Routines: Void Methods in Java

Methods are the basic Big building blocks of programs.
Methods are often called sub-Programs or procedures.
There are two very different kinds of methods:

functions, also called **type** methods, and
routines, also called **void** methods.

Void methods will be considered here first; type methods come later.
Void methods do some action (called a side-effect); they do not return any value;
Type methods return some value; they do not do an action.

Examples of void methods include output methods such as:
`outputInt, outputDouble, outputString, outputlnInt,` etc.
Such void methods stand alone, and end with a semicolon;
they are not parts of other expressions.

Definition of other void methods will be done here.
Many of these methods will also involve outputs, such as:

`outRow (many, mark)`, which outputs a row of many marks,
`spellout (number)`, which spells out a number in English,
`formatMoney (amount)` which prints an amount in a good form,
`sidePlot()` plots out some arithmetic functions or formulas.

OutRow (num, str) is an example of a void method that involves two slots num and str (often called parameters or arguments). When this method is used or called in a command like:

```
outRow (7, "Ho ");
```

it prints out the string "Ho "for a total of 7 tmes in a row, as:

```
Ho Ho Ho Ho Ho Ho Ho
```

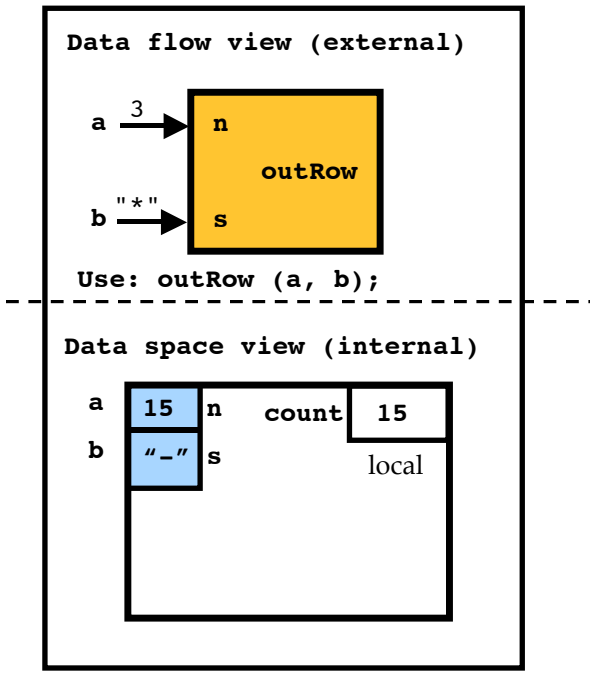
Similarily, the call

```
outRow(30, "-")
```

outputs a row of 30 dashes as follows:

```
------------------------------
```

Graphic of a void method



Externally, this method can be viewed as a black box, which has two input slots (also called parameters):

a, is an integer number of repetitions, and

b, is the string which is being repeated.

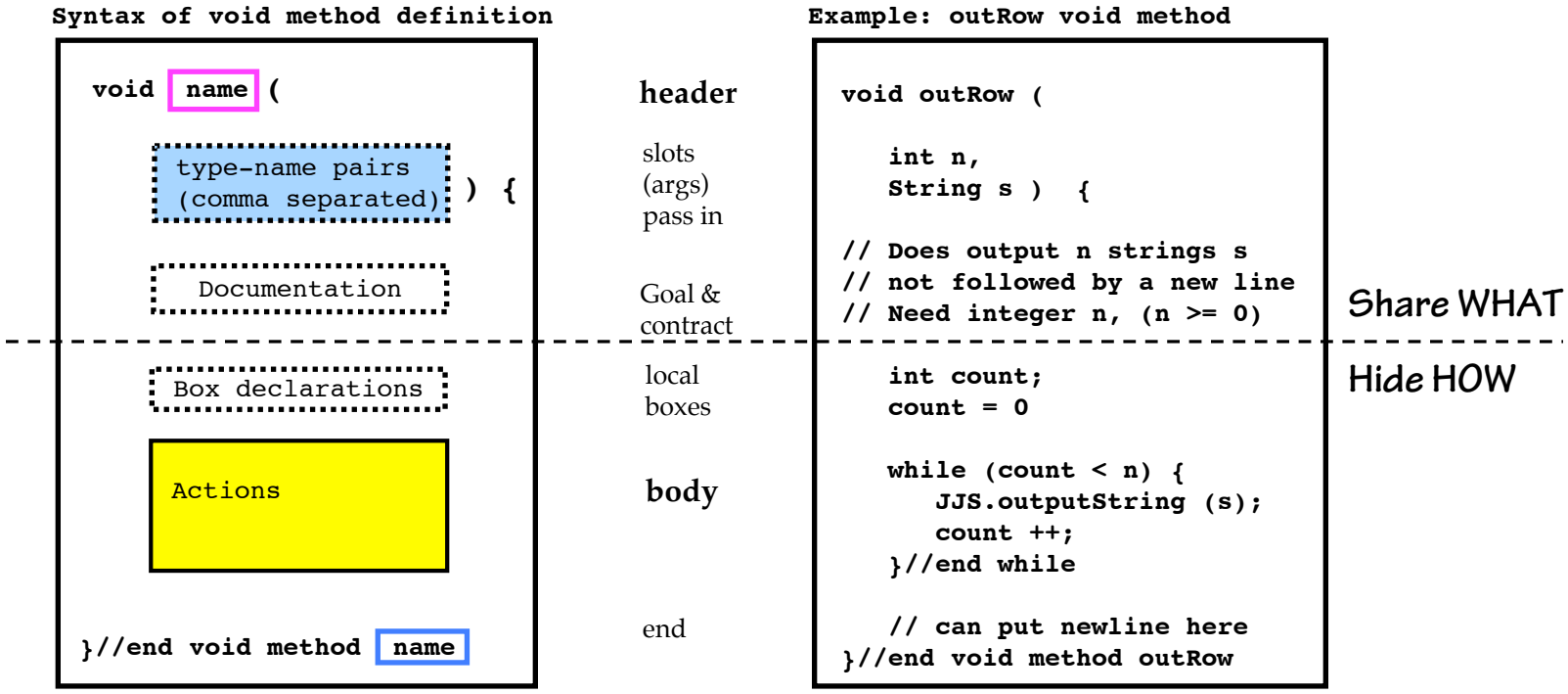
Internally the two slots are labelled n and s, and there is a matching between these two; a is passed to n, and b is passed to s. The order is very important, so that

```
outRow ("Ho ", 30);
```

has no meaning, and is an error; the number is expected first.

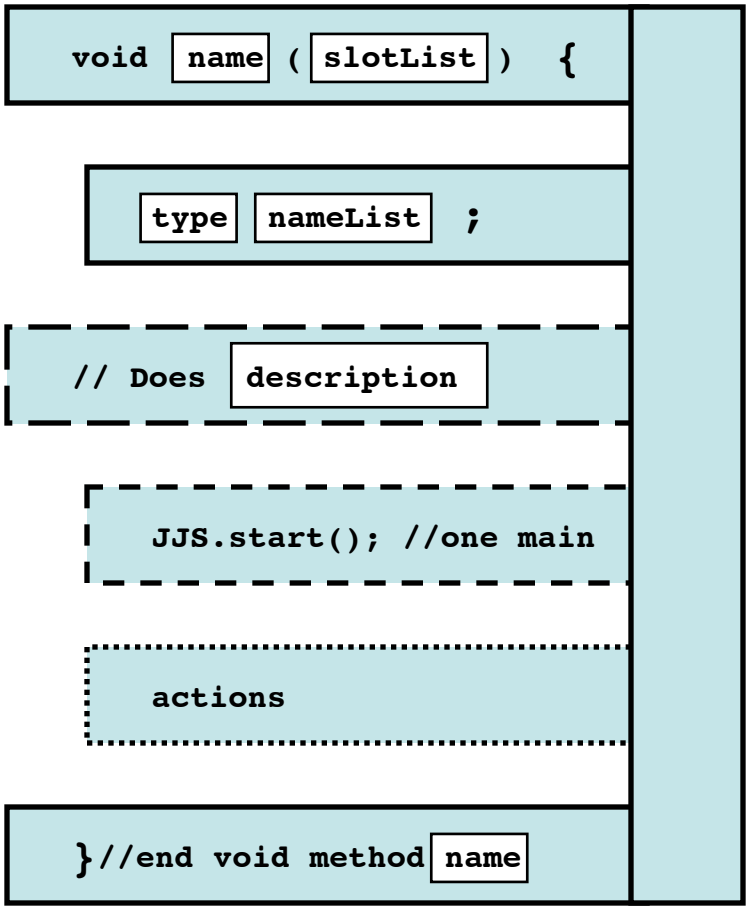
Local, or temporary boxes include the count, which is hidden from the outside. Another box outside of this method may have this same name, count.

Definition of a void method is shown below; it consists of two main parts:
Header is the part which gives the name, list of slots, and goal is shared with others;
Body is the part with local or temporary boxes and local actions which are hidden from others.



The header provides a **contract** between the creator and user of a method.
Does, is also called a **post-condition**; it indicates the outcome of the method.
Need, is a **pre-condition**, indicating what the method requires to satisfy the contract.
In this case the number of repetitions, n, must be a non-negative integer.
If n is provided as a real value, or a string, or boolean or negative the contract is broken.

Syntax of a void method is shown below, at the left;Two examples of void methods are at the right.
main0 calls outRow(5, "Ho ") to print out a row of 5 Hos, "Ho Ho Ho Ho Ho". It also underlines them.




```
void main0() {
   int count;
// Does reuse outRow method

   JJS.start();

   outRow (5,"Ho ");
   JJS.outputlnString ("");//newline

   outRow (14,"-" );
   JJS.outputlnString (" ");

}//end void method main0
```

```
void
outRow (int many, String marks) {
   int count;
// Does output a row of many marks
// Needs (many >= 0)
   count = 0;
   while (count < many) {
      JJS.outputString (marks);
      count = count + 1;
   }//end while
}//end void method outRow
```

JJS.start() is a command that can appear in at most one void method;
It indicates the starting method, and is usually in the main void method.

```
Ho Ho Ho Ho Ho
--------------
```

Another *example of a main program using the outRow method follows.*
It prompts for string, enters it, and "sandwiches" the whole string.

```
// Name An Onymous

void main1() {
   int len;
   String str;
// Does "box" a string using outRow
   JJS.start();

   JJS.outputlnString ("Enter any string ");
   str  = JJS.inputString();
   JJS.outputlnString (str);

   len = str.length();

// output a row of dashes (-)
    outRow (len, "-");
   JJS.outputlnString(" ");

   JJS.outputlnString(str);

// Output a row of equals (=)
    outRow (len, "=");
   JJS.outputlnString(" ");

}//End void method main1
```

```
void
outRow (int many, String marks) {
   int count;
// Does output a row of many marks
// Needs (many >= 0)
   count = 0;
   while (count < many) {
      JJS.outputString (marks);
      count = count + 1;
   }//end while
}//end void method outRow
```

```
Enter a string
Once upon a time there lived 3 bulls
-----------------------------------
Once upon a time there lived 3 bulls
===================================
```

Spellout(number) is a routine which prints out the passed number in English.

```
// Name An Onymous
void main2() {
   int number;
// Does spell out some small numbers
   JJS.start();
   JJS.outputlnString ("Enter an integer ");
   number = JJS.inputInt ();

   spellOut (number);
   JJS.outputln();
}//end void method main2
```

```
Enter an integer
2
two

Enter an integer
4
4

Enter an integer
-2
-2
```

```
void spellOut (int num) {
  //no local, temporary boxes
  // Needs positive int (0 to 3)
  // Does write out the int number
     if (num == 0) {
        JJS.outputString ("zero " );
     }else if (num == 1) {
        JJS.outputString ("one "  );
     }else if (num == 2) {
        JJS.outputString ("two "  );
     }else if (num == 3) {
        JJS.outputString ("three ");
     //grow more here
     }else{
        JJS.outputInt (num);
     }//end if
  }//end void method spellOut
```

```
// Name An Onymous
void main3() {
   int num, tens, units;
// Does write ints 0 to 100 in English
   JJS.start();

   JJS.outputlnString ("Enter an int: ");
   num  = JJS.inputInt ();

   if (num < 10) {
      spellDigit (num);

   }else if (num < 20) {
      spellTeen (num);

   }else{
       tens = num / 10;
       spellTens (10 * tens);

       units = (num % 10);
       if ( (units) != 0) {
          spellDigit (units);
       }//end if
   }//end if
}//end void method main3
```

```
Enter an int:
0
zero

Enter an int:
13
thirteen

Enter an int:
30
thirty

Enter an int:
53
fifty three

Enter an int:
123
???? three
```

```
void spellDigit (int num) {
   // Need (0 <= num) && (num < 10)
   // Does write digits 0 to 9 in English
```

```
void spellTeen (int num) {
   // Need (11 <= num) && (num < 20)
   // Does write numbers 11 to 19 in English
```

```
void spellTens (int num) {
   // Need (10 <= num) && (num < 100) &&
   // Does write out int numbers 10 to 99
```

Does show the use of three methods (given as **stubs**, not in detail) defined by assertions:

Does: is a **post**-condition
Need: is a **pre**-condition

SpellDigit has details which are very similar to spellOut; you do the others.

```
// Name An Onymous
void main4() {
   double amt;
// Does show use of formatMoney method
   JJS.start() {

   JJS.outputlnString ("Enter dollars ");
   amt = JJS.inputDouble ();

   formatMoney (amt);
   JJS.outputString ("\n");

}//end void method main4
```

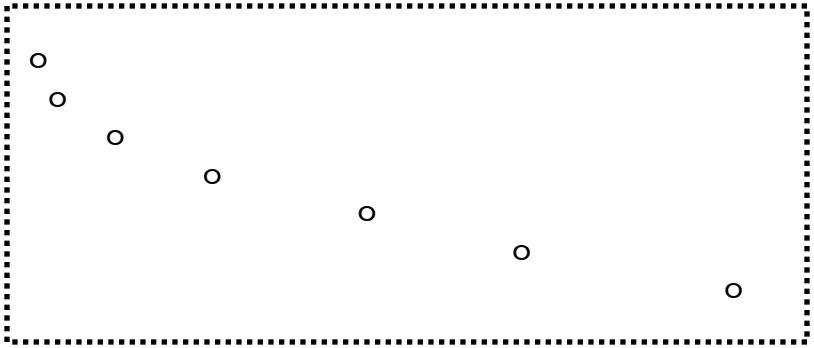
FormatMoney prints out money amounts in a proper way.

```
void
   formatMoney (double amt) {
   int iAmt, dollars, cents;
// Does write money in form $ddd.cc
// Need real input (amt >= 0.0)
   amt   = 100.0 * amt; // in cents
   iAmt  = JJS.doubleToInt (amt + 0.5);
   cents = iAmount % 100;
   JJS.outputString ("$"); //Optional
   JJS.outputDouble (dollars);
   JJS.outputString (".");
   if (cents <= 9) {
      JJS.outputString ("0");
   }//end if cents is small
   JJS.outputInt (cents);
}//end void method formatMoney
```

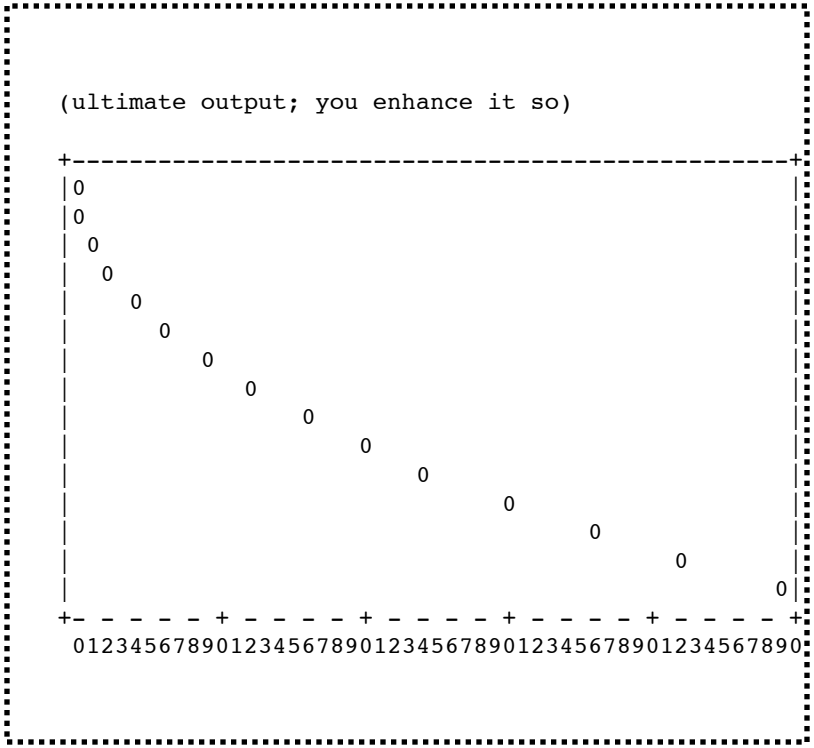
```
Enter dollars
1234.567
$1234.57

Enter dollars
1.2
$1.20

Enter dollars
0.1
$0.10

Enter dollars
0.995
$1.00
```

SidePlot, another way to use outRow

```
// Name An Onymous
void main5() {
   double x,y; //real values
   int iY;     //rounded y
// Does a side plot of y = f(x) using outRow
   JJS.start() {
   x = 0.0;
   while (x <= 6.0) {
   // Compute y and digitize it to iY
      y = 2*x*x; //f(x) is quadratic
      iY = JJS.doubleToInt (y + 0.5); // round
      outRow (iY, " ");               // blanks
      JJS.outputString ("o");         // do mark
      JJS.outputString ("\n");        // newline
      x = x + 0.5;
   }//end while
}//end void method main5
```



```
void outRow (int many, String marks) {
  // Does output a row of many marks
  // without using any local boxes
  // Not in APJS,as slots are changed!
     while (many != 0) {
        JJS.outputString (marks);
        many = many - 1;
     }//end while
  }//end void method outRow
```



Many ways to do outRow (including the For and recursion).

```
// Name An Onymous
void main6() {
   int num;    //number of repetitions
   String str; //string to be repeated
// Does show many ways to do outRow method
   JJS.start() {
   JJS.outputlnString ("Enter an integer ");
   num = JJS.inputInt ();
   JJS.outputlnInt (num);

   JJS.outputlnString ("Enter any string ");
   str = JJS.inputString ();
   JJS.outputlnString (str);

   outRow (num, st);

   JJS.outputString("\n");
}//end void method main6
```

```
Enter an integer
7
Enter a string
----+

----+----+----+----+----+----+----+
```

```
void
outRow (int many, String marks) {
// Does output a row of many marks
// Uses the For loop
    for (int i = 0; i < many; i++) {
         JJS.outputString (marks);
    }//end for
 }//end void method outRow
```

In the following version, note that outRow calls itself (called recursion) but with different slots each time. Try to trace it.

```
void
outRow (int many, String marks) {
// Does output a row of many marks
// using recursion, calling itself!
// Need (many >= 0)
   if (many != 0) {
      JJS.outputString (marks);
      outRow (many - 1, marks);
      }//end while
}//end recursive method outRow
```

Problems on Routines

1.More SpellOut

Enhance the SpellOut routine to spell more values , up to 10.

2. SpellTeen Stubs

Complete the stubs to spell out the teens (11 to 19).

3. SpellTens

ReUse spellDigit and spellTeen to spell out ints from 0 to 99.

4. Format Gas

Change FormatMoney to format any reals to three decimal points.

5. Break Greetings

Use outRow to display a greeting (say "Happy Anniversary")
by writing many lines, but putting a gap after every fifth one.

6. OutGreetPair

ReUse outRow to output a string greeting, two to a line.
For example "Happy Birthday" can be written 21 as 10 full lines
with one half line.

Short Routines

It may be convenient to write various common commands in a short way.
For example, rather than writing

JJS.outputString(s) or System.out.print(s)

a shorter way would be to create a void method outStr and Re-call it often:

outStr(s);

Write void methods for the following

outStr(s) which is a simple way to call to output a specified string.

outLnStr(s) which outputs a string followed by a new line

outLine() which is a short way to output a new line,

outLnInt(I) which outputs an int followed by a new line

outIntSpace(i) which prints an int i followed by a space (not a new line)

outPrompt(s) which prints out the specified prompt (followed by new line)

outReal(r) which prints out a real value r followed by a space

Do also some of your own.

HelpIFul Routines

Write routines to display the following;
You need not include all the details
(exclude the repeated parts)

1. **`outWeek(day)`** which prints out "Sunday" for day 0, etc..

2. **`outWeek1(day)`** which outputs "Sunday" for day 1, etc…
   Hint: ReUse!

3. **`outMonth(num)`** which prints out "January" for day 1, etc..

4. **`outMorse(digit)`** which prints out a given digit in Morse code.

BIG-DIGITS

1. DrawADigit

Write a method drawDigit(d) which draws the decimal digit specified
but on it's side consisting of "Bixels" 8 high.
For example drawDigit(4) draws the following:

```
    0000
    0
00000000
```

2. drawPercent

Write another method drawPercent(p) which draws any percentage p (from 0 to 100) as a series of 1, 2, or 3 such bigDigits on their side.

3. drawInt

Write another method drawInt(i) which draws any given integer i as a bigInt.

4. drawChr(c)

Write many method to draw a given capitalized character from "A" to "Z";
This may be best done with a team.

5. More For BigDigits;

Include the outPeriod(), outDollar(), outSpace, outSpaces(n), etc

More draw Routines

1. `drawBigTime(mil)` for given military time mil.

2. `drawBigLowerLetters(low)`: bigger team
for all 16 lower case letters,
But done with a total height of 12 bixels.

3. `drawBigLetter(x)`: team project
for all 26 capital letters of the alphabet.

4. `drawADie(dots)`, showing any of the 6 faces, such as

```
+-------+
| O O O |
|       |
| O O O |
+-------+
```

5. DrawATotem Pole