# Opdracht 5: Uitwerkingen

– Objectgeoriënteerd Programmeren in Greenfoot –

Renske Smetsers-Weeda & Sjaak Smetsers

mei, 2015

## 1 Voorbeelduitwerkingen van de opgaven

- Variabelen tracen
  1. `aantalEierenGevonden` == 4
  2. `aantalEierenGevonden` == 6
  3. `aantalEierenGevonden` == 0
  4. `getal1` == 6 en `getal2` == 10
  5. `getal1` == 4 en `getal2` == 5
  6. `getal1` == 4 en `getal2` == 16
  7. `getal1` == 4 en `getal2` == 4
  8. `getal1` == 6 en `getal2` == 3
  9. `getal1` == 12 en `getal2` == 9
  10. `getal` == 11
  11. `getal` == 9
  12. `getal1` == 3 en `getal2` == 6. Verwisselen. De `getalTemp` is alleen maar tijdelijk nodig om het verwissellen van de waarden mogelijk te maken.

- Draai naar het oosten

| **Kijkrichting** | **Resultaat `int getDirection()`** | **Waarde `int myDirection` bij 'Inspect'** |
|---|---|---|
| North | 0 | 0 |
| East | 1 | 1 |
| South | 2 | 2 |
| West | 3 | 3 |

```
/*
* Deze methode zorgt ervoor dat Mimi naar het oosten kijkt.
* Werkt onafhankelijk van de begin kijkrichting
*/

public void faceEast(){
    if( getDirection() == 0){ // facing north
        turnRight();
    }else{
        if( getDirection() == 2){ //facing south
            turnLeft();
        }else{
            if( getDirection() == 3){//facing west
                turnRight();
                turnRight();
            }else{ // facing east, do nothing
            }
        }
    }

}
```

- Hoeveel graden draaien?

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

- Draai en stap

```
public boolean facingCorrectDirection( int direction ){
    if( direction == getDirection() ){
        return true;
    } else {
        return false;
    }
}

public void turnToDirectionAndMove( int direction ){
    while(! facingCorrectDirection ( direction ) ){
        turnRight();
    }
    move();
}
```

- Ga naar een bepaalde locatie

```
public boolean validCoordinates (int coordX, int coordY) {
   World world =  getWorld();
   if (coordX <= world.getWidth()-1 && coordY <= world.getHeight()-1 && coordX>=0 && coordY>=0 ){
        return true;
    } else {
        return false;
    }
}

public boolean locationReached(int coordX, int coordY){
    if ( coordX == getX() && coordY == getY()){
        return true;
    } else {
        return false;
    }
}
```

methode `goToLocation` (mag ook zonder 'else if' en mbv nesting geschreven worden)

```
public void goToLocation( int coordX, int coordY){
    if( validCoordinates  ( coordX, coordY ) ){
        while (! locationReached( coordX, coordY ) ){
            if( coordX < getX() ){
                setDirection(WEST); //turn facing west
            }
            if(coordX > getX() ) {
                setDirection(EAST); //turn facing east
            }
            if( coordY < getY() ) {
                setDirection(NORTH); //turn facing north
            }
            if( coordY > getY() ) {
                setDirection(SOUTH); //turn facing sounth
```

```
            }
            move();

        }
    }else {
        showError("invalid coordinates");
    }
}
```

- While trace

  1. `aantalStappenGezet` == 8 en `aantalStappenOmTeZetten` == 8 en `move()` wordt 8 keer aangeroepen.
  2. `counter` == 7 en `aantalStappenOmTeZetten` == 6. `move()` wordt 7 keer aangeroepen. De code is dus niet goed.
  3. `counter` == 4 en `aantalStappenOmTeZetten` == 4. `move()` wordt 4 keer aangeroepen. De code is dus wel goed.
  4.

| Aantal **`while`**loops uitgevoerd | Waarde van `getal1` | Waarde van `getal2` |
|---|---|---|
| 0 | 2 | 5 |
| 1 | 3 | 5 |
| 2 | 4 | 5 |
| 3 | 5 | 5 |
| 4 | 6 | 5 |

  5.
```
int getal1 = 10;
int getal2 = 8;

while( getal1 >= getal2 ){
    move();
    getal1 --;
}
```

  OF

```
int getal1 = 10;
int getal2 = 8;

while( getal1 > getal2 -1 ){
    move();
    getal1 --;
}
```

  6. Jump

- Aantal stappen tot het einde van de wereld

```
public int walkToWorldEdgeAndCountSteps(){
    int stepsTaken=0;
    while(!borderAhead() && canMove() ){
        stepsTaken++;
        move();
    }
    return stepsTaken;
}
```

- Aantal eieren in een rij tellen

```
public int walkToWorldEdgeAndCountEggs (){
    int nrEggsFound=0;
    while(!borderAhead() && canMove() ){
        if(eggAhead() ){
            nrEggsFound++;
        }
        move();
    }
    return nrEggsFound;
}
```

- Spoor van eieren

```
public void layTrailOfEggs ( int nrOfEggsToLay ){
     int nrOfEggsAlreadyLaid = 0;
     while ( nrOfEggsAlreadyLaid < nrOfEggsToLay ){
        layEgg();
        nrOfEggsAlreadyLaid++;

        if(canMove()){
             move();
        }
    }
}
```

- Aantal eieren in de wereld

1. loop naar beginpunt: `goToLocation` (0,0);
2. loop naar einde rij en tel eieren **`int`** `walkToWorldEdgeAndCountEggs` ()
3. bepaal of eindpunt bereikt is

```
public boolean hasReachedEndCoord(){
    int x_coordinate = getX();
    int y_coordinate = getY();

    World world =  getWorld();

    if( (y_coordinate == (world.getHeight()-1)) && x_coordinate == endCoordinateX() ){
        return true;
    } else {
        return false;
    }
}
```

4. loop naar volgende rij:

```
/*
 * Move down to next row and turn so that facing towards the world
 */
public void goToNextRowAndTurn(){
    if( getDirection()==1 ){
        turnRight();
        move();
        turnRight();
    }else{
        turnLeft();
        move();
```

```
            turnLeft();
        }
    }
```

5. totaaloplossing

```
    public void walkThroughWorldAndCountEggs(){
        int nrEggsFound = 0;
        goToLocation(0,0);
        faceEast();

        nrEggsFound += walkToWorldEdgeAndCountEggs();

        while ( !hasReachedEndCoord() ){
            goToNextRowAndTurn();
            nrEggsFound += walkToWorldEdgeAndCountEggs();

        }
        System.out.println("Walked through the world and found " +nrEggsFound + " eggs.");
        showCompliment("Walked through the world and found " +nrEggsFound + " eggs.");
        Greenfoot.stop();

    }
```

- bepaal de rij de met meeste eieren

```
public void findRowWithMostEggs (){
    World world =  getWorld();

    int maxEggsFoundInRow = 0;
    int rowNrWithMostEggs = -1;

    int currentRowNr = 0;
    int nrEggsThisRow = 0;

    while( currentRowNr <= world.getHeight()-1  ){
        goToLocation (0, currentRowNr); // start in left cell of row
        faceEast();  // face to the right
        nrEggsThisRow = walkToWorldEdgeAndCountEggs();

        if( nrEggsThisRow >= maxEggsFoundInRow ){
            maxEggsFoundInRow = nrEggsThisRow;
            rowNrWithMostEggs = currentRowNr;
        }
        currentRowNr++;

    }
    showCompliment ("Row nr: " + rowNrWithMostEggs + " has "+ maxEggsFoundInRow + " max eggs" );
    System.out.println("Row nr: " + rowNrWithMostEggs + " has most eggs, namely: " +maxEggsFoundInRow

}
```

- bepaal hoeveel eieren er per rij gemiddeld liggen

```
public void walkThroughWorldAndCountAverageEggsPerRow(){
    int nrEggsFound = 0;
    int nrRowsCounted = 0;

    goToLocation(0,0);
```

```
        faceEast();

        nrEggsFound += walkToWorldEdgeAndCountEggs();
        nrRowsCounted++;

        while ( !hasReachedEndCoord() ){
            goToNextRowAndTurn();
            nrEggsFound += walkToWorldEdgeAndCountEggs();
            nrRowsCounted++;

        }
        double averageEggsPerRow = ( (double)nrEggsFound / nrRowsCounted );
        System.out.println("Walked through the world and found " +nrEggsFound + " eggs.");
        System.out.println("Walked through  " +nrRowsCounted + " rows.");
        System.out.println("Walked through the world and found average of " +averageEggsPerRow + " eggs pe
        Greenfoot.stop();

    }
```