



Algorithmic Thinking and Structured Programming (in Greenfoot)

Teachers:

Renske Smetsers-Weeda

Sjaak Smetsers

Assignment 5: difficulties

- Write a method

```
int degreesNeededToTurnToFaceNorth( )
```

which returns how many degrees Mimi must turn (counterclockwise) to face North.

- Can you do think of an algorithm which does this without using an if...then...else statement?

```
public int degreesNeededToTurnToFaceNorth() {  
    int degreesNeeded = 0;  
    while ( getDirection( ) != Dodo.NORTH ) {  
        turnLeft();  
        degreesNeeded = degreesNeeded + 1;  
    }  
    return degreesNeeded;  
}
```

What is wrong?

Answer: it changes Mimi's state



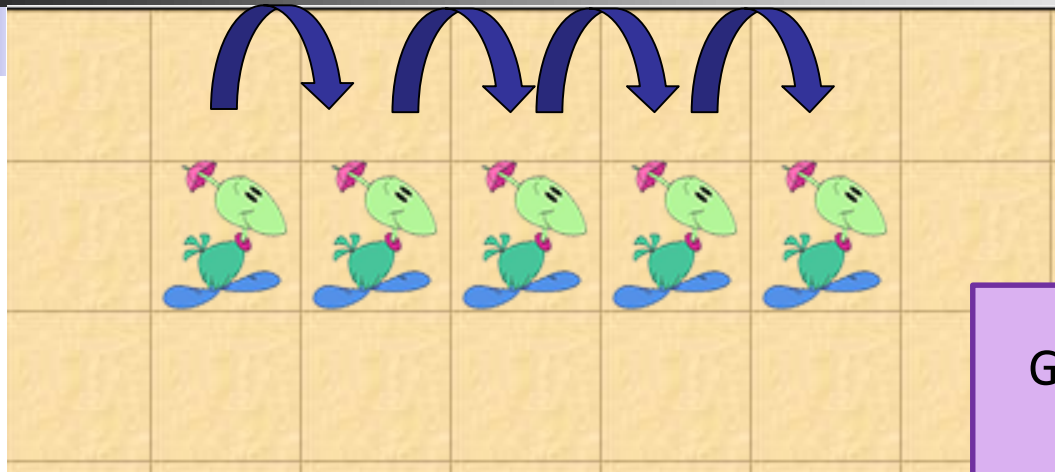
Assignment 5: solution

```
public int degreesNeededToTurnToFaceNorth() {  
    return getDirection() * 90;  
}
```

Repetition



Example: Mimi moves random times



GetRandomNumber(N) will give a random number between 0 and N (N not included)

Sketch how would you make Mimi move forward a random number of 0-9 cells (**jumpRandomly** method) using:

- **getRandomNumber(10)**
- a variable to remember how many moves must be made
- Dodo's **move()** method

Intermezzo: rolling dice



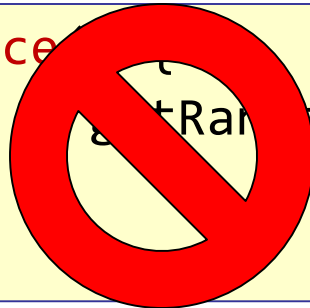
- Write a method that *simulates* the roll of a pair of dice
 - it should return a (random) value which is the sum of the outcomes of each die.
 - use `int getRandomNumber(int limit)`

```
public int roll2Dice() {
```

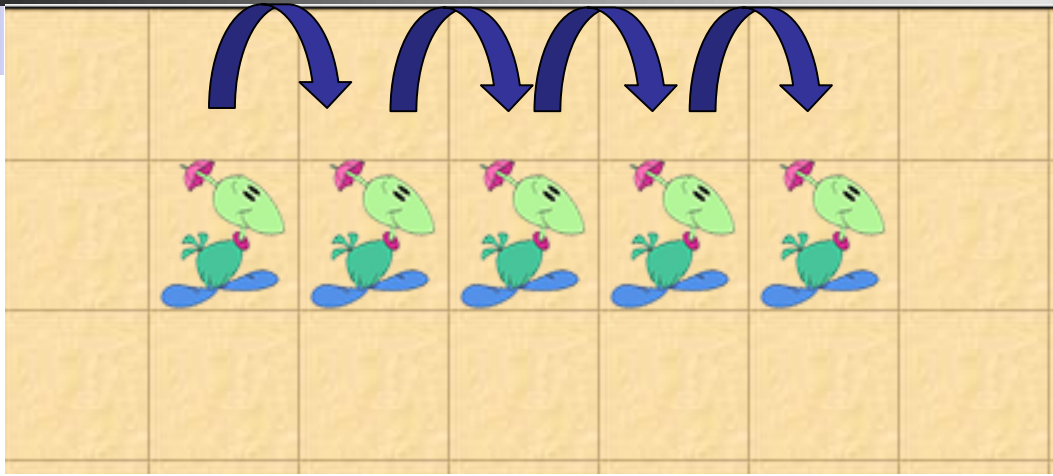
```
}
```

- What is the difference with

```
public int roll2Dice() {  
    int twodice = 2 * getRandomNumber( 11 );  
    return twodice;  
}
```



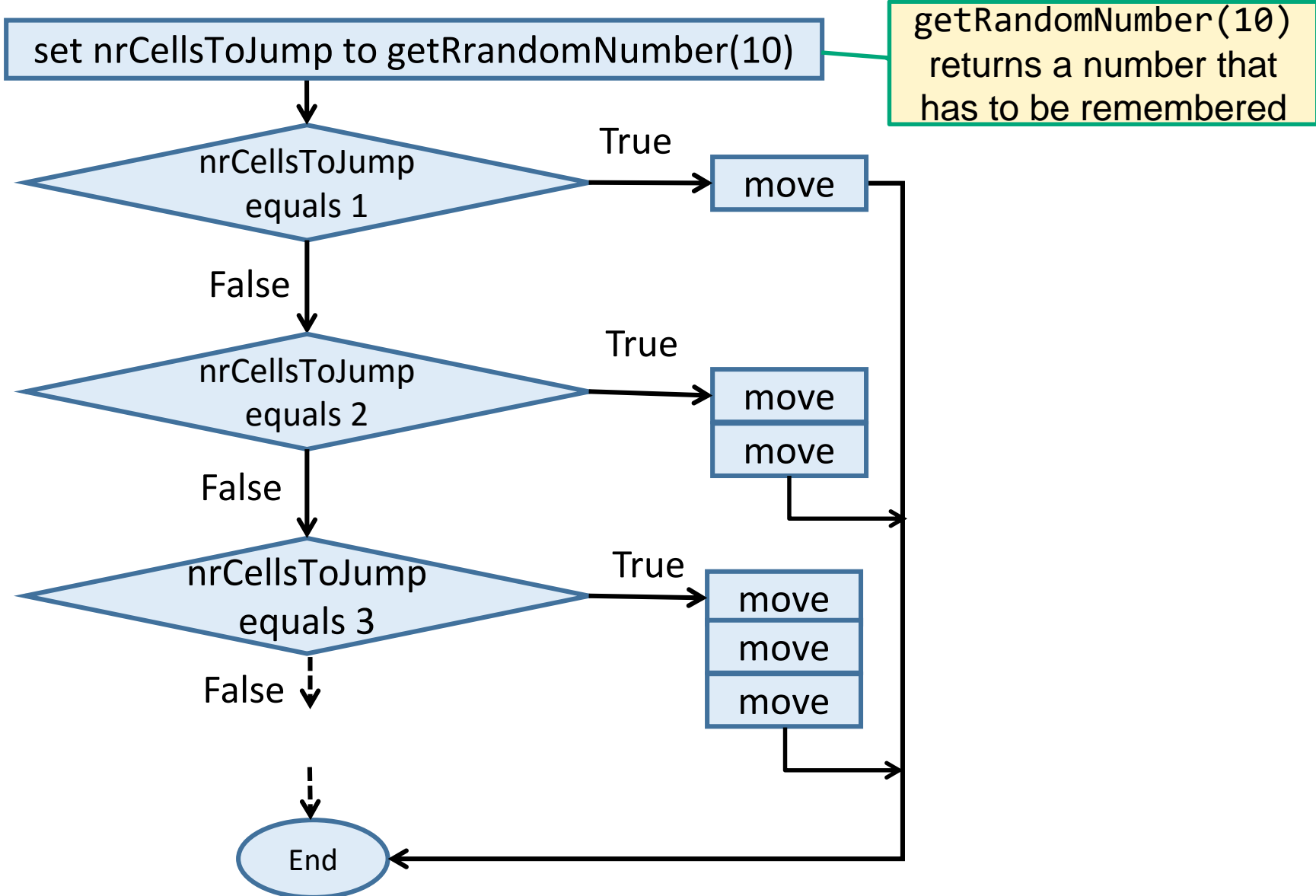
Example: Mimi moves random times



Sketch how would you make Mimi move forward a random number of 0-9 cells (`jumpRandomly` method) using:

- `getRandomNumber(10)`
- a variable to remember how many moves must be made
- Dodo's `move()` method

Nested if ... then ... else statements



Move a random number of times

We use a (local) `int` variable with name `nrCellsToJump` to store the random number

`getRandomNumber(10)` returns a number that has to be remembered

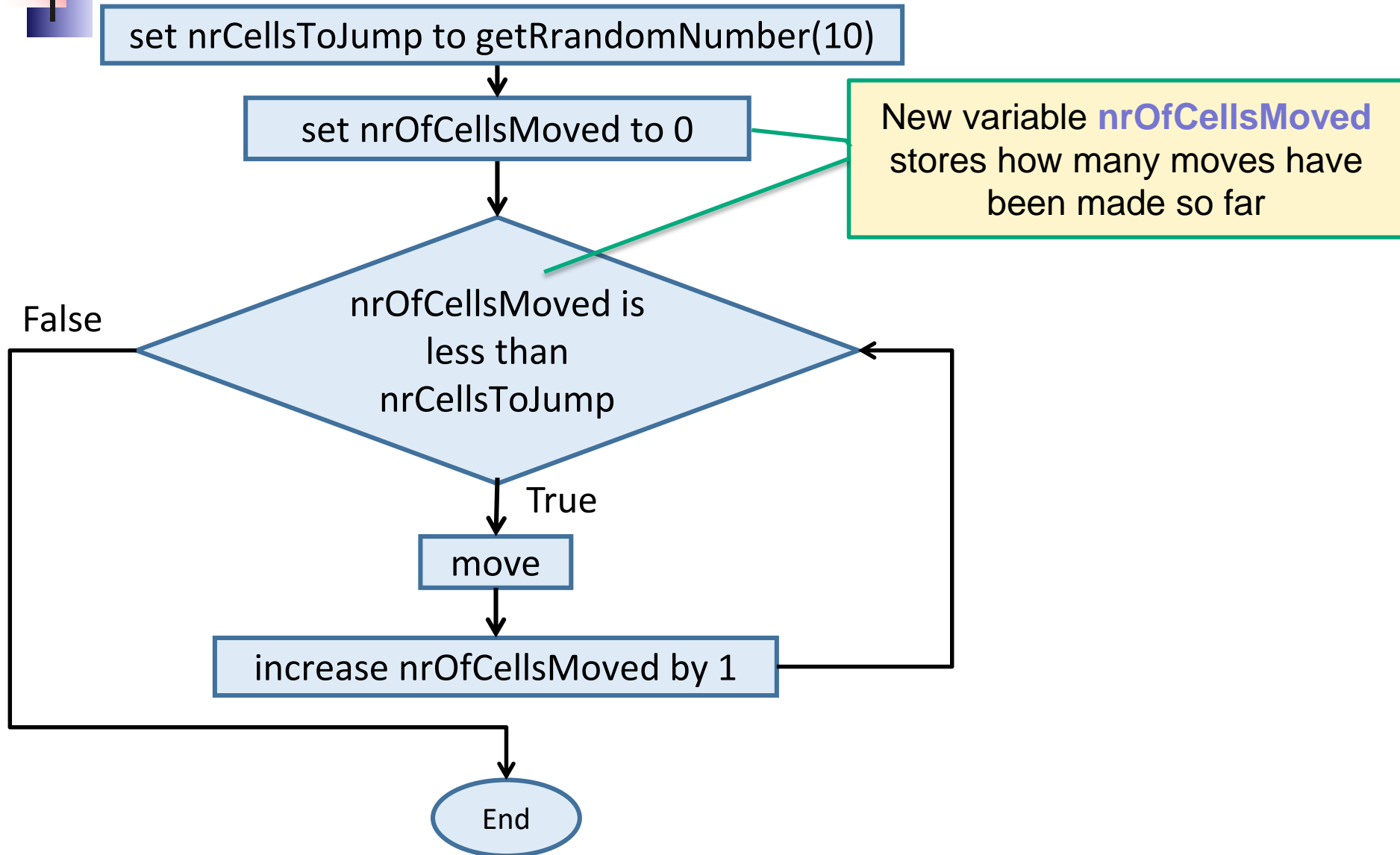
```
public void jumpRandomly () {  
    int nrCellsToJump = getRandomNumber(10);  
    if ( nrCellsToJump == 1 ){  
        move();  
    } else if ( nrCellsToJump == 2 ){  
        move();  
        move();  
    } else if ( nrCellsToJump == 3 ){  
        move();  
        move();  
        move();  
    }  
    ...  
}
```

Bah!

Mind the difference:
= (assignment)
== (comparison)



... alternative with while and counter



... alternative with counter and while

```
public void jumpRandomly () {  
    int nrCellsToJump = Greenfoot.getRandomNumber(10);  
    int nrCellsMoved = 0;  
    while ( nrCellsMoved < nrCellsToJump ){  
        move ();  
        nrCellsMoved = nrCellsMoved + 1;  
    }  
}
```

To store how many moves have been made so far.

The current value of `nrCellsMoved`...

... incremented and assigned to `nrCellsMoved`



Difficult question

- Does it make any difference if we write:

```
public void jumpRandomly () {  
    int nrCellsMoved = 0;  
    while ( nrCellsMoved < Greenfoot.getRandomNumber(10) ){  
        move ();  
        nrCellsMoved = nrCellsMoved + 1;  
    }  
}
```

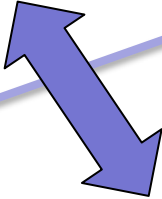
- Answer: not all distances are equally likely.





Comparing with(out) counter & while

```
public void jumpRandomly () {  
    int nrCellsToJump = Greenfoot.getRandomNumber(10);  
    if (nrCellsToJump == 1){  
        move();  
    } else if (nrCellsToJump == 2){  
        move();  
        move();  
    } else if (nrCellsToJump == 3){  
        move();  
        move();  
        move();  
    }  
    ...  
}
```



```
public void jumpRandomly () {  
    int nrCellsToJump = Greenfoot.getRandomNumber(10);  
    int nrCellsMoved = 0;  
    while ( nrCellsMoved < nrCellsToJump ) {  
        move ();  
        nrCellsMoved = nrCellsMoved + 1;  
    }  
}
```



Topics for assignment 6

- ❑ Constructors, instance variables
- ❑ Access modifiers: private, public (protected):
information hiding
- ❑ Getter/setter methods



Variable Scope (lifetime)

- What happens to variable **nrCellsMoved** after this method?

```
public void jumpRandomly () {  
    int nrCellsToJump = Greenfoot.getRandomNumber(10);  
    int nrCellsMoved = 0;  
    while ( nrCellsMoved < nrCellsToJump ) {  
        move ();  
        nrCellsMoved = nrCellsMoved + 1;  
    }  
}
```



Variable Scope (lifetime)

- ❑ After the method, **nrCellsMoved** is destroyed!
- ❑ So we can't use **nrCellsMoved** in another method....

```
public void jumpRandomly () {  
    int nrCellsToJump = Greenfoot.getRandomNumber(10);  
    int nrCellsMoved = 0;  
    while ( nrCellsMoved < nrCellsToJump ) {  
        move ();  
        nrCellsMoved = nrCellsMoved + 1;  
    }  
}
```

- ❑ Unless, we use **instance variables**.



Instance variables

- To store (remember) values for longer periods of time
 - Outside of method:
 - 'normal' method variables lose their values
 - Use instance variables when using same variable by two different methods
 - When act is called again:
 - Only instance variables are stored
 - All other values are lost
- You can even 'inspect' object value at all times

How Objects are Created

```
new MyDodo ( );
```

Java creates object in memory

initialize state of object by invoking constructor

```
// constructor's job is to  
// initialize a new object  
public MyDodo( ) { ... }
```

The Constructor

- When Java creates a new object, it calls the class's **constructor**.
- Instance variables are initialized.

The constructor has the **same name** as the class.

Instance variable

super() calls the constructor of Dodo.

```
public class MyDodo extends Dodo
{
    private int myNrOfEggsHatched;

    public MyDodo(int init_direction) {
        super ( init_direction );
        myNrOfEggsHatched = 0;
    }
    ...
}
```

Class code

```
public class MyDodo extends Dodo
```

```
{
```

```
/* DECLARATIES VAN ATTRIBUTEN */
```

```
private int myNrOfEggsHatched;
```

```
public MyDodo( int init_direction ) {
```

```
/* INITIALISATIE VAN ATTRIBUTEN */
```

```
myNrOfEggsHatched = 0;
```

```
}
```

```
/* METHODES VAN DE KLASSE */
```

```
public void act() {
```

```
}
```

```
}
```

Class header

Declaration of instance variables

Initialisation of instance variables

Class methods

Class code



Visibility of variables / methods

Visibility	Explanation
public	accessible from outside the class
private	only accessible from within the class itself
protected	only accessible from within the class or its subclasses



Information hiding

- Rule: make instance variables **private**

Visibility	Explanation
public	accessible from outside the class
private	only accessible from within the class itself
protected	only accessible from within the class or its subclasses

- This means: other objects can't reach it!
- Solution: create (if needed)
 - **public getter** method
 - **public setter** method



Getter method

Visibility	Explanation
public	accessible from outside the class
private	only accessible from within the class itself
protected	only accessible from within the class or its subclasses

int myAge is private, no one needs to know... so...

```
private int myAge;
```

But... if myAge needs to **asked** for a (real) reason:

```
public int getMyAge( ) {  
    if ( youHavePermissionToKnow ( ) ){  
        return myAge( ) ;  
    } else {  
        return 0;  
    }  
}
```

To call (object Teacher) from another method, use:

```
Teacher.getMyAge()
```



Setter method

Visibility	Explanation
public	accessible from outside the class
private	only accessible from within the class itself
protected	only accessible from within the class or its subclasses

String myPassword is private, so:

```
private string myPassword;
```

But... if myPassword needs to be **changed** for a (real) reason:

```
public void setMyPassword ( string newPassword ) {  
    myPassword = newPassword;  
}
```

How to call (object Teacher) from another method, call:

```
Teacher.setMyPassword ( "doorbell" );
```




Wrapping up

Homework for Wednesday 8:30 May 11th:

□ Assignment 6:

■ **FINISH assignment 6 up to and incl 5.3**

(you may advance if you wish

-> less homework next time)

■ email Java code and 'IN'-answers to
`sjaaksm@live.com`