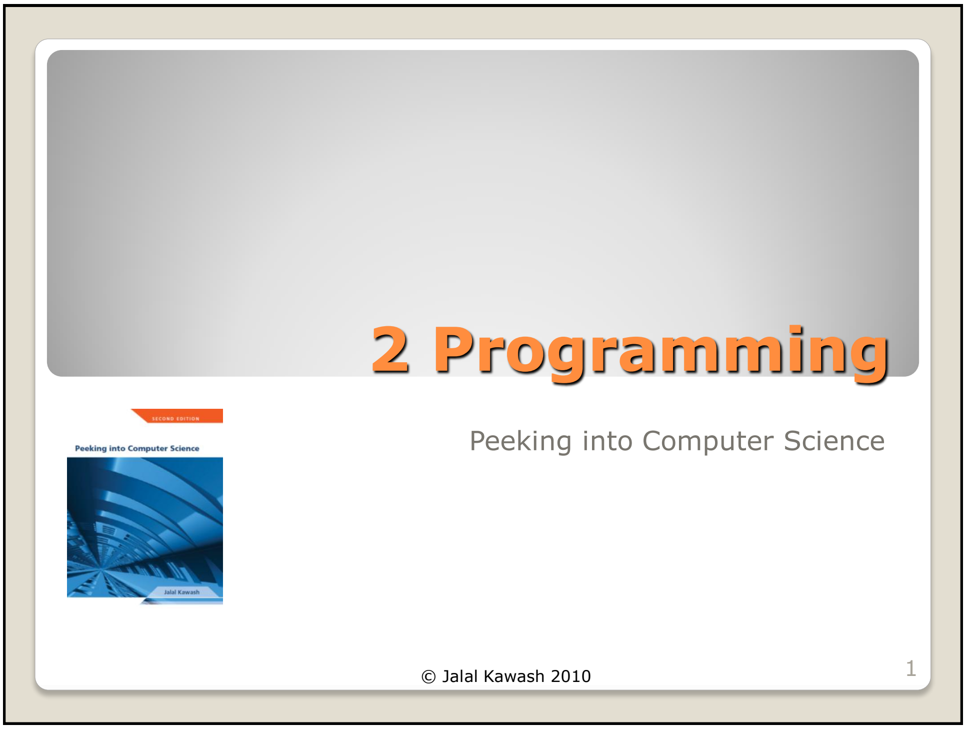
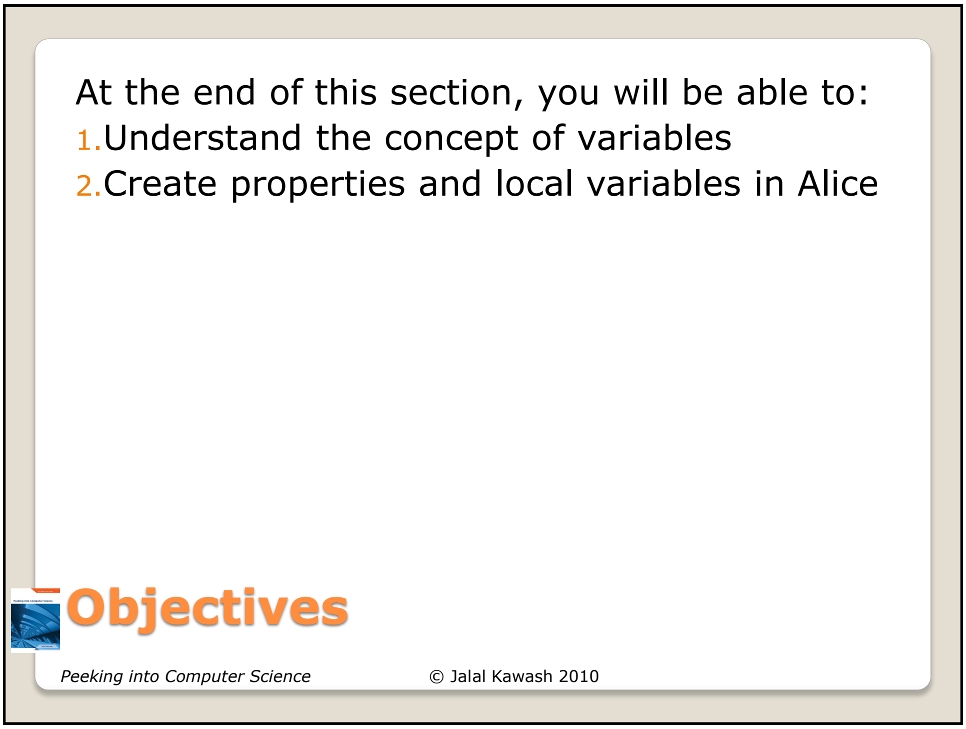


2 Programming

Peeking into Computer Science

SECOND EDITION
Peeking into Computer Science
Jalal Kawash



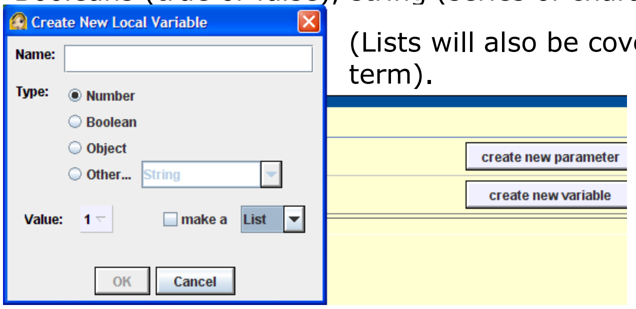
Objectives

At the end of this section, you will be able to:

1. Understand the concept of variables
2. Create properties and local variables in Alice

JT's Extra: Variables

- Are used by a program to temporarily store information.
- Many types of information can be stored by a variable.
  - For this class you will be mostly concerned with: numbers, Booleans (true or false), string (series of characters).



Create New Local Variable

Name:

Type: ● Number ○ Boolean ○ Object ○ Other... String

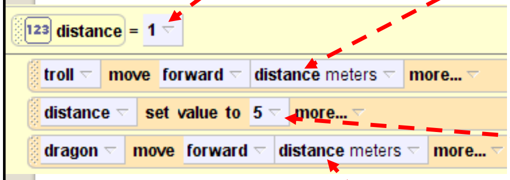
Value: 1 □ make a List

OK Cancel

(Lists will also be covered this term).

Example 3: variables

JT's Extra: Example Use Of A Variable



```
123 distance = 1
troll move forward distance meters more...
distance set value to 5 more...
dragon move forward distance meters more...
```

Distance is a variable with an initial value of 1.

The no. of meters forward that the troll moves is determined by the current value stored in distance (1).

Variable distance now stores the number 5.

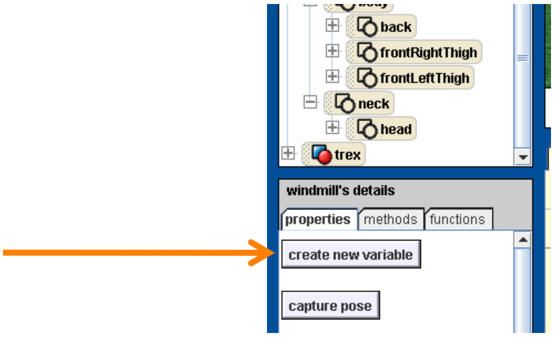
The no. of meters forward that the dragon moves is determined by the current value stored in distance (which is now 5).

JT’s Extra: Variable Naming Conventions

- Should be meaningful
  - E.g., ‘x’, ‘y’ (bad) vs. ‘age’, ‘distance’ (better)
- Variable names should generally be all lower case.
- For variable names composed of multiple words separate each word by capitalizing the first letter of each word (save for the first word) or by using an underscore. (Be consistent!)
  - netIncome
  - gross_pay
  - (Avoid using spaces).

Property Variables

- Properties of an object:



JT's Extra: Memory Is Compartmentalized

- Normally the information associated with a program (memory for that program) only lasts while that program runs.

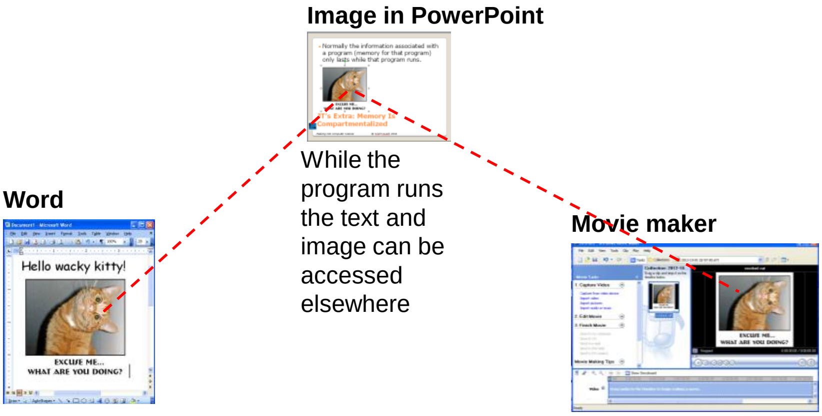


Image in PowerPoint

While the program runs the text and image can be accessed elsewhere

Word

Movie maker

JT's Extra: Memory Is Compartmentalized

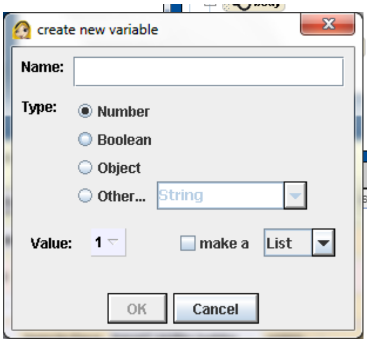
- Normally the information associated with a program (memory for that program) only lasts while that program runs.

The program (PowerPoint) ends and it's data is no longer accessible (localized to the program unless copied)

Word

Movie maker

- Local to methods
  - Do not exist outside the method in which they are contained

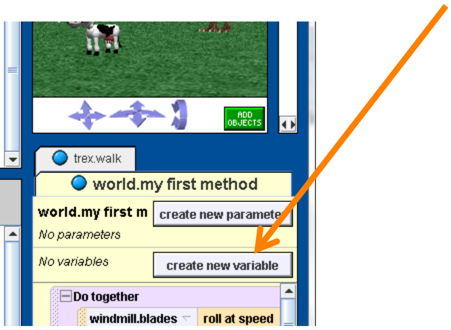


create new variable

Name:

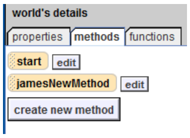
Type: ● Number ○ Boolean ○ Object ○ Other... String

Value: 1 □ make a List

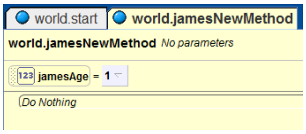


Local Variables

Methods of the example program: ‘start’, ‘jamesNewMethod’



Local variable of ‘jamesNewMethod’: ‘jamesAge’

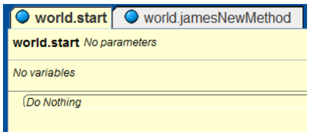


world.jamesNewMethod *No parameters*

123 jamesAge = 1

Do Nothing

Local variable ‘jamesAge’ not in the ‘start’ method



world.start *No parameters*

No variables

Do Nothing

JT’s Extra: Local Variables

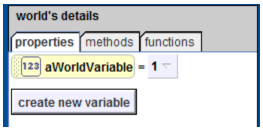
JT’s Extra: World Level Variables

- As the name implies they can be accessed anywhere in the world.
- In practice it means that these variables can be used in any method of that world.

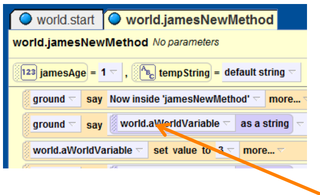
JT’s Extra: World Level Variables

Example 4: Local vs. World variables, string conversion

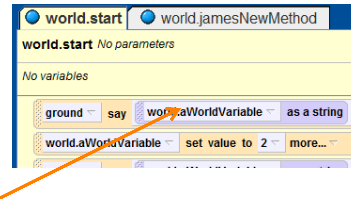
World-level variables: properties of the world



```
world's details
properties | methods | functions
123 aWorldVariable = 1
create new variable
```



```
world.start | world.jamesNewMethod
world.jamesNewMethod No parameters
123 jamesAge = 1 , ABC tempString = default string
  ground say Now inside 'jamesNewMethod' more...
  ground say world.aWorldVariable as a string
  world.aWorldVariable set value to 3 more...
```



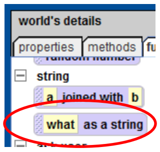
```
world.start | world.jamesNewMethod
world.start No parameters
No variables
  ground say world.aWorldVariable as a string
  world.aWorldVariable set value to 2 more...
```

World level variable can be directly accessed in the methods.

JT's Extra: String Conversion

- An object can only display string information when it 'says' or 'thinks' something.
- Numeric information must be converted to a string before it can be displayed:
  - World->functions->"what as string"

JT's Extra: String Conversion (2)



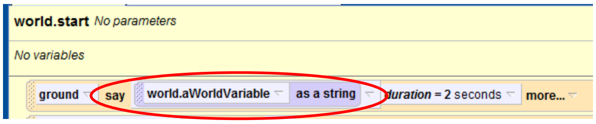
world's details

properties | methods | fu

string

a joined with b

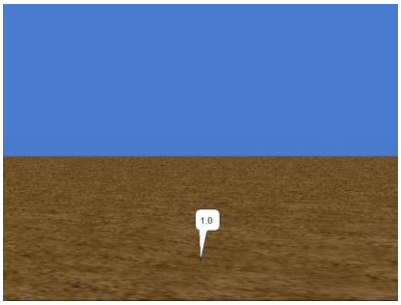
what as a string



world.start *No parameters*

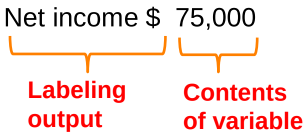
No variables

```
ground say world.aWorldVariable as a string duration = 2 seconds more...
```



JT’s Extra: Captioning Output

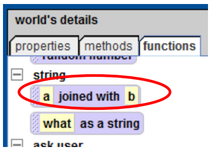
- Sometimes when many variables are being displayed (say/think) it’s hard to track the information.
- Captioned or labeling which variable is being displayed can be useful.
  - World->functions->”joined with”
  - (You join the caption with the display of the variable’s contents).

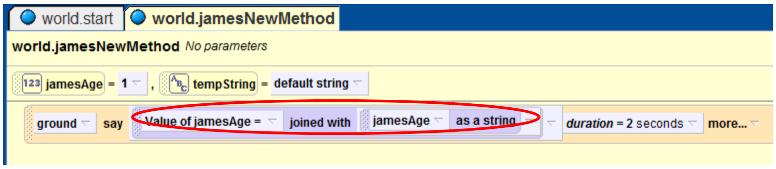

Net income $ 75,000

Labeling output

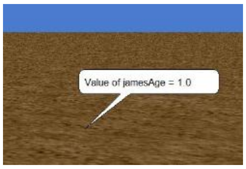
Contents of variable

JT’s Extra: Captioning Output (2)





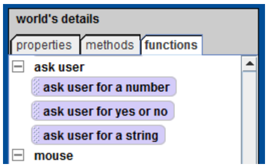
```
world.jamesNewMethod No parameters
123 jamesAge = 1 , tempString = default string
ground say Value of jamesAge = joined with jamesAge as a string duration = 2 seconds more...
```



JT' Extra: Getting User Input

Example 5:
captioning output
(applying 'join'),
getting user input

Getting user input: functions of class world

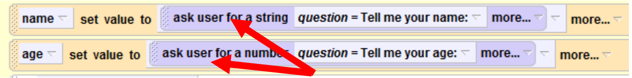


world's details

properties | methods | functions

- ask user
  - ask user for a number
  - ask user for yes or no
  - ask user for a string
- mouse

Drag the appropriate function when setting the value of variables

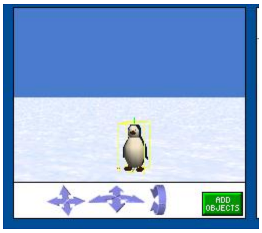


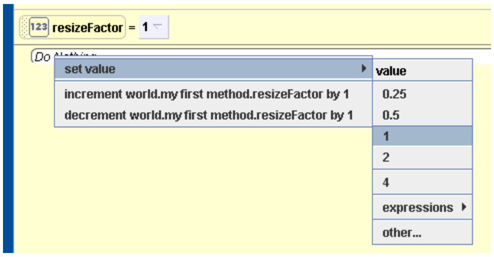
```
name set value to ask user for a string question = Tell me your name: more... more...
age set value to ask user for a number question = Tell me your age: more... more...
```

Source of information for the variable is the user.

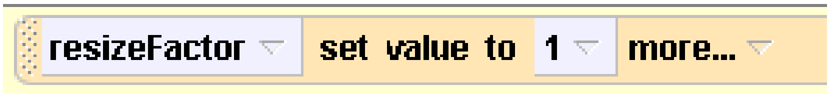
Assignment Statement & User Input

Assignment Statement



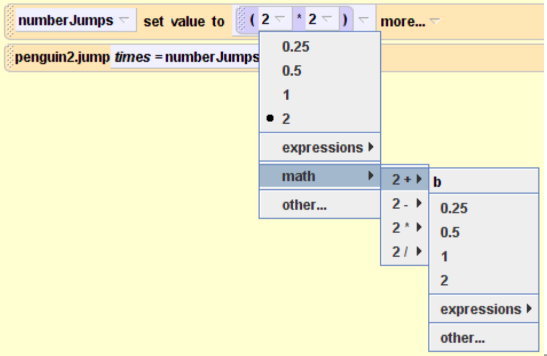

123 resizeFactor = 1

Do Nothing
set value
increment world.my first method.resizeFactor by 1
decrement world.my first method.resizeFactor by 1
value
0.25
0.5
1
2
4
expressions
other...


resizeFactor set value to 1 more...

JT’s Extra: Mathematical Expressions

- Most any arbitrary mathematical expression can be created in Alice.
- Expressions are created graphically.
- Example:


numberJumps set value to (2 * 2) more...

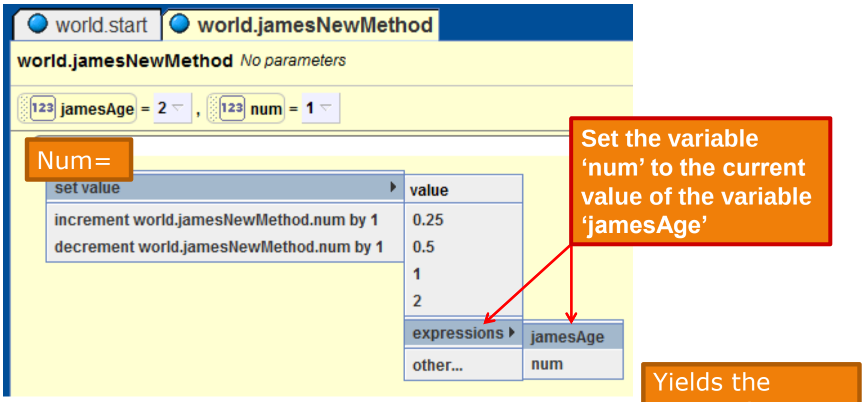
penguin2.jump *times* = numberJumps

JT's Extra: Common Mathematical Operations

Mathematical operation	Symbol used in Alice
Addition	+
Subtraction	-
Multiplication	*
Division	/

JT's Extra: Copying From a Variable Is An Expression

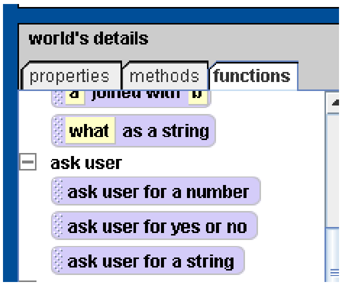
Example:
We want to add the expression: num = jamesAge

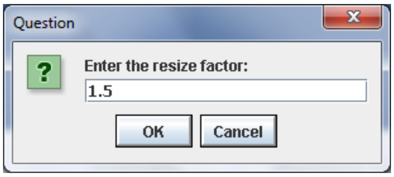

world.start
world.jamesNewMethod
world.jamesNewMethod No parameters
Num=
set value
increment world.jamesNewMethod.num by 1
decrement world.jamesNewMethod.num by 1
value
0.25
0.5
1
2
expressions
other...
jamesAge
num

123 jamesAge = 2 , 123 num = 1

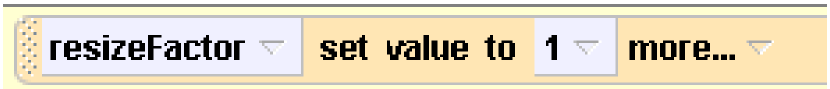
Set the variable 'num' to the current value of the variable 'jamesAge'

Yields the expression: Num = jamesAge



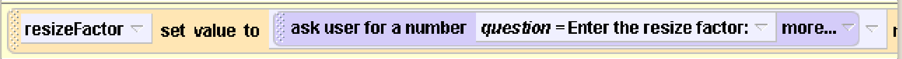


Enter the resize factor: 1.5



```
resizeFactor set value to 1 more...
```

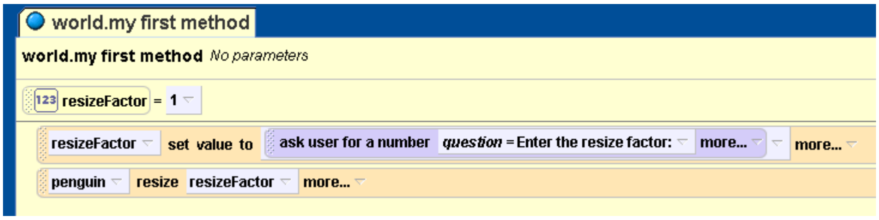
Drag and drop *ask user for a number* function to replace the value 1



```
resizeFactor set value to ask user for a number question = Enter the resize factor: more...
```

User Input

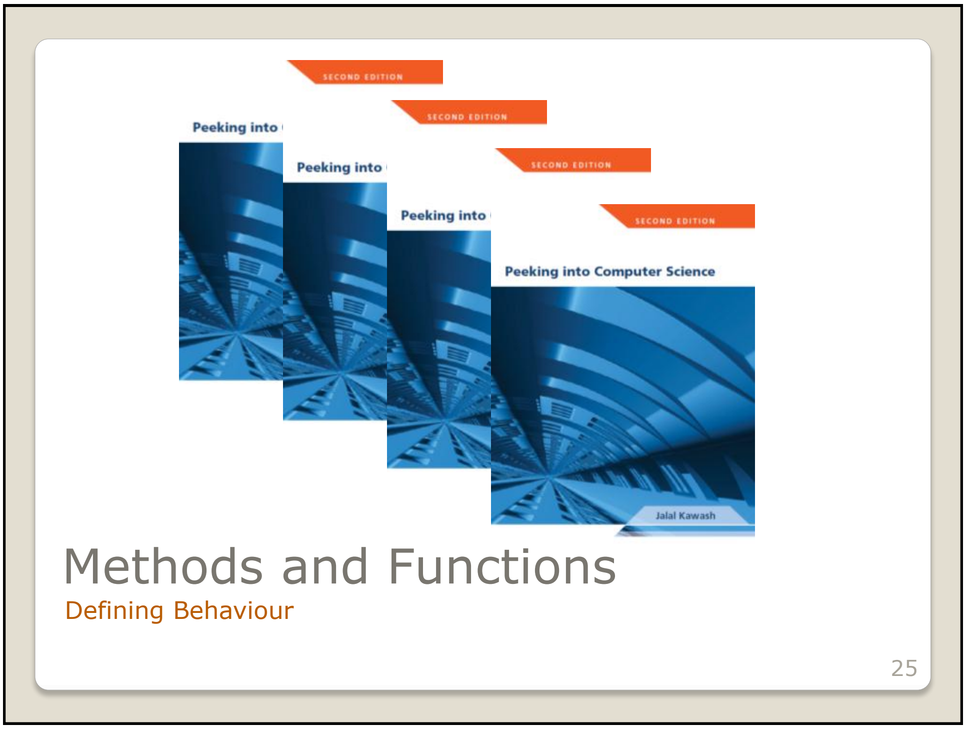
Peeking into Computer Science © Jalal Kawash 2010 23



```
world.my first method No parameters
123 resizeFactor = 1
  resizeFactor set value to ask user for a number question = Enter the resize factor: more... more...
  penguin resize resizeFactor more...
```

The Complete Program

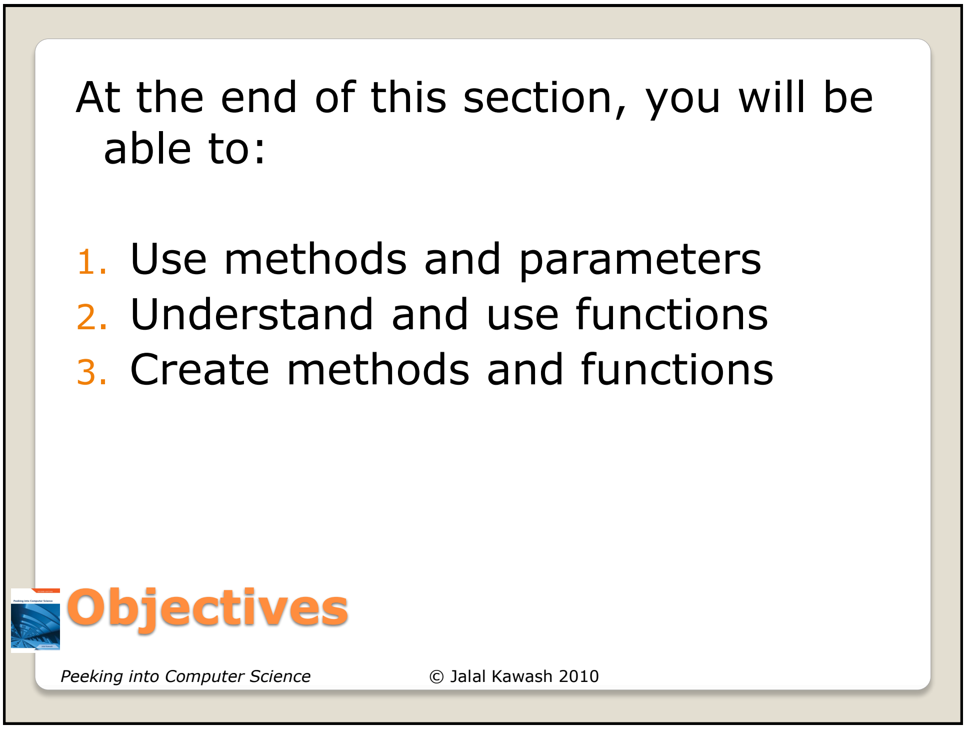
Peeking into Computer Science © Jalal Kawash 2010 24



Methods and Functions

Defining Behaviour

25



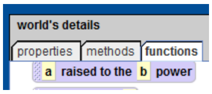
At the end of this section, you will be able to:

1. Use methods and parameters
2. Understand and use functions
3. Create methods and functions

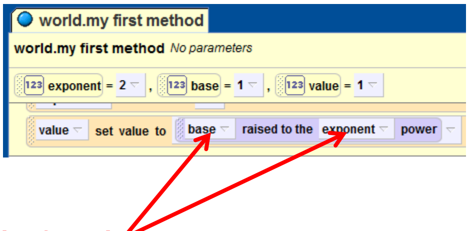
Objectives

Peeking into Computer Science © Jalal Kawash 2010

- Parameters is information that is passed into a function or a method.
- Example: exponent function of world (base, exponent)
  - Based raised to the power of the exponent.
  - Based and exponent is information passed to the exponent function.


world's details

a raised to the b power


world.my first method

world.my first method *No parameters*

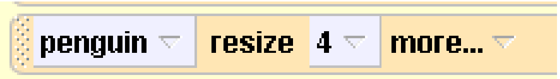
123 exponent = 2 , 123 base = 1 , 123 value = 1

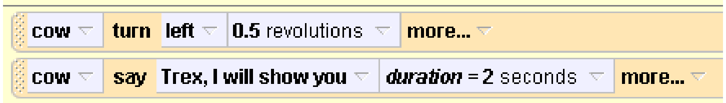
value set value to base raised to the exponent power

Parameters to the function

JT's Extra: Parameters

- A method is defined inside an object
- It has a name
- And zero or more parameters
- A method is called within an object:


penguin resize 4 more...

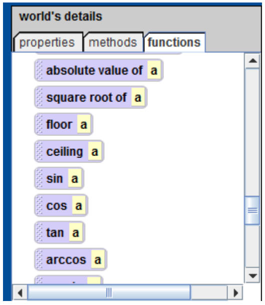

cow turn left 0.5 revolutions more...

cow say Trex, I will show you *duration* = 2 seconds more...

Methods

JT's Extra: Function Definition

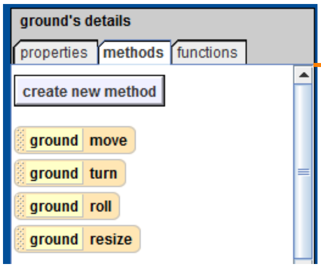
- A function definition indicates what a function will do when it is run.
- Examples (functions of class world which have already been defined by the creators of Alice).



Execute some instructions and return a value

JT's Method Definition

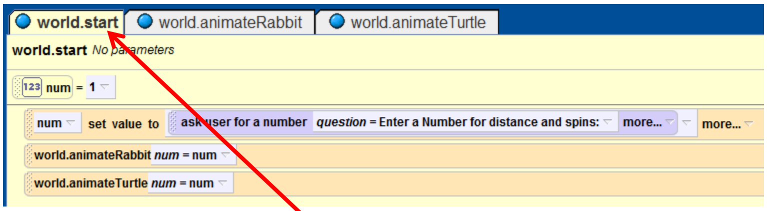
- A method definition indicates what a method will do when it is run.
- Examples (methods of class world which have already been defined by the creators of Alice).



Methods act on or do something to the object that it belongs to.

- When a method is actually run.

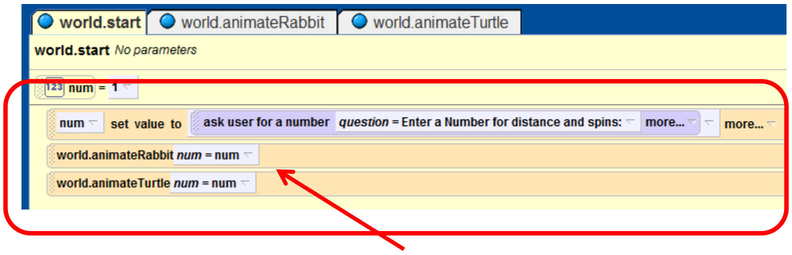
Example 6: method definitions



```
world.start | world.animateRabbit | world.animateTurtle
world.start No parameters
123 num = 1
num set value to ask user for a number question = Enter a Number for distance and spins: more... more...
world.animateRabbit num = num
world.animateTurtle num = num
```

This is the one special method that automatically executes when the program is run.

JT's Extra: Method Call

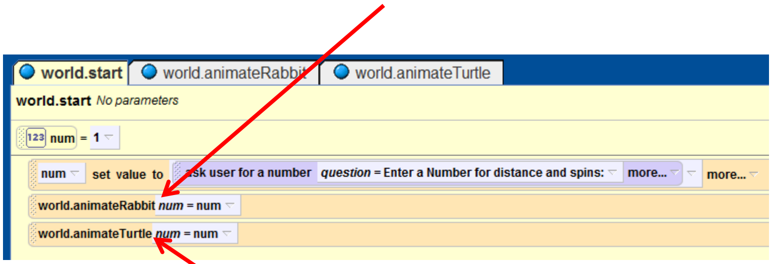


```
world.start | world.animateRabbit | world.animateTurtle
world.start No parameters
123 num = 1
num set value to ask user for a number question = Enter a Number for distance and spins: more... more...
world.animateRabbit num = num
world.animateTurtle num = num
```

Defines the instructions that will execute when the start method executes

JT's Extra: Method Definitions And Calls

Method call: runs the animateRabbitMethod

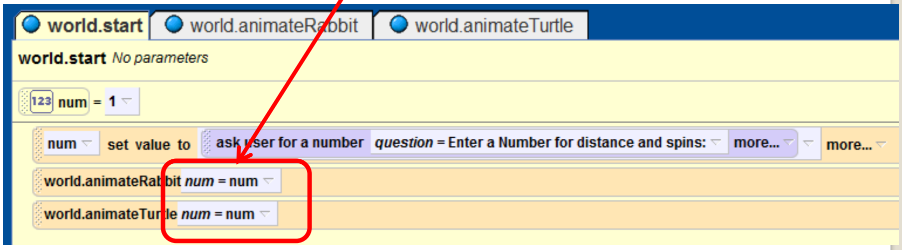


```
world.start No parameters
123 num = 1
    num set value to ask user for a number question = Enter a Number for distance and spins: more... more...
    world.animateRabbit num = num
    world.animateTurtle num = num
```

Method call: runs the animateTurtleMethod

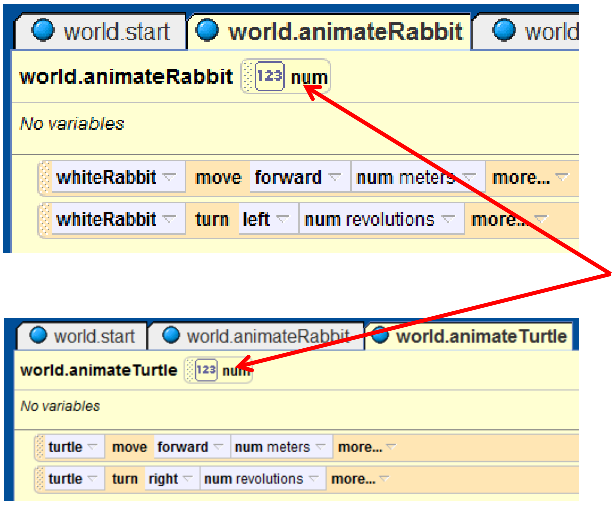
JT's Extra: Method Definitions And Calls

Method parameters: the number entered by the user will be passed into each method.



```
world.start No parameters
123 num = 1
    num set value to ask user for a number question = Enter a Number for distance and spins: more... more...
    world.animateRabbit num = num
    world.animateTurtle num = num
```

JT's Extra: Method Parameters



```
world.start | world.animateRabbit | world
world.animateRabbit  123 num
No variables
whiteRabbit  move  forward  num meters  more...
whiteRabbit  turn  left  num revolutions  more...
```

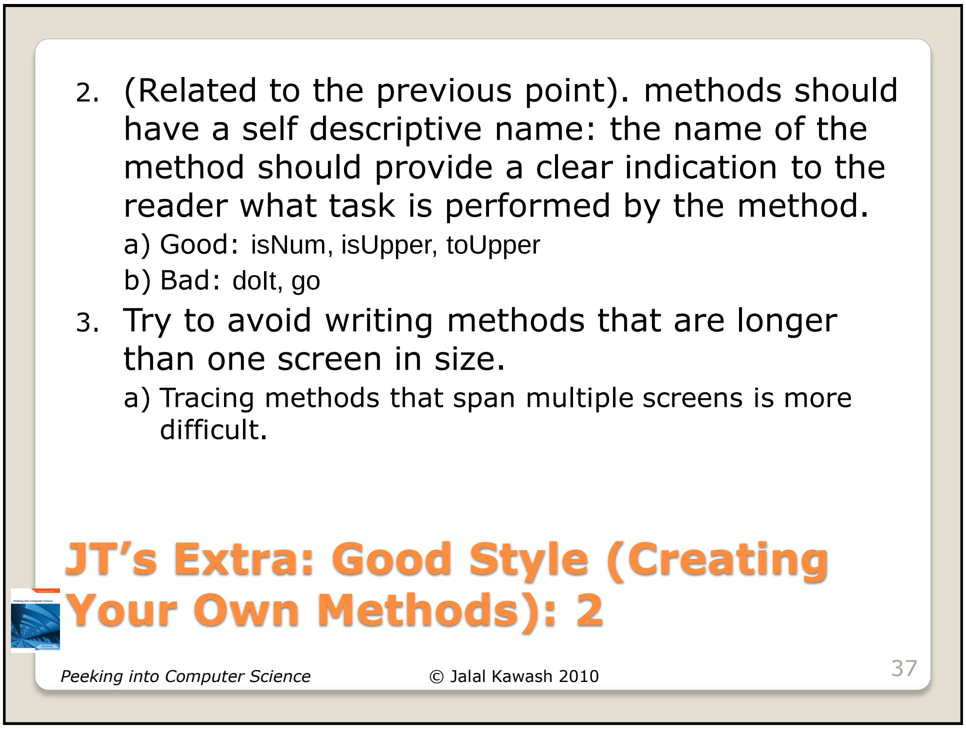
```
world.start | world.animateRabbit | world.animateTurtle
world.animateTurtle  123 num
No variables
turtle  move  forward  num meters  more...
turtle  turn  right  num revolutions  more...
```

Method parameter: the number passed in determines the distance moved and the number of rotations.

JT's Extra: Method Definitions

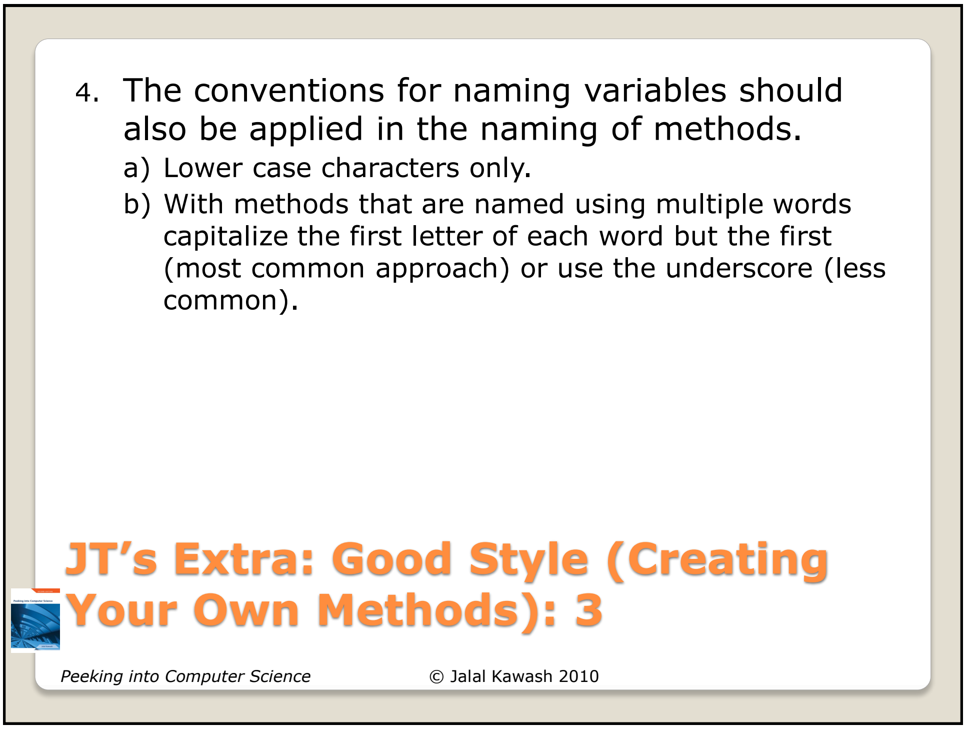
1. Each method should have one well defined task. If it doesn't then it may be a sign that it should be decomposed into multiple sub-methods.
   a) Clear method: A method that converts lower case input to capitals.
   b) Ambiguous method: A method that prompts for a string and then converts that string to upper case.

JT's Extra: Good Style (Creating Your Own Methods)



JT’s Extra: Good Style (Creating Your Own Methods): 2

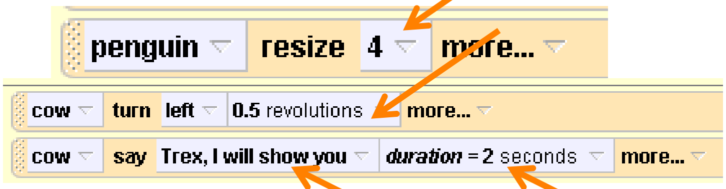
2. (Related to the previous point). methods should have a self descriptive name: the name of the method should provide a clear indication to the reader what task is performed by the method.
   a) Good: isNum, isUpper, toUpper
   b) Bad: doIt, go
3. Try to avoid writing methods that are longer than one screen in size.
   a) Tracing methods that span multiple screens is more difficult.



JT’s Extra: Good Style (Creating Your Own Methods): 3

4. The conventions for naming variables should also be applied in the naming of methods.
   a) Lower case characters only.
   b) With methods that are named using multiple words capitalize the first letter of each word but the first (most common approach) or use the underscore (less common).

- Parameters of a method are like local variables in the method
- The difference is that a value must be assigned to parameters when a method is called



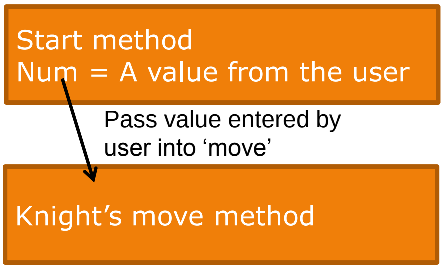
penguin resize 4 more...

cow turn left 0.5 revolutions more...

cow say Trex, I will show you duration = 2 seconds more...

Parameters

- Also you can think of parameters as a mechanism for communicating information into a method.



Start method
Num = A value from the user

Pass value entered by user into ‘move’

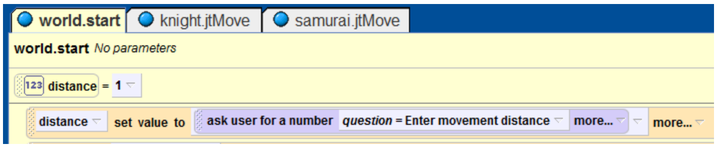
Knight’s move method

JT’s Extra: Parameters

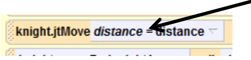
JT's Extra: Parameter Passing

Example 7:
Parameters

Get value for distance from user in 'start' method



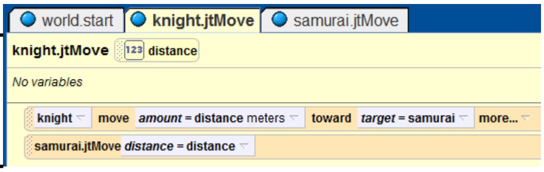
```
world.start No parameters
123 distance = 1
  distance set value to ask user for a number question = Enter movement distance more... more...
```



```
knight.jtMove distance = distance
```

Pass the distance into the move method as a parameter

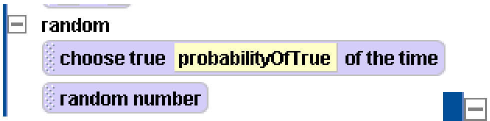
Now 'distance' can be used in the 'move' method



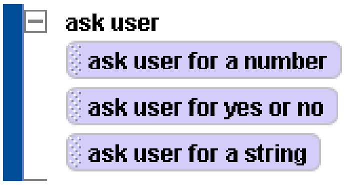
```
knight.jtMove 123 distance
No variables
  knight move amount = distance meters toward target = samurai more...
  samurai.jtMove distance = distance
```

Functions

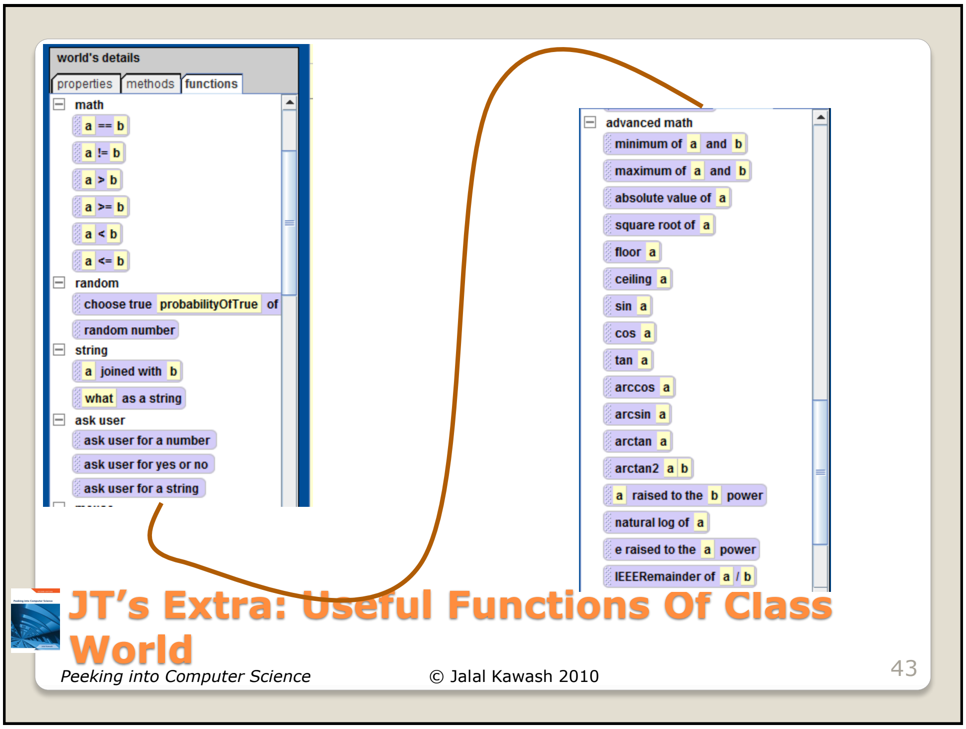
- Functions are methods that return values
- Mathematically, f(n) = 2n
- f(5) returns 10



```
random
  choose true probabilityOfTrue of the time
  random number
```



```
ask user
  ask user for a number
  ask user for yes or no
  ask user for a string
```



JT's Extra: Useful Functions Of Class World

Peeking into Computer Science © Jalal Kawash 2010

Do in order and Do together

Sequential and parallel execution

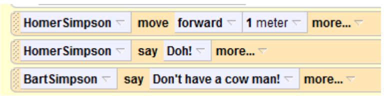
Objectives

At the end of this section, you will be able to:

1. Understand the difference between sequential and parallel execution
2. Make use of the Do in order and do together statements
3. Draw flowcharts for story lines with parallel and sequential actions

Sequential and Parallel Execution

- This we have created is sequential
- Straight-line code
- Finish one instruction before starting the next



```
HomerSimpson move forward 1 meter more...
HomerSimpson say Doh! more...
BartSimpson say Don't have a cow man! more...
```

Parallel Execution

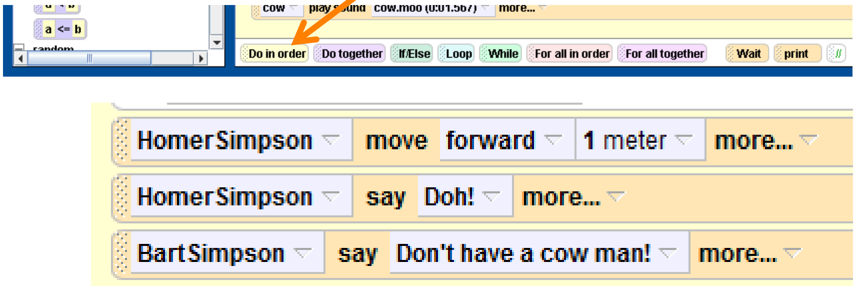
- Two methods executed at the same time
  - In parallel, or concurrently
- The windmill blades roll concurrently with the rest of the movie actions

Trex, I will show you

- Cow turns her head while Trex was running

Do in order

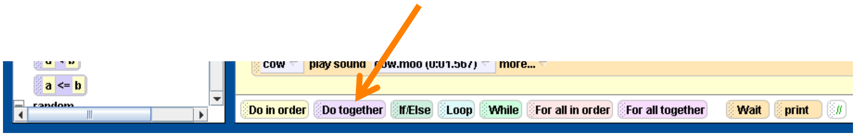
- Default mode of execution
- Can group statements in a *Do in order* statement

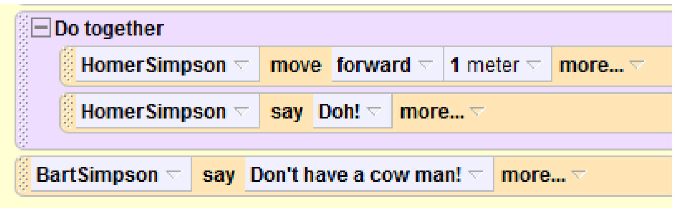


```
HomerSimpson move forward 1 meter more...
HomerSimpson say Doh! more...
BartSimpson say Don't have a cow man! more...
```

- Can group statements in a *Do together* statement to run concurrently

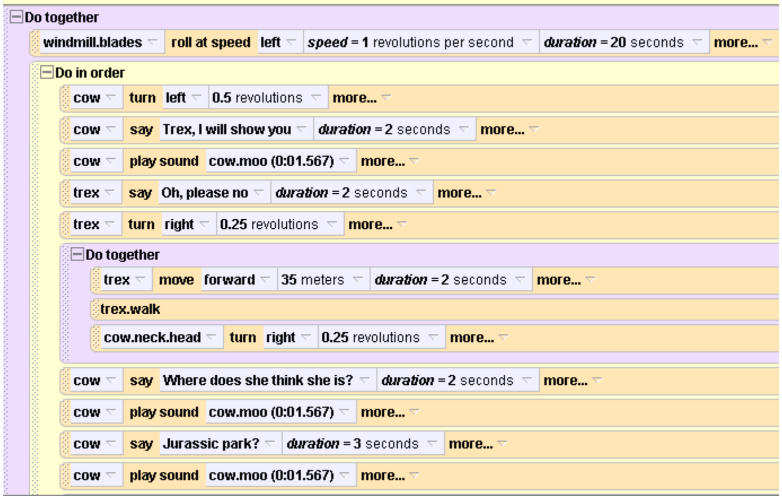
Example 8: simultaneous actions





```
Do together
    HomerSimpson move forward 1 meter more...
    HomerSimpson say Doh! more...
BartSimpson say Don't have a cow man! more...
```

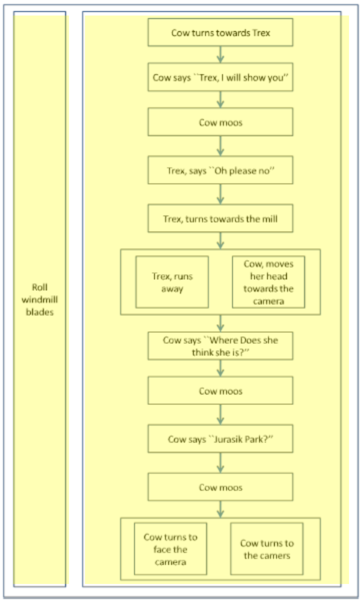
Do together



```
Do together
    windmill.blades roll at speed left speed = 1 revolutions per second duration = 20 seconds more...
    Do in order
        cow turn left 0.5 revolutions more...
        cow say Trex, I will show you duration = 2 seconds more...
        cow play sound cow.moo (0:01.567) more...
        trex say Oh, please no duration = 2 seconds more...
        trex turn right 0.25 revolutions more...
        Do together
            trex move forward 35 meters duration = 2 seconds more...
            trex.walk
            cow.neck.head turn right 0.25 revolutions more...
        cow say Where does she think she is? duration = 2 seconds more...
        cow play sound cow.moo (0:01.567) more...
        cow say Jurassic park? duration = 3 seconds more...
        cow play sound cow.moo (0:01.567) more...
```

Intertwining in order and together

Flowcharts for Story Lines



Roll windmill blades

Cow turns towards Trex

Cow says ``Trex, I will show you''

Cow moos

Trex, says ``Oh please no''

Trex, turns towards the mill

Trex, runs away

Cow, moves her head towards the camera

Cow says ``Where Does she think she is?''

Cow moos

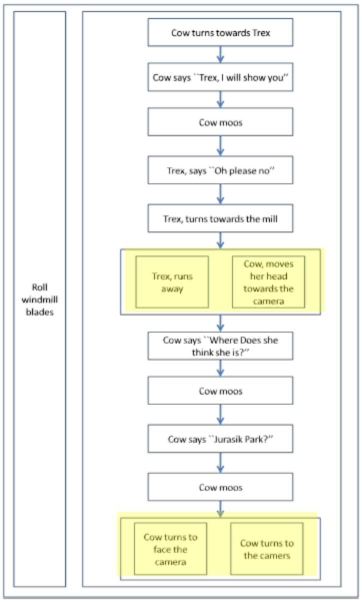
Cow says ``Jurasik Park?''

Cow moos

Cow turns to face the camera

Cow turns to the camers

Flowcharts for Story Lines



Roll windmill blades

Cow turns towards Trex

Cow says ``Trex, I will show you''

Cow moos

Trex, says ``Oh please no''

Trex, turns towards the mill

Trex, runs away

Cow, moves her head towards the camera

Cow says ``Where Does she think she is?''

Cow moos

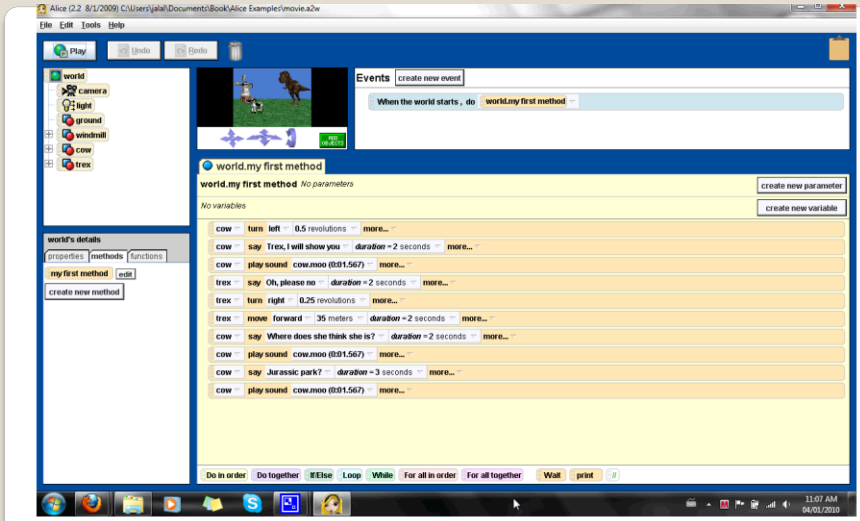
Cow says ``Jurasik Park?''

Cow moos

Cow turns to face the camera

Cow turns to the camers

Do together and in order movie



Alice (2.2 8/1/2009) C:\Users\jalal\Documents\Book\Alice Examples\movie.a2w

```
When the world starts , do  world.my first method

world.my first method  No parameters
No variables
cow  turn  left  0.5 revolutions  more...
cow  say  Trex, I will show you  duration = 2 seconds  more...
cow  play sound  cow.moo (0:01.567)  more...
trex  say  Oh, please no  duration = 2 seconds  more...
trex  turn  right  0.25 revolutions  more...
trex  move  forward  35 meters  duration = 2 seconds  more...
cow  say  Where does she think she is?  duration = 2 seconds  more...
cow  play sound  cow.moo (0:01.567)  more...
cow  say  Jurassic park?  duration = 3 seconds  more...
cow  play sound  cow.moo (0:01.567)  more...
```

JT's Extra: Flowcharts

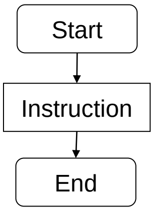
- It's a graphical notation that's used to specify how a computer program will execute.
- Symbols:
  - Terminal (for the start and end of the program).

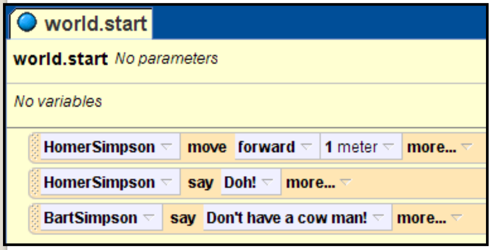
    Start/end

  - Processing (for a step in the program)

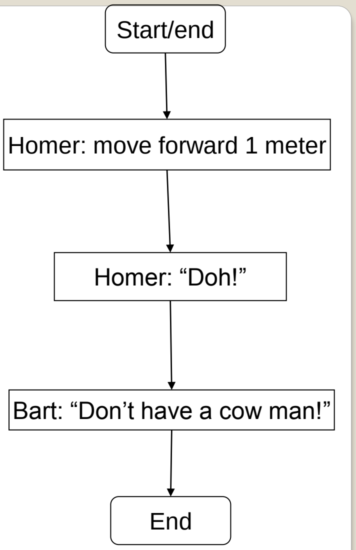
    Processing

- Arrows are used to connect the symbols (show the order and direction of execution).





```
world.start
world.start No parameters
No variables
    HomerSimpson  move  forward  1 meter  more...
    HomerSimpson  say  Doh!  more...
    BartSimpson  say  Don't have a cow man!  more...
```



Start/end

Homer: move forward 1 meter

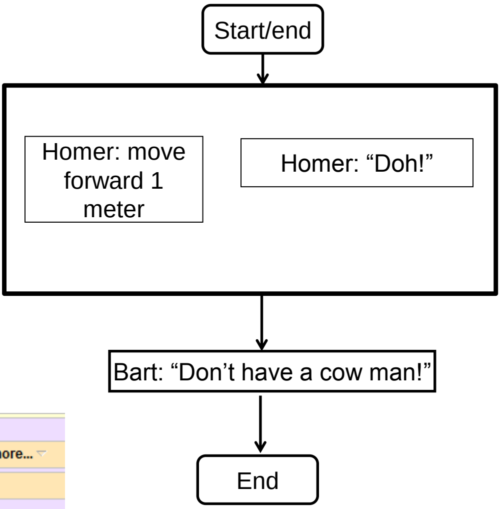
Homer: “Doh!”

Bart: “Don’t have a cow man!”

End

JT’s Extra: Example Program: Flowchart

Peeking into Computer Science © Jalal Kawash 2010



Start/end

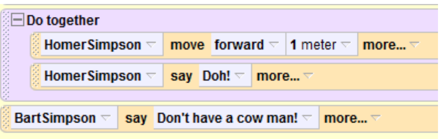
Homer: move forward 1 meter

Homer: “Doh!”

Bart: “Don’t have a cow man!”

End

Encompasses all the instructions that execute at the same time (encompassed by the ‘do together’ container)



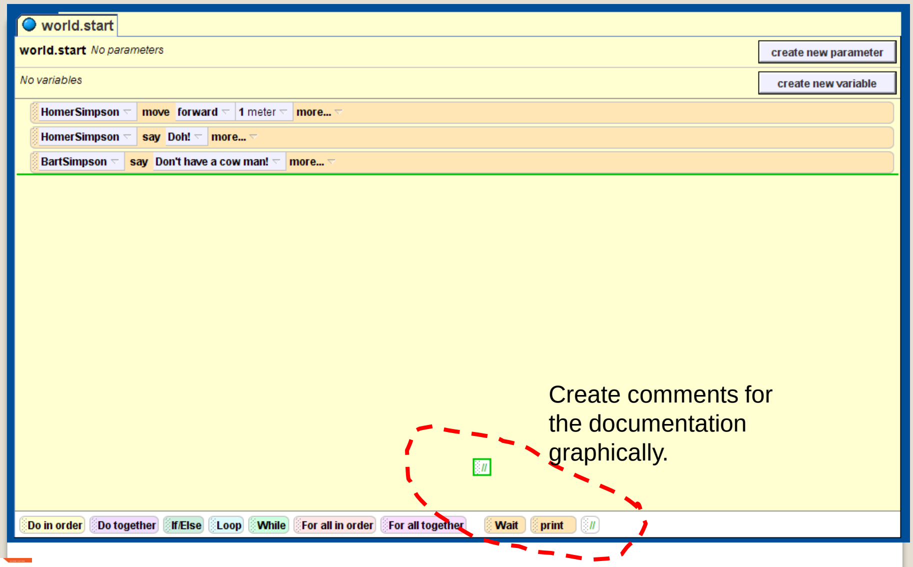
```
Do together
    HomerSimpson  move  forward  1 meter  more...
    HomerSimpson  say  Doh!  more...
BartSimpson  say  Don't have a cow man!  more...
```

JT’s Extra: Flowchart (Do-Together)

Peeking into Computer Science © Jalal Kawash 2010

- English statements inserted into a computer program.
- They are not written in a programming language.
- They are for the reader of the program and won't be executed by the computer.
- It describes 'what' the program does (but not 'how' it does it.
  - Correct: the program sorts contacts into alphabetical order
  - Incorrect: the program uses the quicksort algorithm.

JT's Extra: Program Documentation



world.start

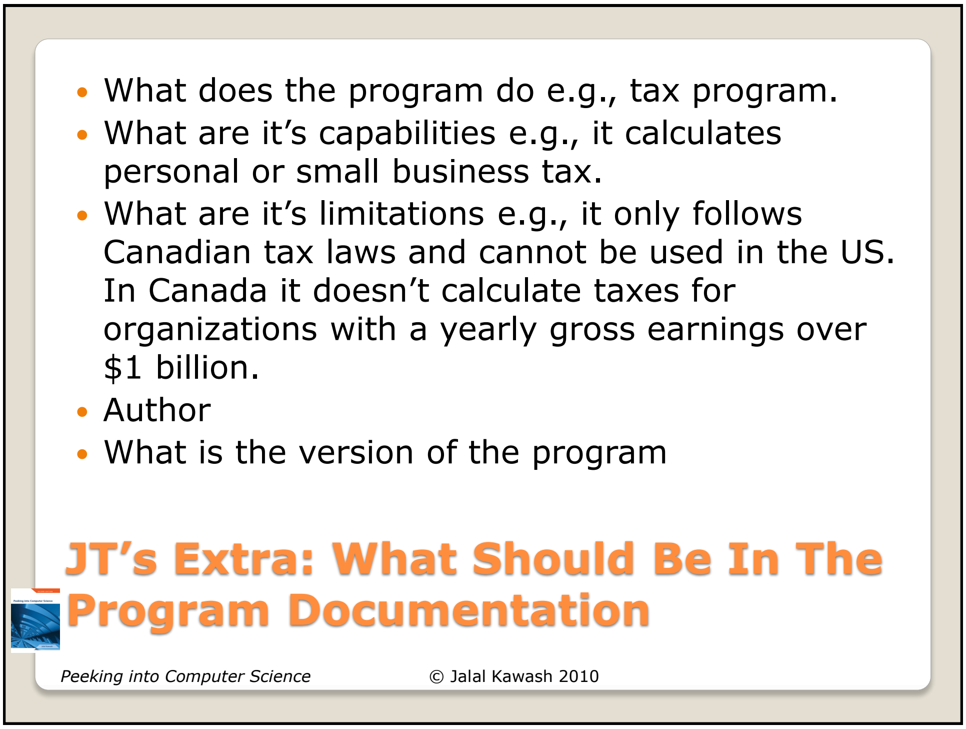
world.start *No parameters*

No variables

```
HomerSimpson move forward 1 meter more...
HomerSimpson say Doh! more...
BartSimpson say Don't have a cow man! more...
```

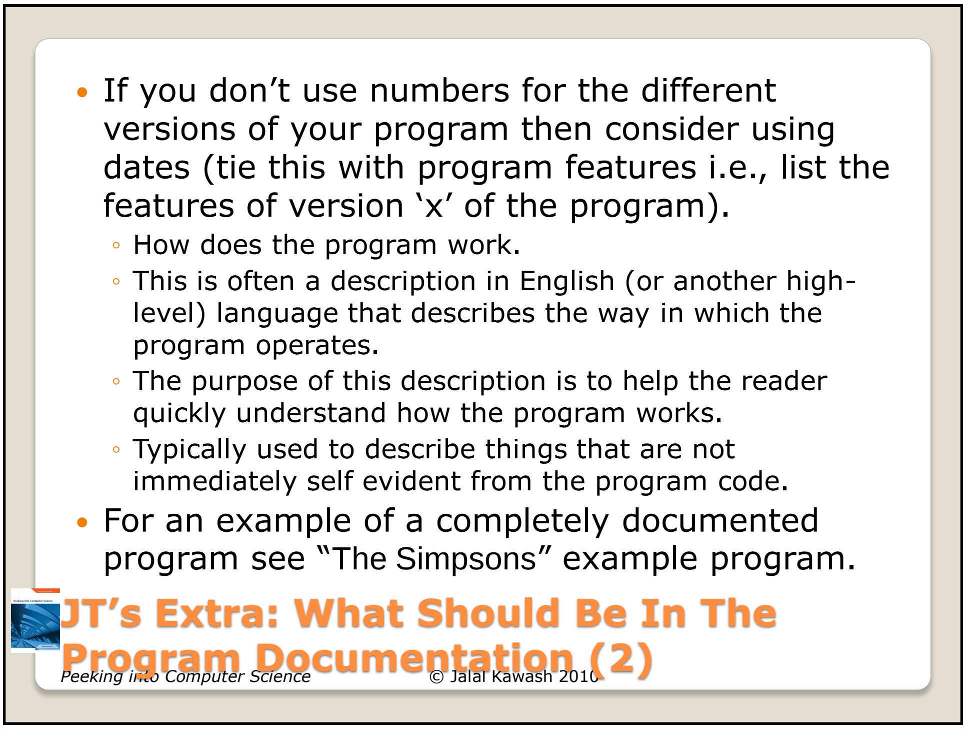
Create comments for the documentation graphically.

JT's Extra: Creating Program Documentation



JT’s Extra: What Should Be In The Program Documentation

- What does the program do e.g., tax program.
- What are it’s capabilities e.g., it calculates personal or small business tax.
- What are it’s limitations e.g., it only follows Canadian tax laws and cannot be used in the US. In Canada it doesn’t calculate taxes for organizations with a yearly gross earnings over $1 billion.
- Author
- What is the version of the program



JT’s Extra: What Should Be In The Program Documentation (2)

- If you don’t use numbers for the different versions of your program then consider using dates (tie this with program features i.e., list the features of version ‘x’ of the program).
  - How does the program work.
  - This is often a description in English (or another high-level) language that describes the way in which the program operates.
  - The purpose of this description is to help the reader quickly understand how the program works.
  - Typically used to describe things that are not immediately self evident from the program code.
- For an example of a completely documented program see “The Simpsons” example program.