More methods

Methods, parameters and return types

Methods and Robots were Effective

- Methods and variables
- Variables are used
  - to pass information to a method (parameters)
  - to get information back from a method (return values)

Remember the Robots

- Remember the move8() instruction.

```
public void move8()
{
   move();
   move();
   move();
   move();
   move();
   move();
   move();
   move();
}
```

Surely this can this be improved using a for loop

Using a for loop

- Using a for loop makes the method more concise:

```
public void move8()
{
   int i;
   for(i = 0; i < 8; i++)
      move();
}
```

Varying the effect (parameters)

- Wouldn't it be nice to have an instruction that took a number (parameter); i.e. so you could say **`run(6)`** to move six squares, or **`run(4)`** to move four squares.

```
public void go()
{
   run(7);
   turnLeft();
   run(3);
   turnLeft();
}
```

Parameters

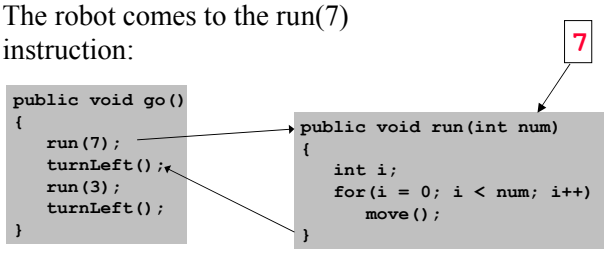
- How would you write such a method? You'd need some way of telling the method how far to move.

```
public void run(int num)
{
   int i;
   for(i = 0; i < num; i++)
      move();
}
```

The parameter controls how many moves are made

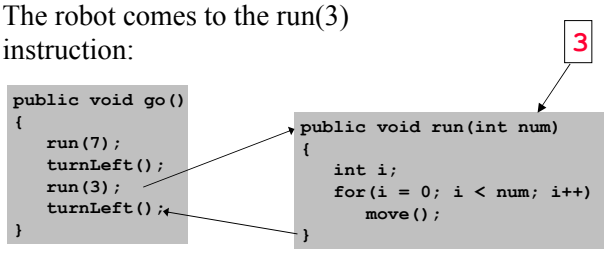
How does it work?

The robot comes to the run(7) instruction:



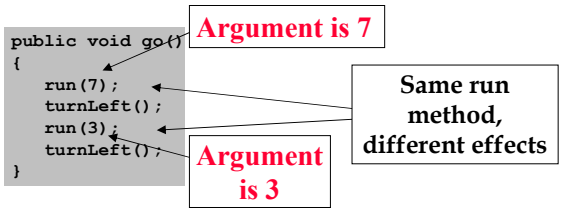
How does it work?

The robot comes to the run(3) instruction:



Parameters modify method behaviour

If you have a method with a parameter, then you can control the effect of the method by passing a different argument.



Arguments, Formal Parameters

- An **argument** is the actual value supplied when the method is invoked.
- The **formal parameter** is the name for the value that will be supplied, i.e. *num* is the formal parameter.
- The first time run() is called the argument is *7*, the second time it is called the argument is *3*.

Method Syntax

Syntax:

```
returnType methodName(paramType paramName, ...)
{
   methodBody
}
```

Example

```
void run(int num)
{
   for(int i = 0; i < num; i++)
      move();
}
```

Purpose:

To define the behaviour of a method.

Exercise

- Can you think of other places which could use this technique?

Return values

- A method can return a value.
- We have already used methods that return values, e.g. frontIsClear() returns a boolean value, Console.readInt() returns an integer value.
- This value would often be used in an assignment statement, e.g.

```
int x = Console.readInt();
```

Example

- A robot is somewhere in a world with no walls. There is one beeper at the origin
- The task is
  - move to the origin, cell(0, 0)
  - pick up the beeper
  - place the beeper on the cell that the robot started out from

Example

- How can we do this?
- We must remember where we are before we head out.
- We have already looked at a related problem
  - move the robot to the origin

Review

- Moving the robot to the origin (pseudocode)
  - turn to face west
  - keep moving til blocked
  - turnLeft
  - keepmoving til blocked

How do we return?

- We need to remember how many steps we took to get to the wall. Now the first part of the problem becomes
  - turn to face west
  - keep moving til blocked ***counting as you go***
  - turnLeft
  - keepmoving til blocked ***counting as you go***
- We could remember these values as x and y.

How do we return? (pseudocode)

- Once we get to the origin:
  - pick up the beeper
  - turn East
  - move x spaces to get to the original x coordinate
  - turn north
  - move y spaces to get to the original y coordinate
  - drop the beeper

Methods returning a value

- A useful method be something that moved to the wall, counting as it went, and then produced (returned) the value. E.g. we would like to be able to write

```
int x;
x = countToWall();
```

How?

- The method is going to return an int, i.e. the return type is int so this is how the method is defined

Return type →

```
int countToWall()
{
    :
    :
}
```

How?

- In addition, the method will have a return statement

```
int countToWall()
{
    int numMoves;
        :
        :
    return numMoves;
}
```

types must match (or you get a compile type error)

How?

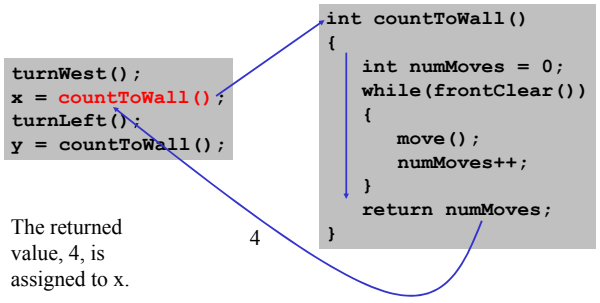
```
int countToWall()
{
    int numMoves = 0;
    while(frontIsClear())
    {
        move();
        numMoves++;
    }

    return numMoves;
}
```

Using the method

```
// Got to origin
turnWest();
int x = countToWall();
turnLeft();
int y = countToWall();

// Pick up beeper

// return to starting position

// drop the beeper
```

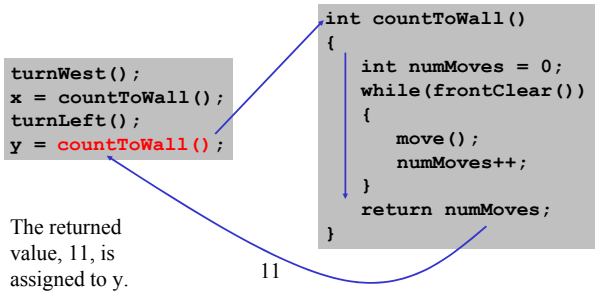
```
// Got to origin
turnWest();
int x = countToWall();
turnLeft();
int y = countToWall();

// Pick up beeper
pickBeeper();

// return to starting position
turnLeft();
run(x);
turnLeft();
run(y);

// lose the beeper
putBeeper();
```

Executing the method



The returned value, 4, is assigned to x.

Executing the method 2



The returned value, 11, is assigned to y.

Return Syntax

Syntax: The return statement

`return` *`expression`*`;`

or

`return ;`

Example

`return numMoves;`

Purpose:

To specify a value that a method returns and exit the method immediately. The return value becomes the value of the method call expression.

Summary

- We have seen two powerful techniques with methods
  - passing information to a method (parameters).
  - returning information from a method
- Both these techniques were illustrated with robot programs, the run() method and the countToWall() method.

Questions

Exercise

Write a method that takes a String and returns a String consisting of the first two characters of the original String. Note that you can use the substring method to extract parts of a String.