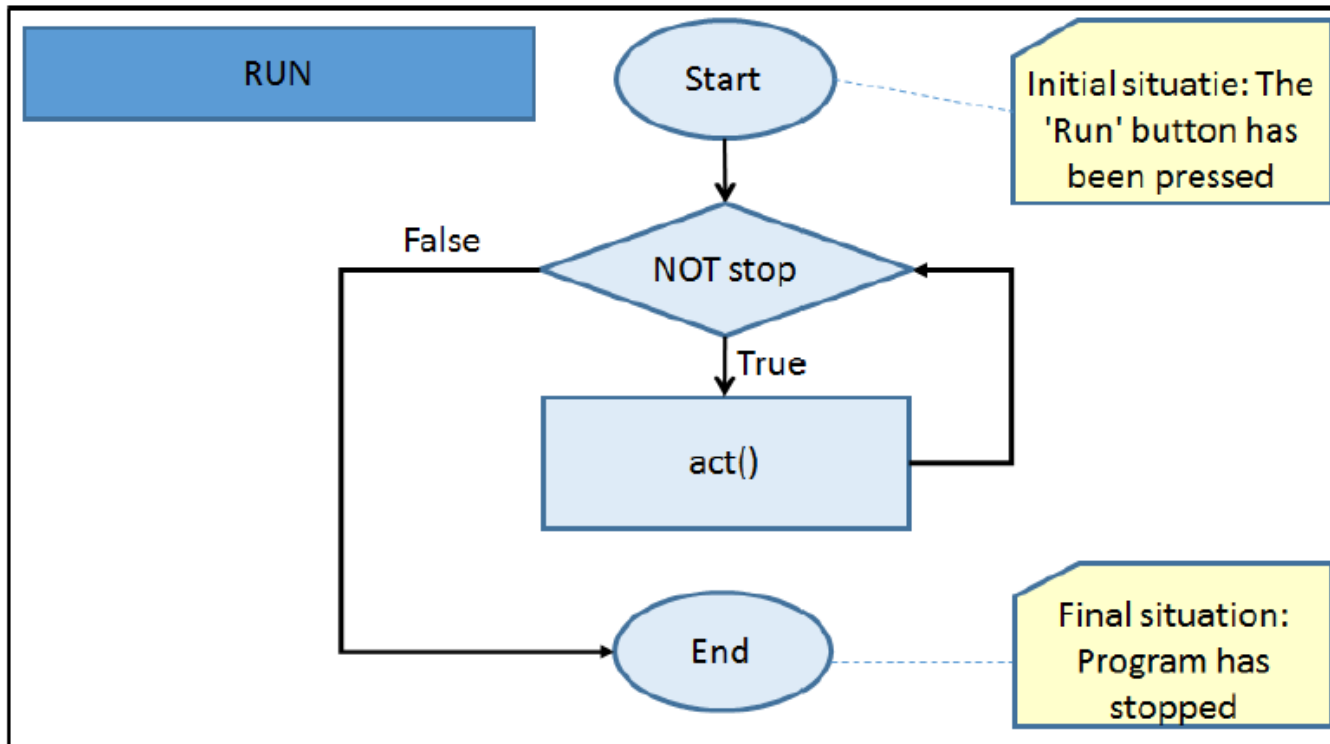
# Retrospective: Run

- Greenfoot Run: a while loop
- When does this stop?



# Retrospective: Run

- Greenfoot Run: a while loop
- Only stops if:
  - User presses  Pause
  - Calling `Greenfoot.stop( )`; in the code





# Retrospective: Generic solutions

---





# Conditionals

---

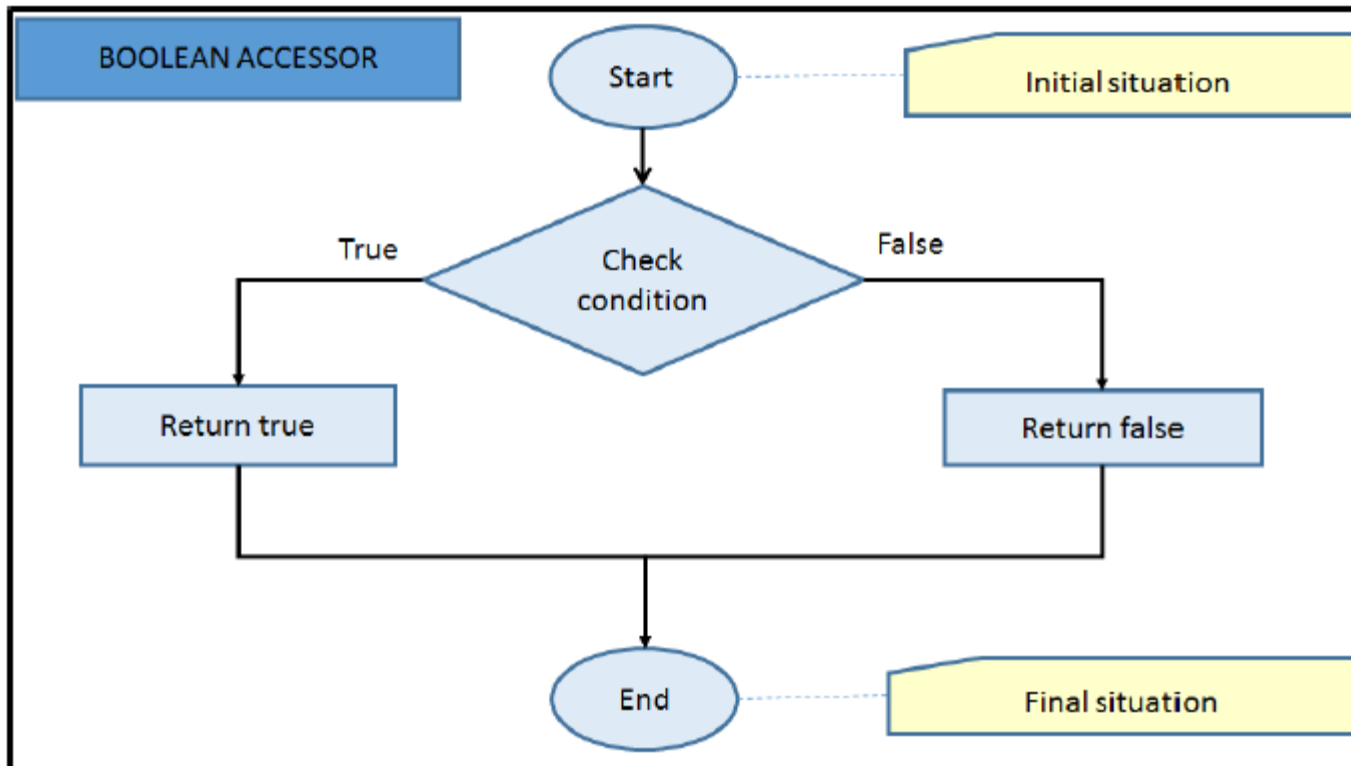
- Conditionals:

- boolean methods
- logical operators: ||, &&, !
  - || means OR
    - fenceAhead ( ) || borderAhead ( )
  - && means AND
    - canMove ( ) && eggAhead ( )
  - ! Means NOT
    - ! eggAhead ( )

# Return Reminder

## Return:

- After a return, End follows immediately
- No more steps executed after a return





# Jump Joyfully

---

Example with:

- ❑ Nested if-then-else
- ❑ Using return statements
- ❑ Complex Boolean statements



# Jump Joyfully

---

Example with:

- ❑ Nested if-then-else
- ❑ Using return statements
- ❑ Complex Boolean statements

# Jump up and down joyfully

If Mimi has a nest on each side,  
she jumps up and down joyfully



MyDodo methods:

```
boolean nestAhead ( ) // returns true if nest in cell ahead
void turnLeft ( ) // turns 90 degrees clockwise
void turnRight ( ) // turns 90 degrees counterclockwise
void move ( ) // step forward if possible
```

## Strategy:

Sketch a **high-level** flowchart for jumpJoyfully

Tip:

- First **assume** nestBehind and jumpUpAndDown exist
- Then: design, implement & test them separately

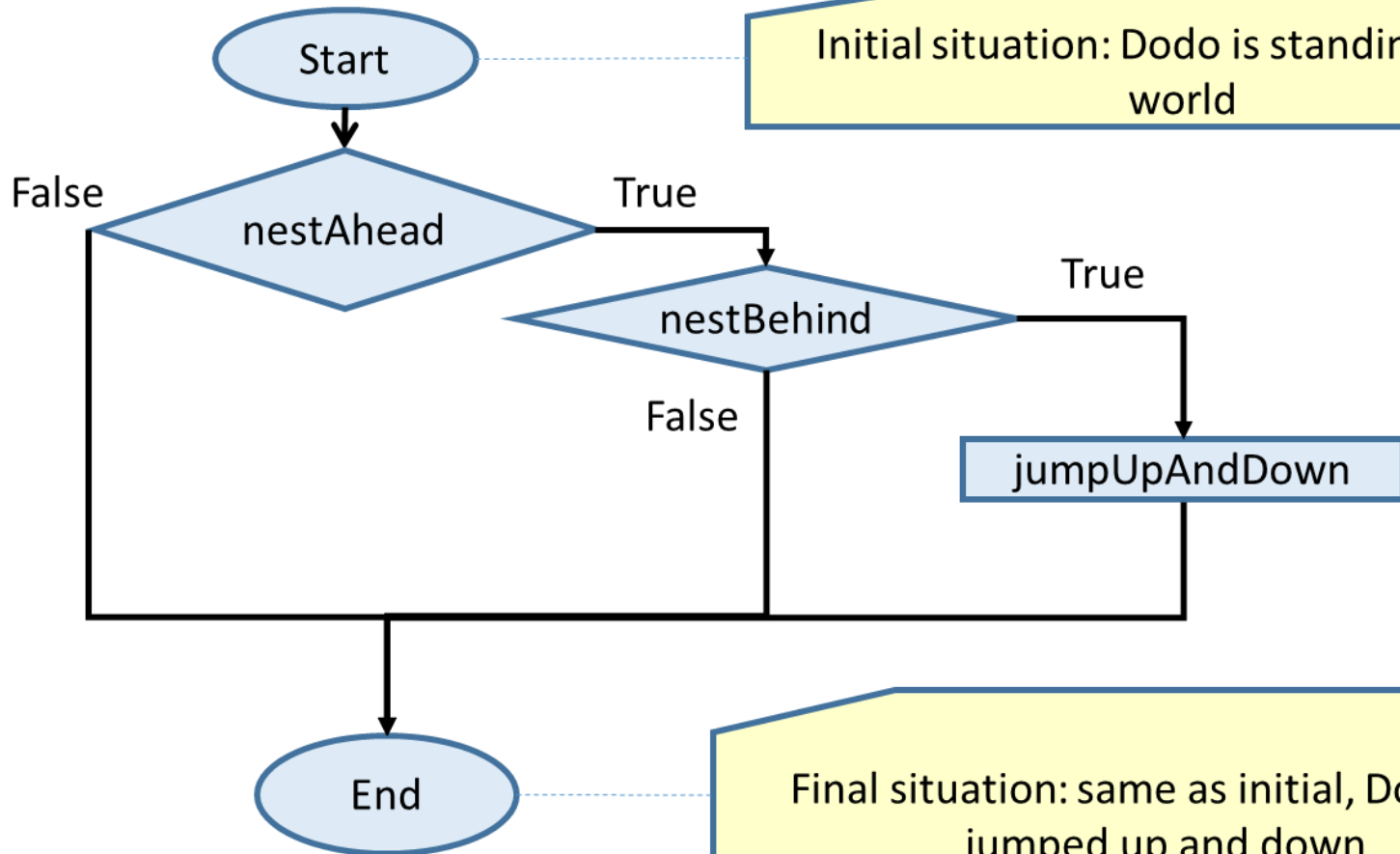


# Sketch high-level flowchart

---

# Test using: Nested if..then..else

JUMP JOYFULLY

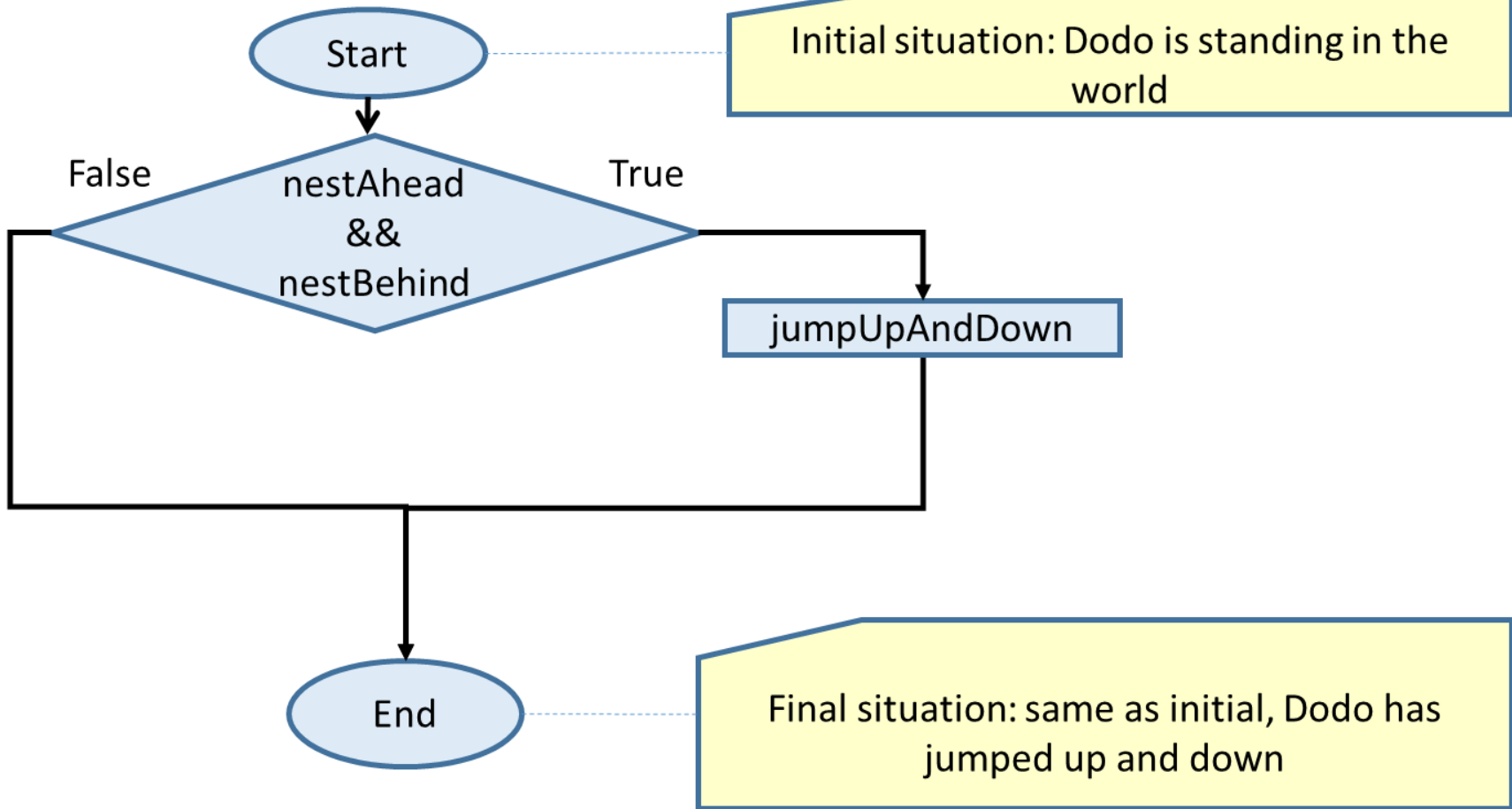


Initial situation: Dodo is standing in the world

Final situation: same as initial, Dodo has jumped up and down

# Test using: conjugated Boolean &&

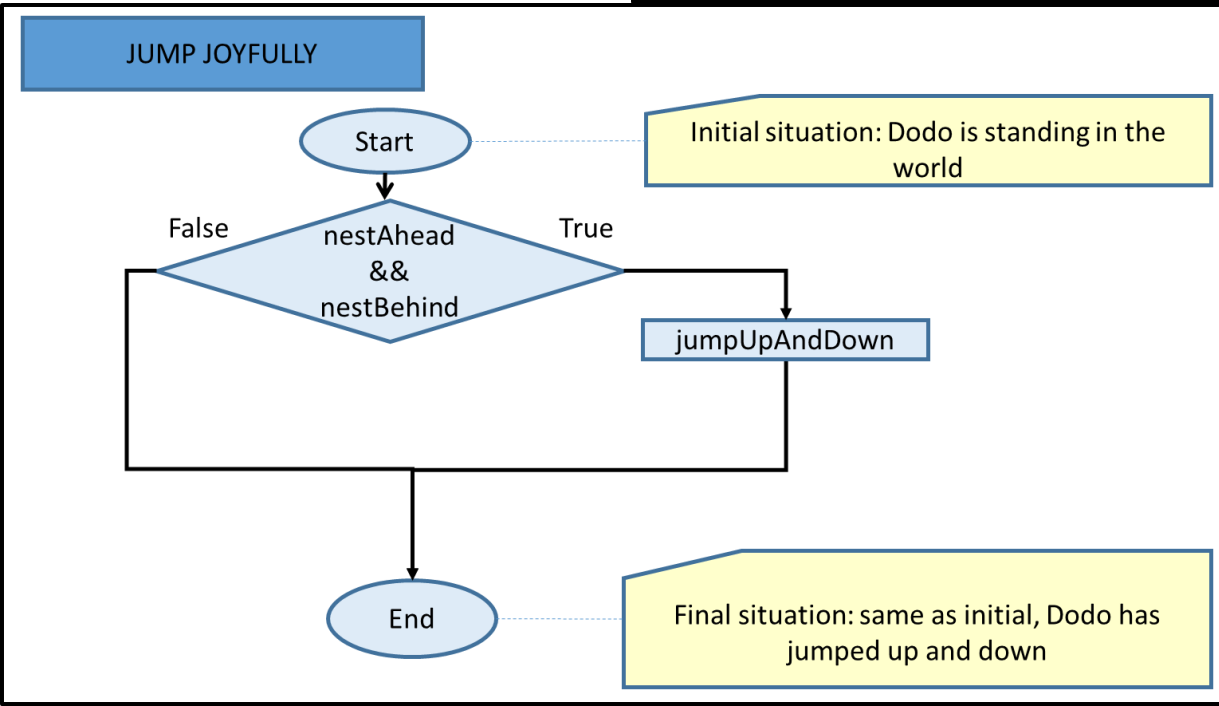
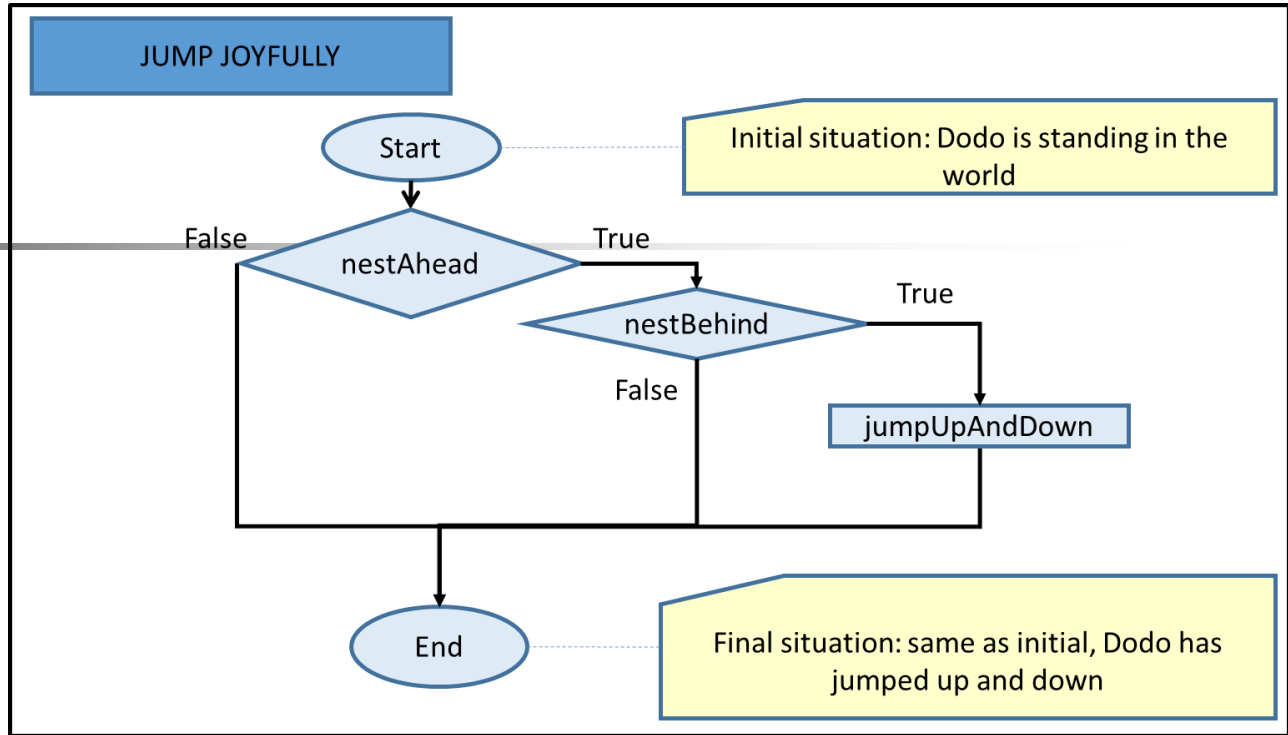
## JUMP JOYFULLY





# Compare:

- Which do you prefer?
- Why?





# Now: design nestOnLeft

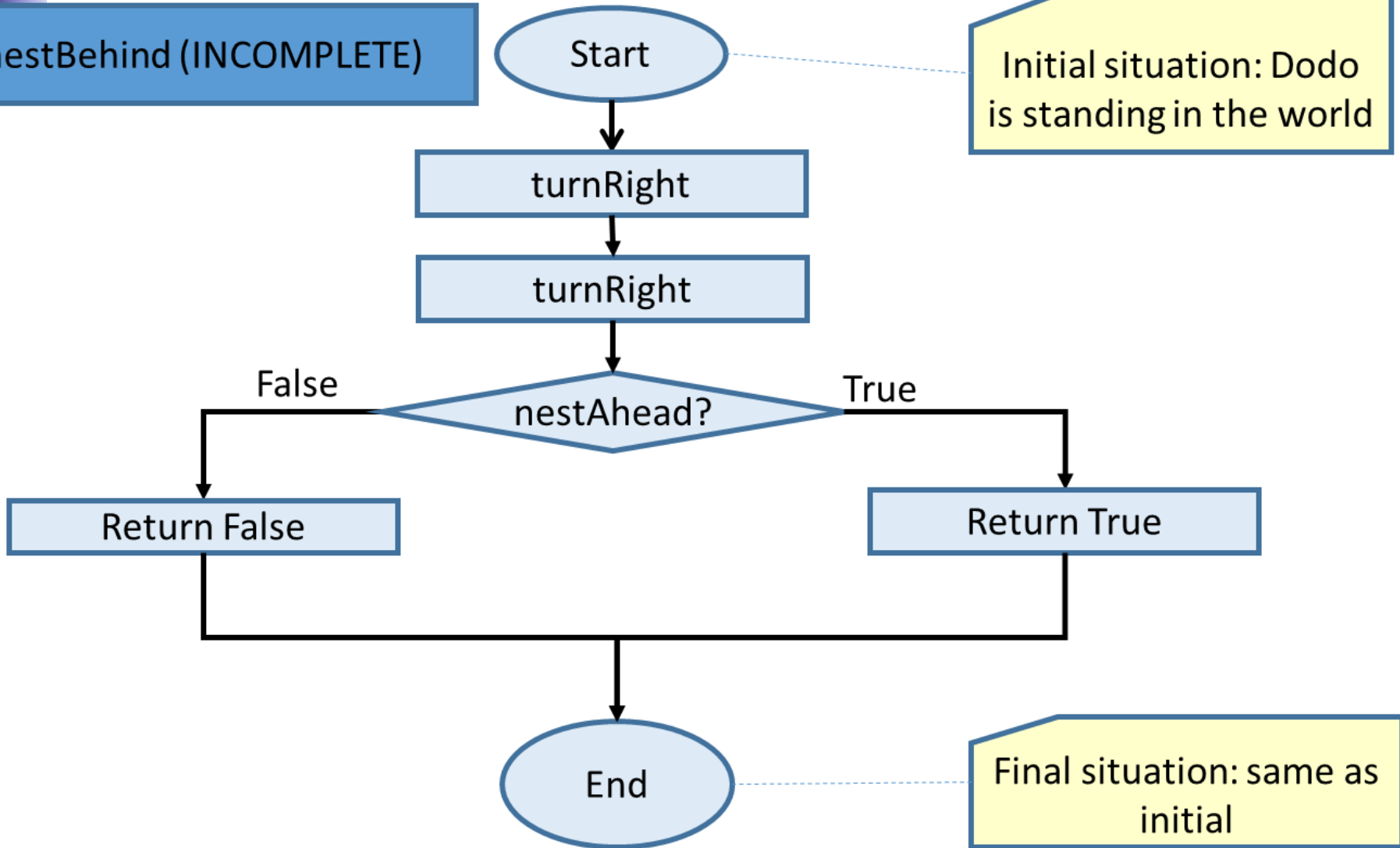
---

- ✓ Finished high-level flowchart
- .. Now the Boolean nestBehind()

Draw the flowchart

# Boolean nestBehind

nestBehind (INCOMPLETE)

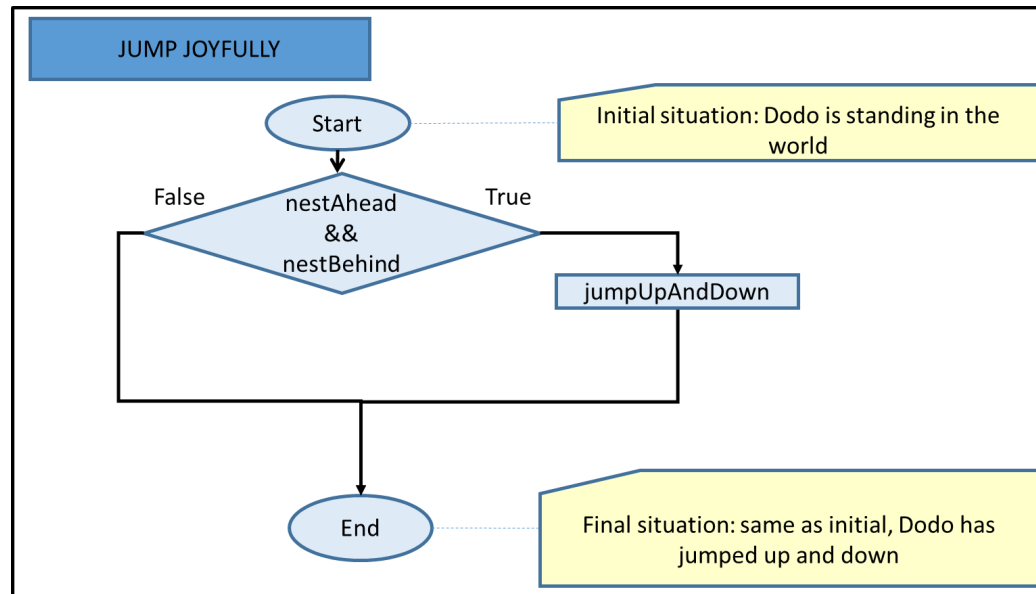


# Now: test nestOnLeft ( )

- ✓ Finished high-level flowchart
- ✓ Designed nestBehind( )
- ... now test nestBehind( )

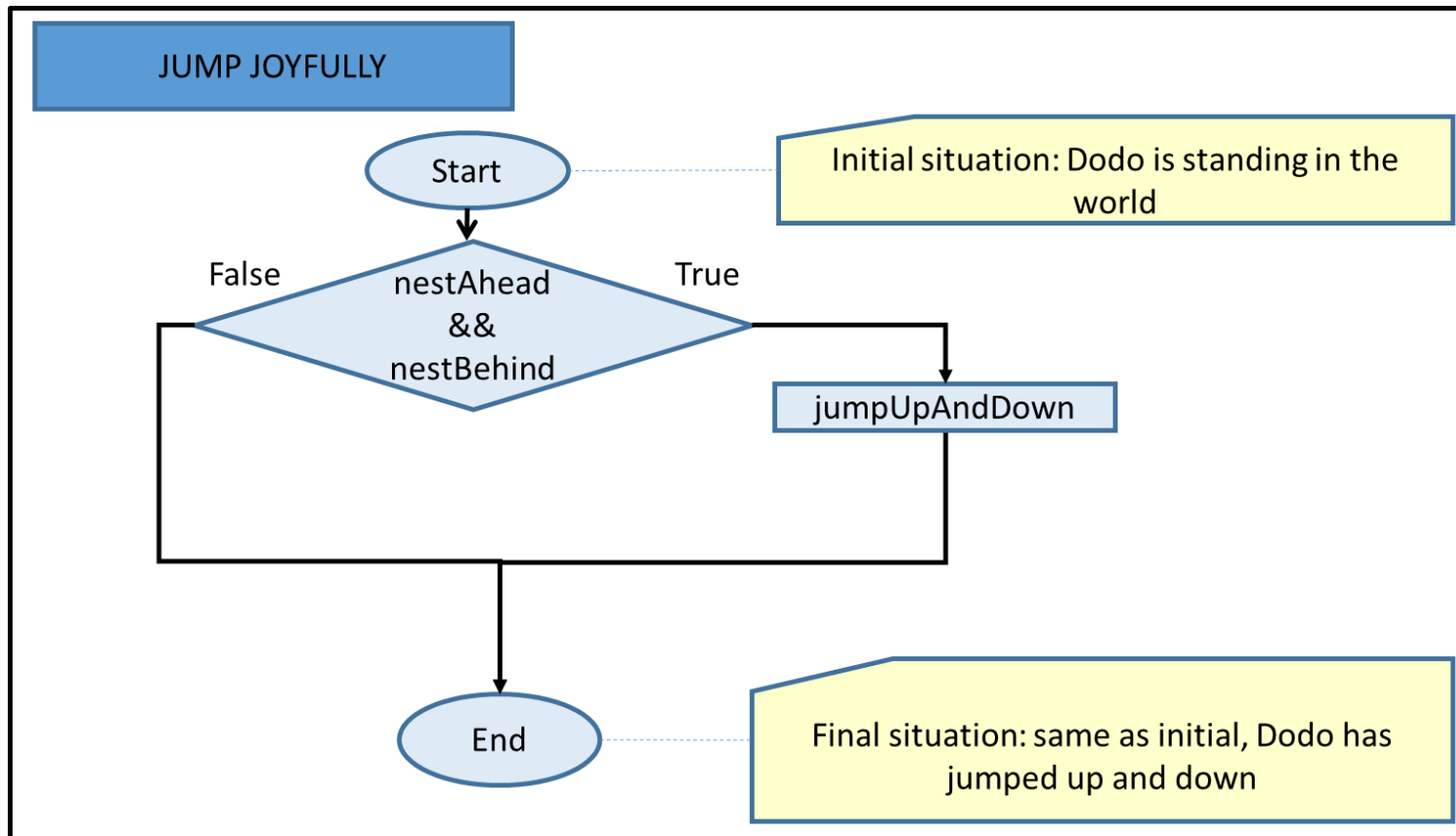
What are we doing:

Testing small pieces before we use them!



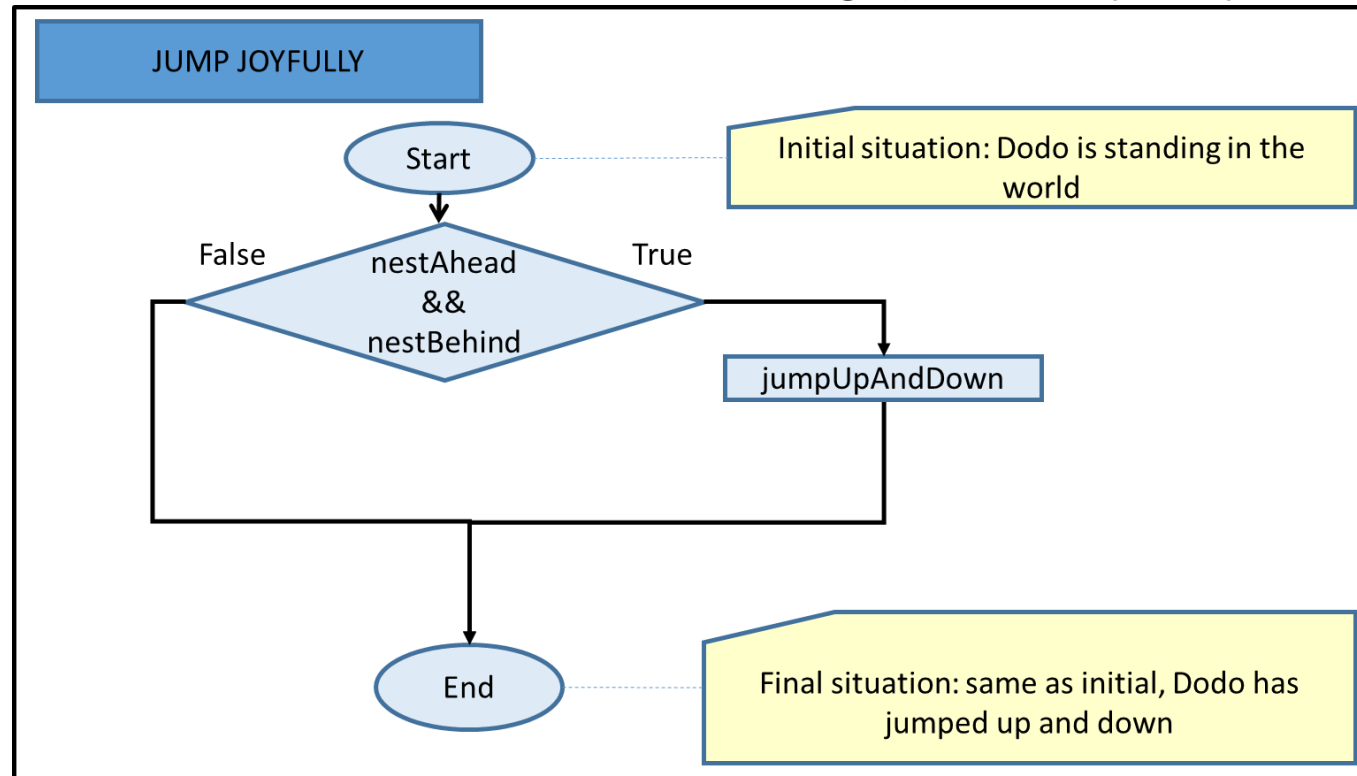
# Now: design and test jumpUpAndDown

- ✓ Finished high-level flowchart
- ✓ Designed and tested nestBehind( )
- ... now design and test jumpUpAndDown ( )



# Now: test the whole thing

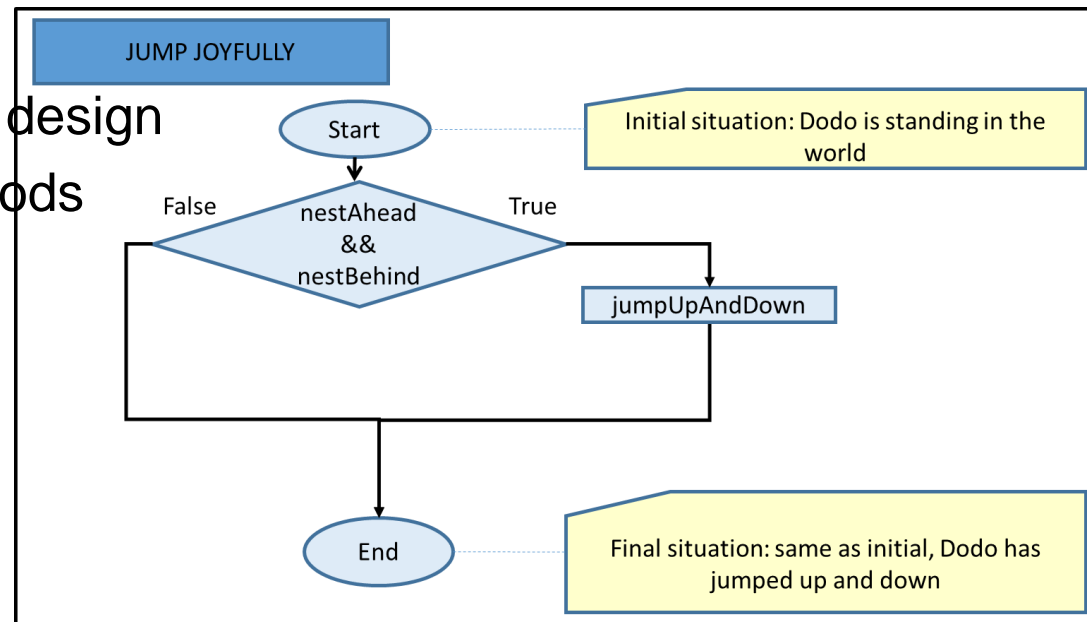
- ✓ Finished high-level flowchart
- ✓ Designed and tested nestBehind( )
- ✓ Designed and tested jumpUpAndDown ( )
- .. Now combine parts and test whole thing: jumpJoyfully



# Now: enjoy and be proud

- ✓ Finished high-level flowchart
- ✓ Designed and tested nestBehind( )
- ✓ Designed and tested jumpUpAndDown ( )
- ✓ Combined parts and tested whole thing: jumpJoyfully

So, first start with high level design  
Then implement small methods  
Then test the whole thing





# What did we just practice?

---

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





# 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



# Work on Assignment 4/5

---

[movie](#)





# Wrapping up

---

Homework for Wednesday 8:30 April 6<sup>th</sup>:

□ Assignment 5:

■ **UNTIL AND INCL 5.1.5**

■ **Hand via email to [sjaaksm@live.com](mailto:sjaaksm@live.com)**

□ Reflection/Evaluation: Tips & Tops