Structure Charts

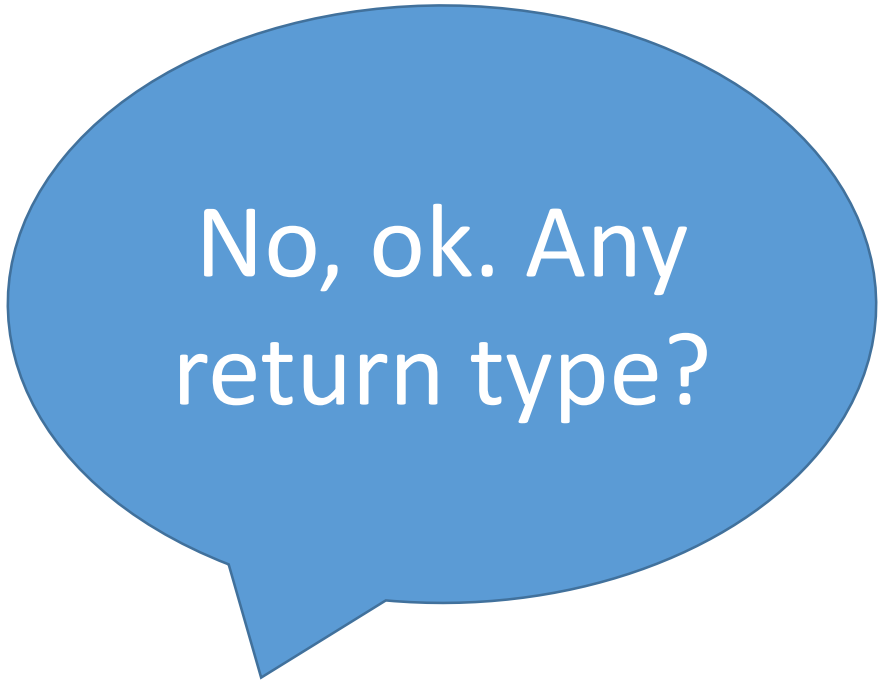
But first,
a review of
method
calling.

A method signature:

```
public void bob()
```

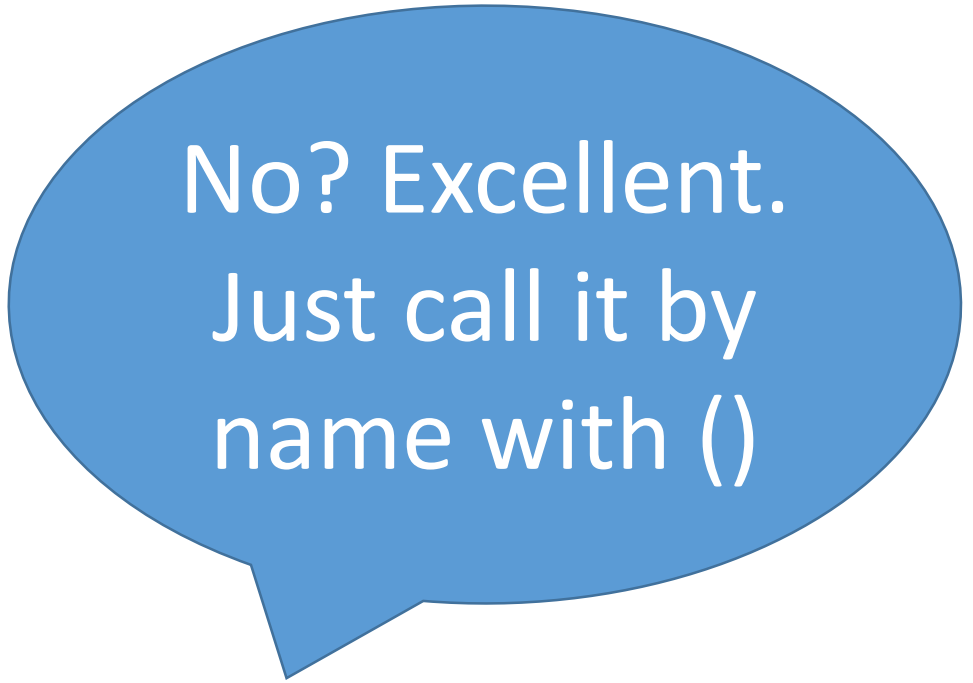
A method signature:

```
public void bob()
```



A method signature:

```
public void bob()
```



A method signature:

```
public void bob()
```

Method call:

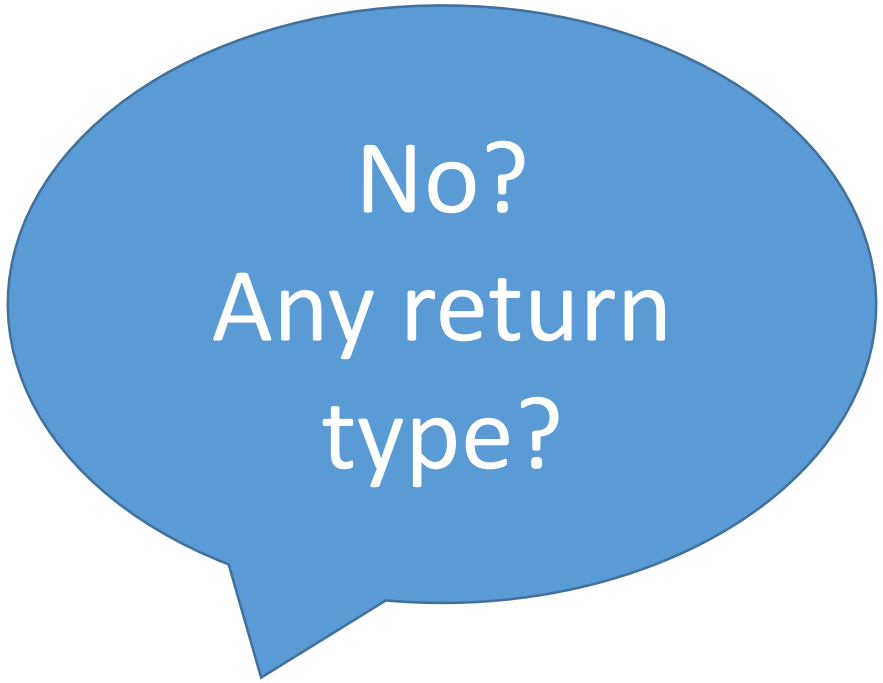
```
bob();
```

A second method signature:

```
public int turtle()
```

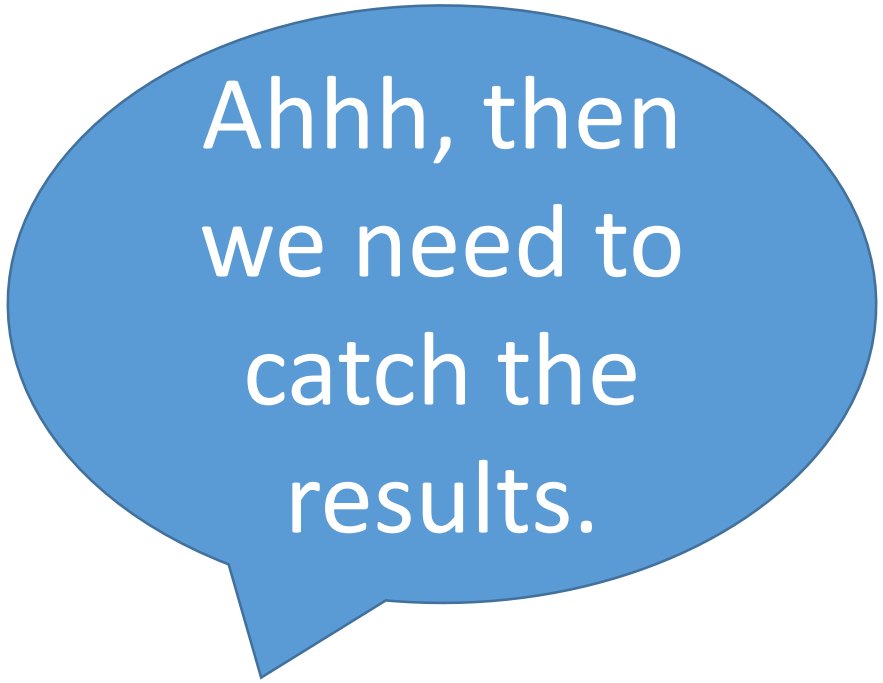
A second method signature:

```
public int turtle()
```



A second method signature:

```
public int turtle()
```



A second method signature:

```
public int turtle()
```

Method Call:

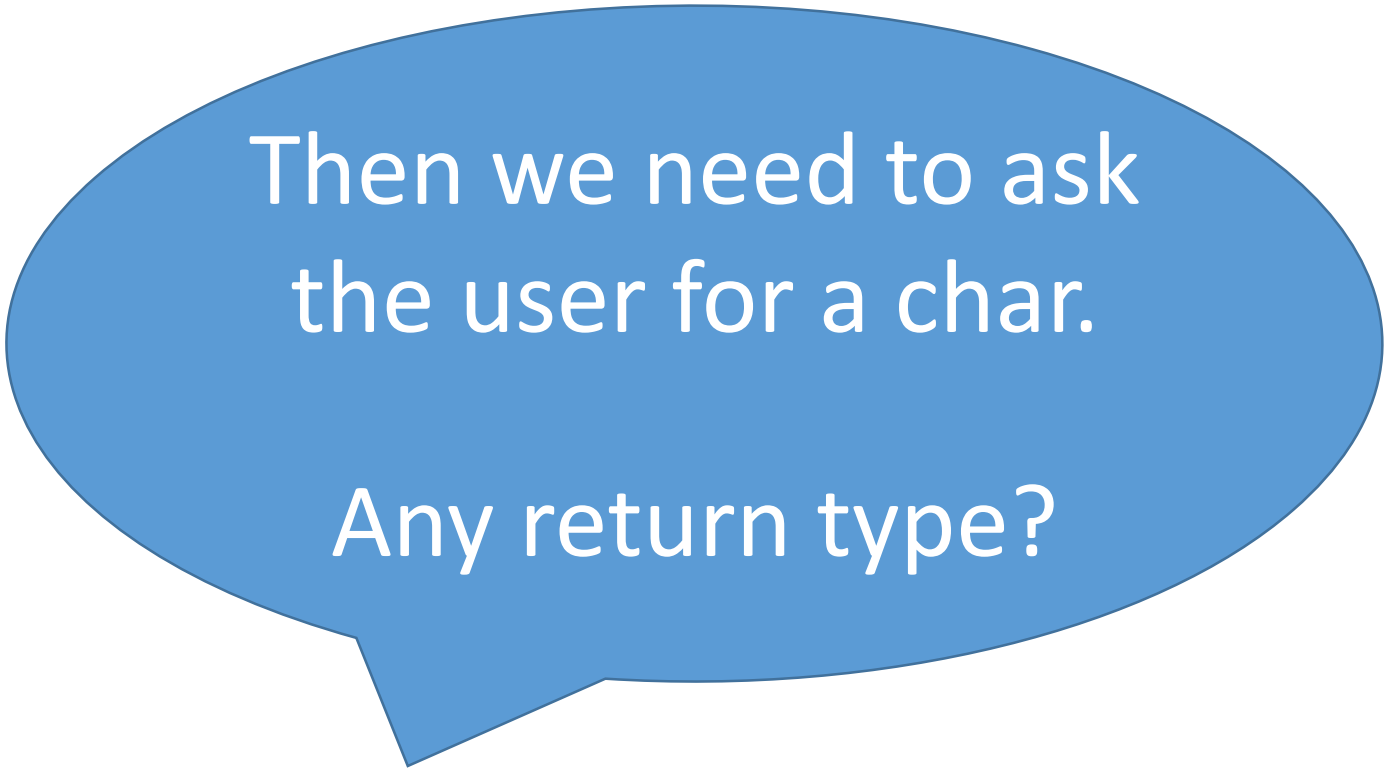
```
int t = turtle();
sop("The answer is "+t);
```

A third method signature:

```
public double newt(char c)
```

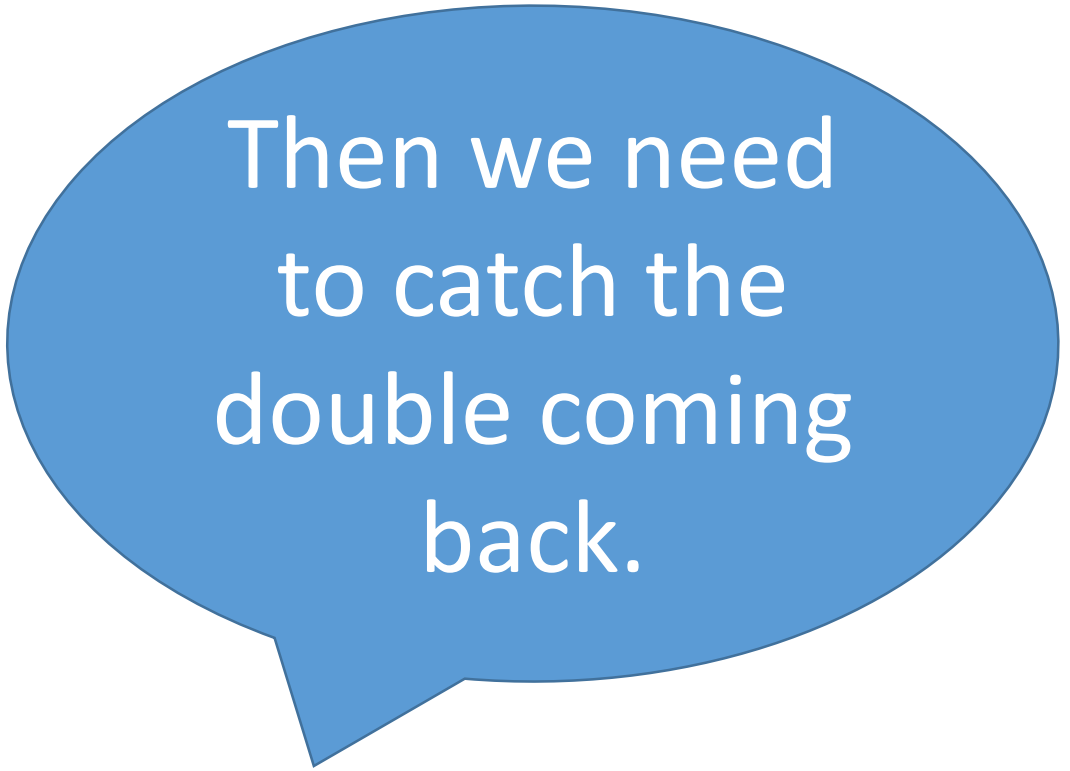
A third method signature:

```
public double newt(char c)
```



A third method signature:

```
public double newt(char c)
```

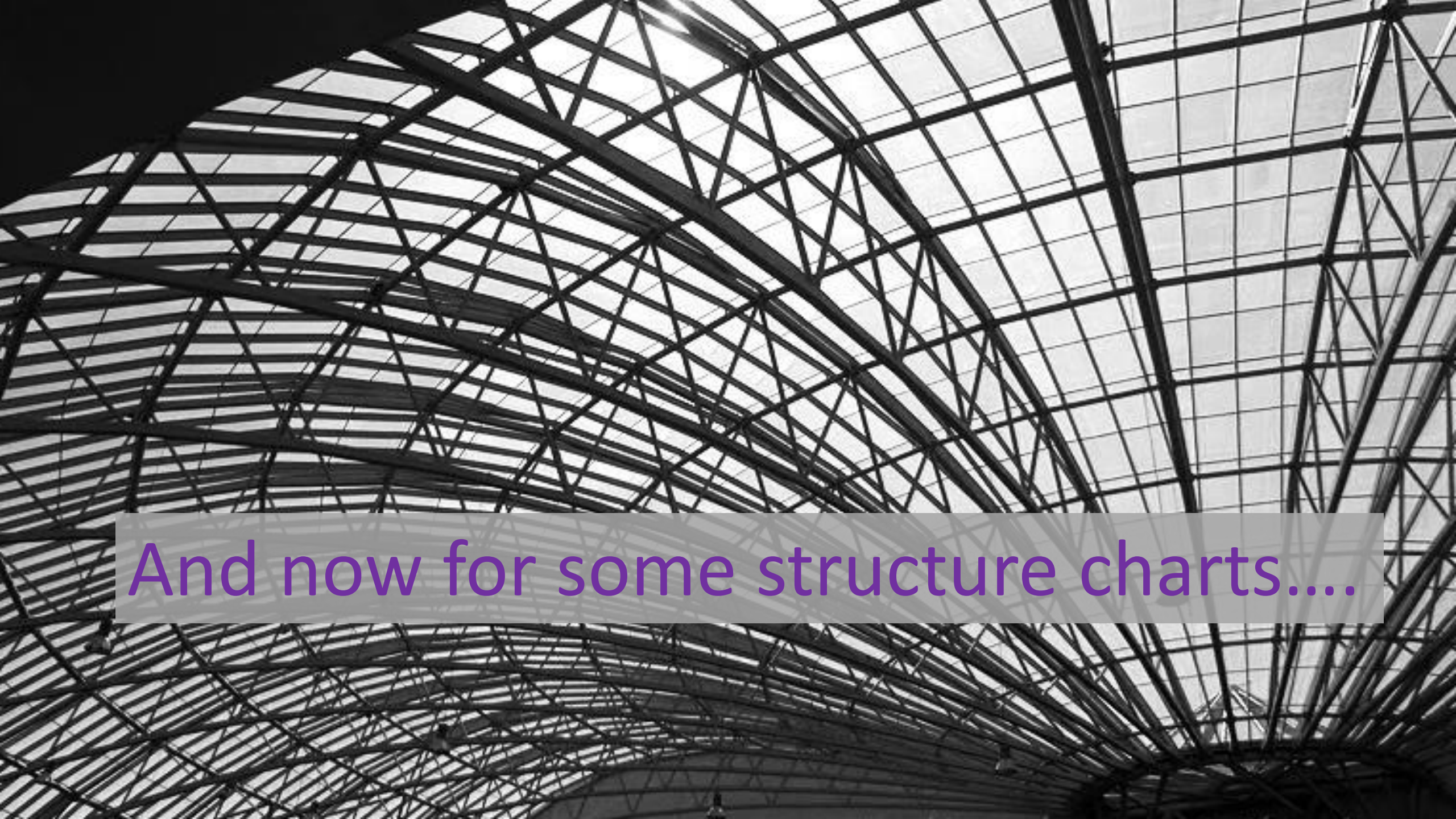


A third method signature:

```
public double newt(char c)
```

Method Call:

```
char x = IO.inputChar("Letter?");
double ans = newt(x);
sop("The answer is " + ans);
```

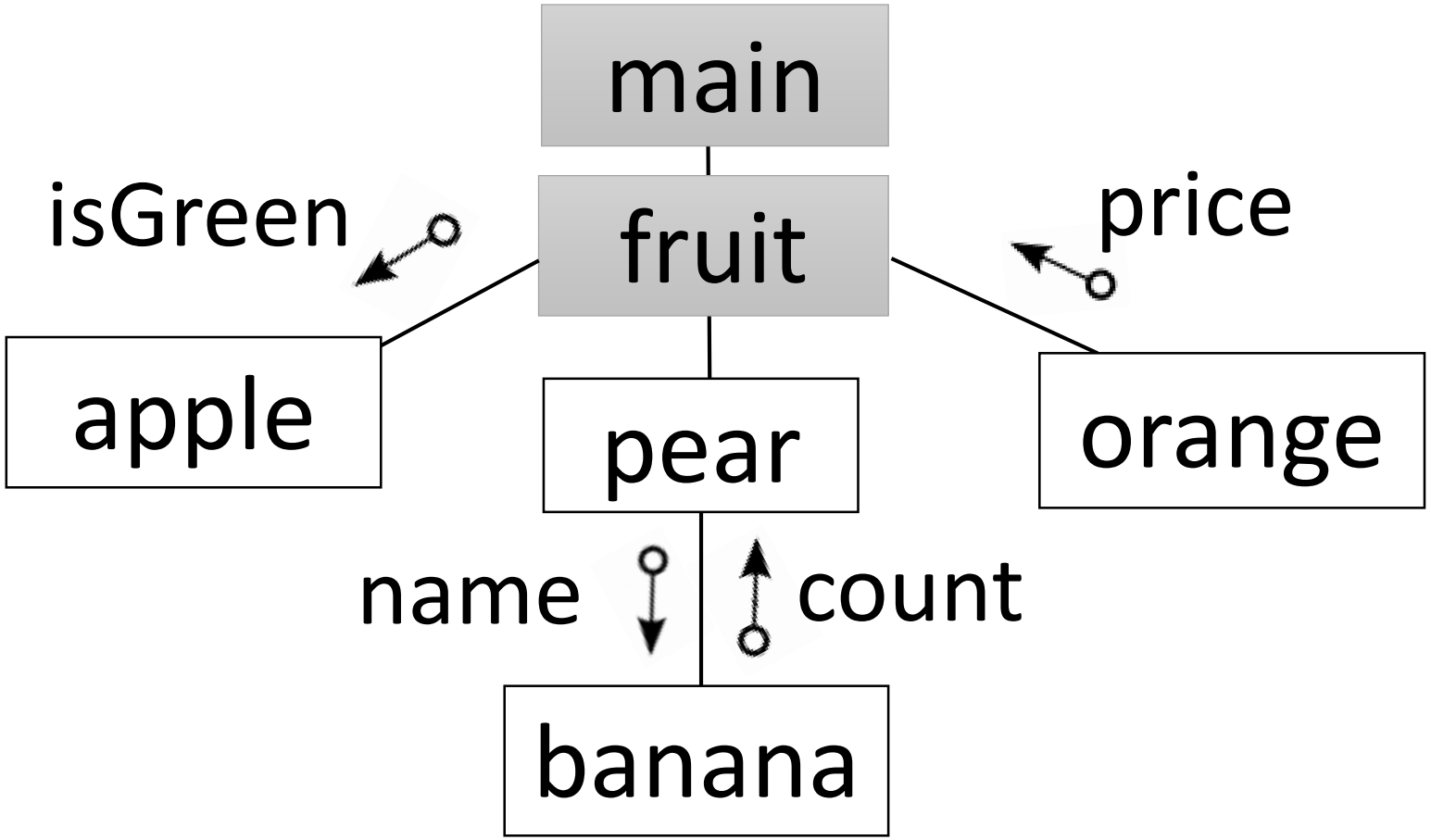

And now for some structure charts….

Structure Charts:

- Used to design the methods.
- During the Design Phase (Criteria B).

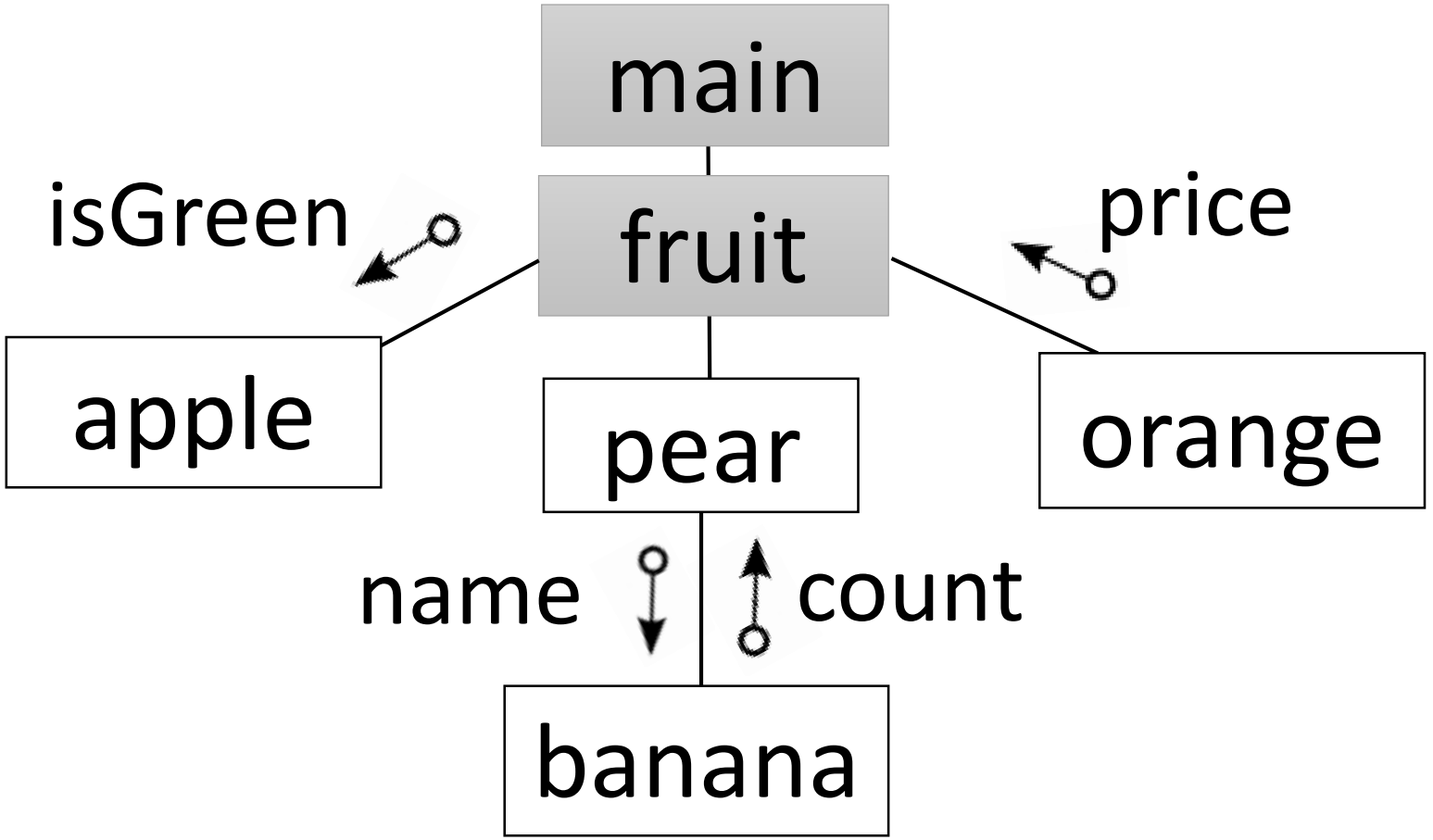
Structure Chart Rules:

- Each method is in a box.
- Methods called by other methods are shown under their caller.
- Connect methods with lines (not arrows)
- Show the input (parameters) and the output (return type) with arrows.



How many methods?

What's the constructor name?



How many methods?

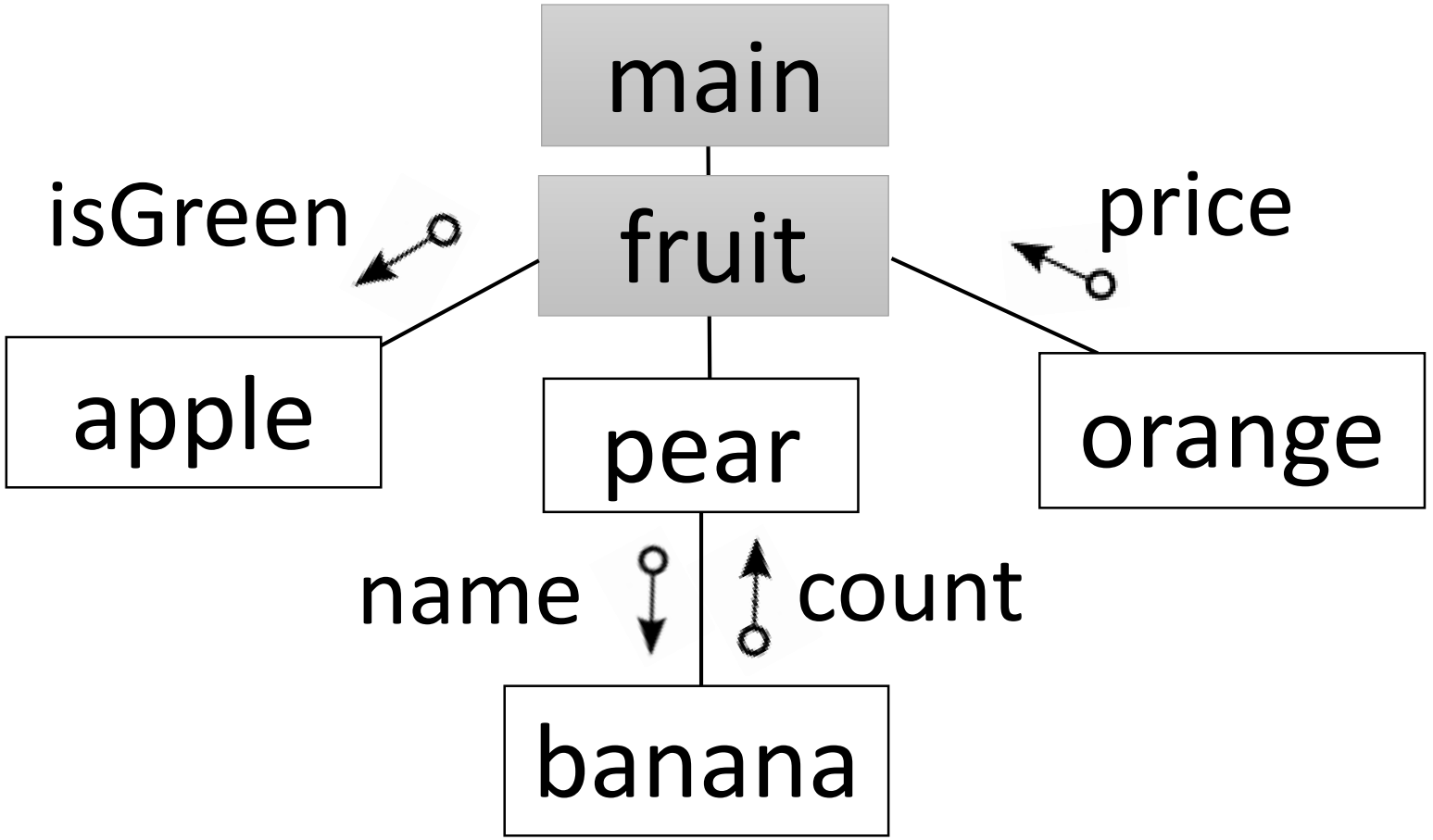
4

Don't count constructor and main

What's the constructor name?

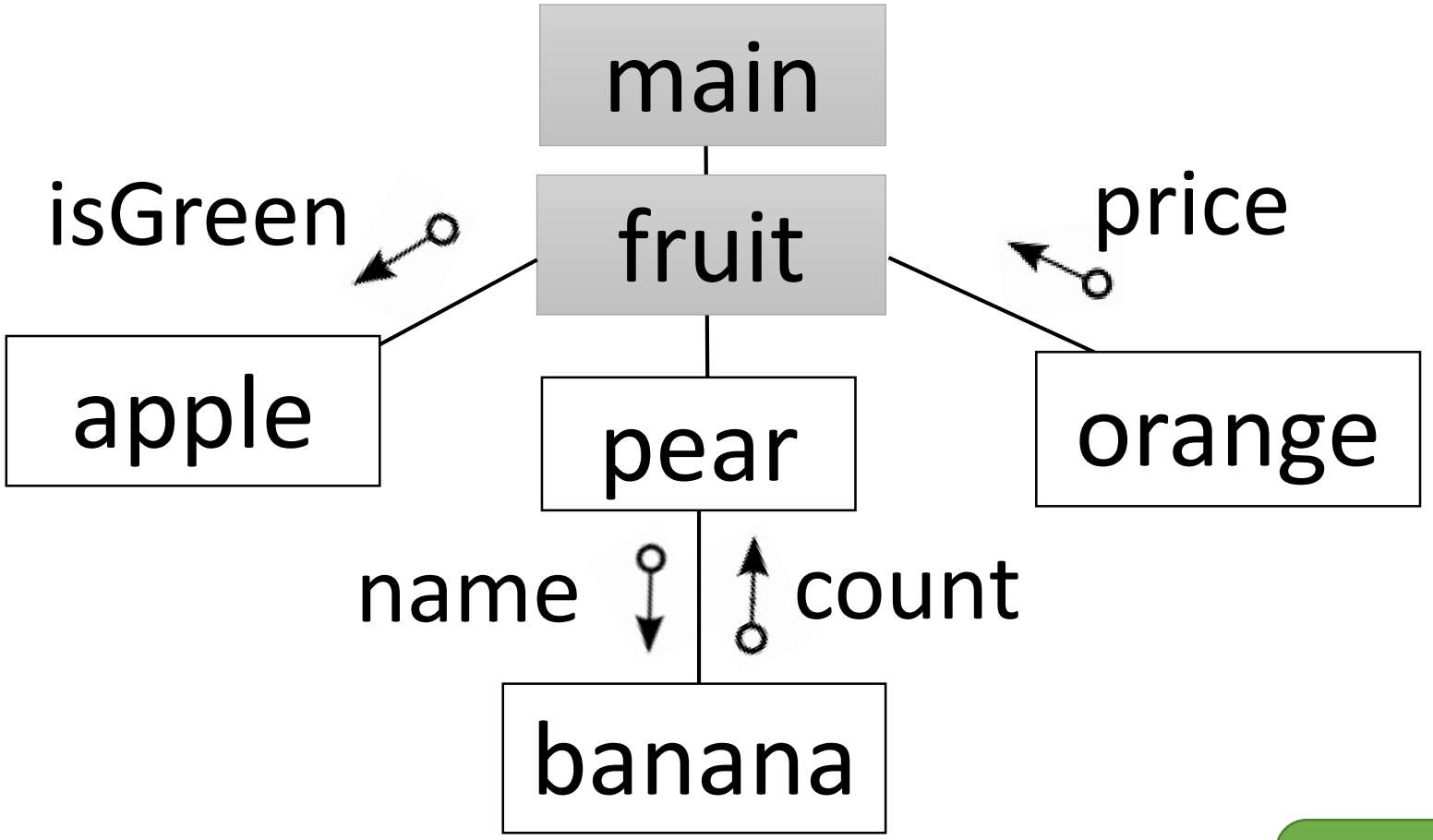
fruit

It's called by the main method



What method calls orange?

What method calls banana?



What method calls orange?

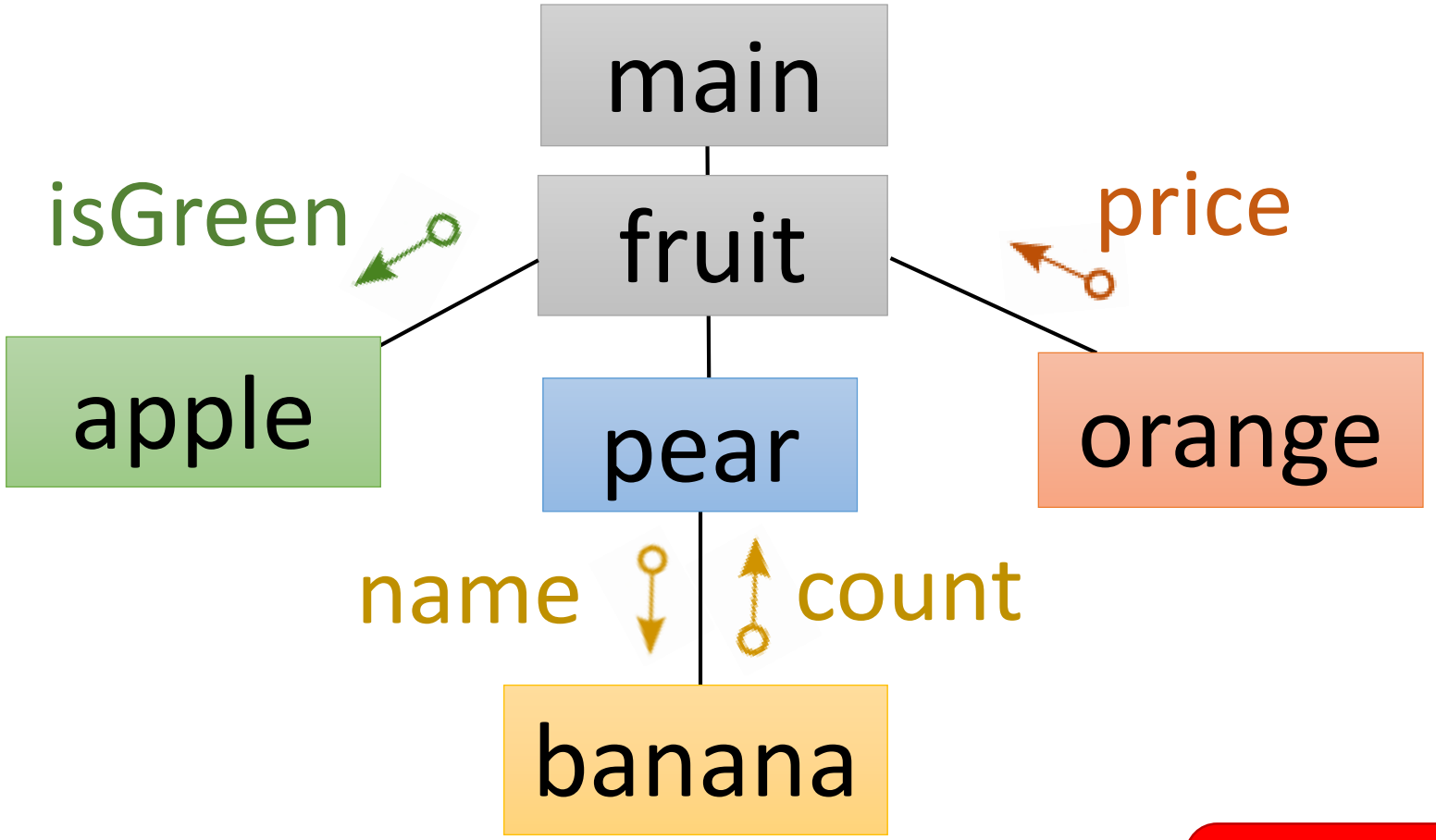
fruit

The constructor

What method calls banana?

The method on top calls the method below

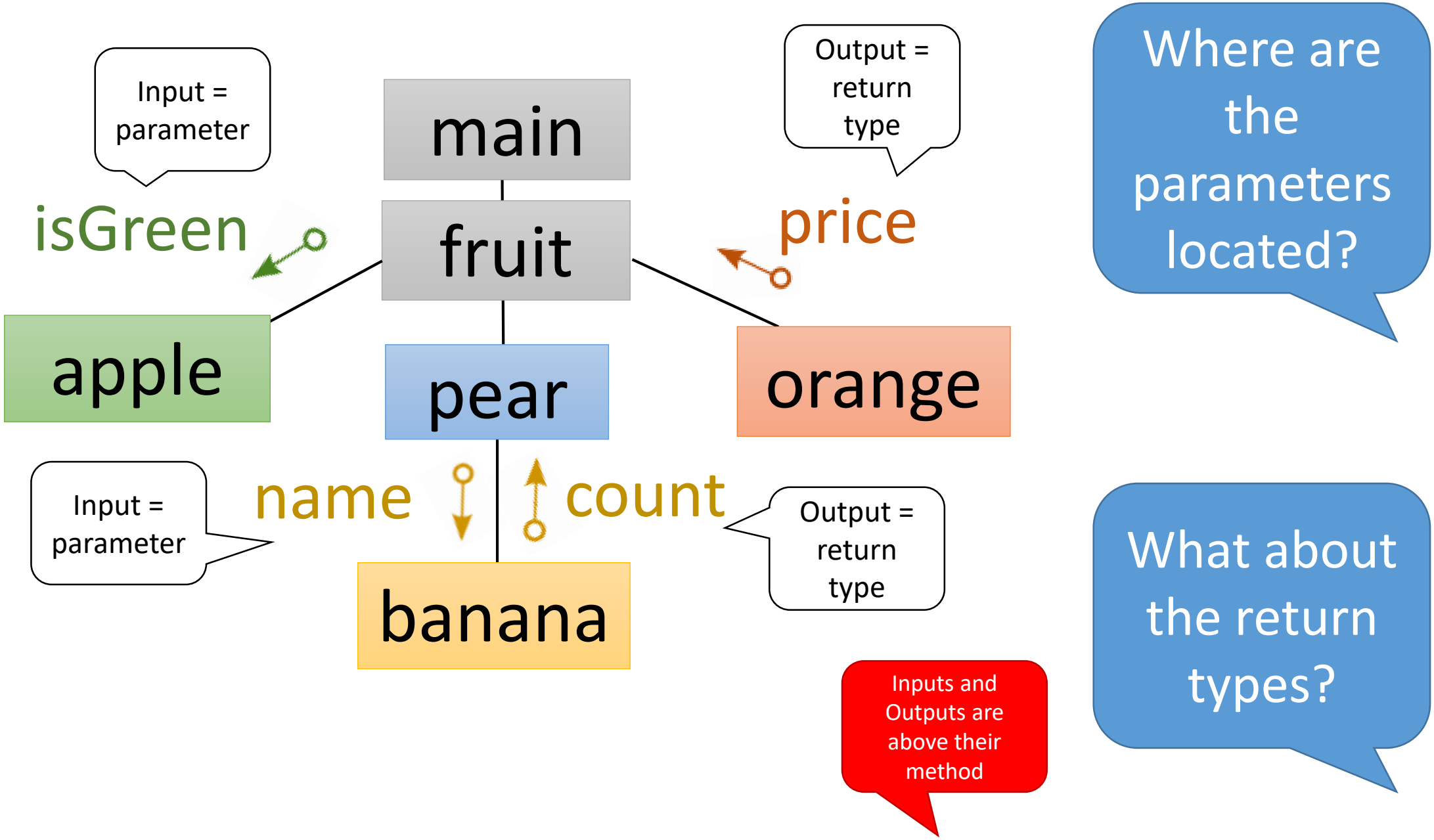
pear



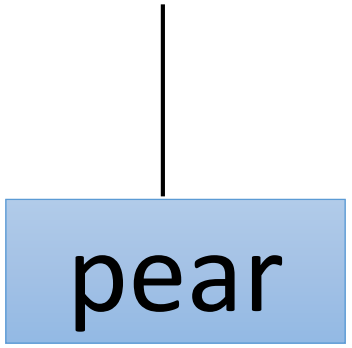
Where are the parameters located?

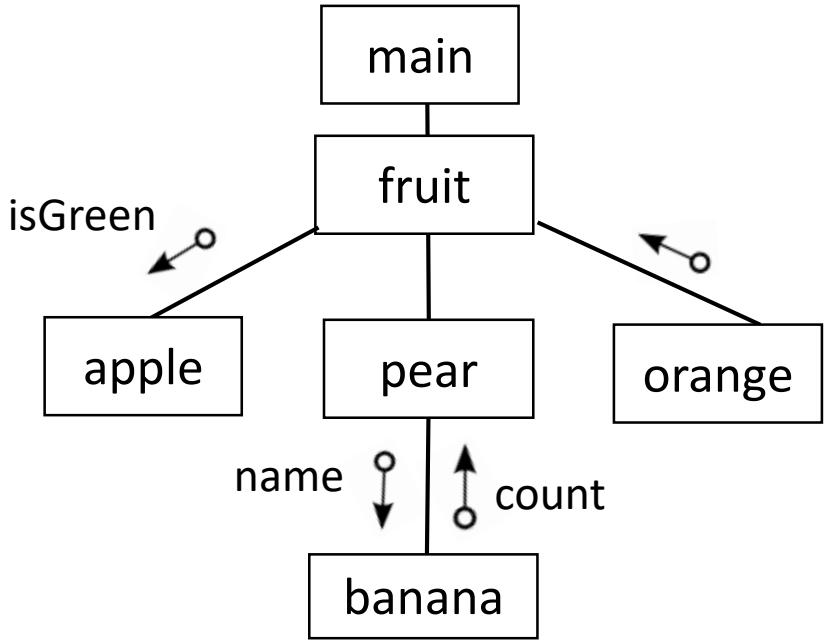
What about the return types?

Inputs and Outputs are above their method


Input = parameter
main
Output = return type
isGreen
fruit
price
apple
pear
orange
Input = parameter
name
count
Output = return type
banana
Inputs and Outputs are above their method
Where are the parameters located?
What about the return types?

Let's build the method signature for pear.

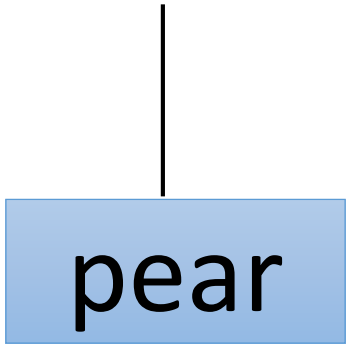


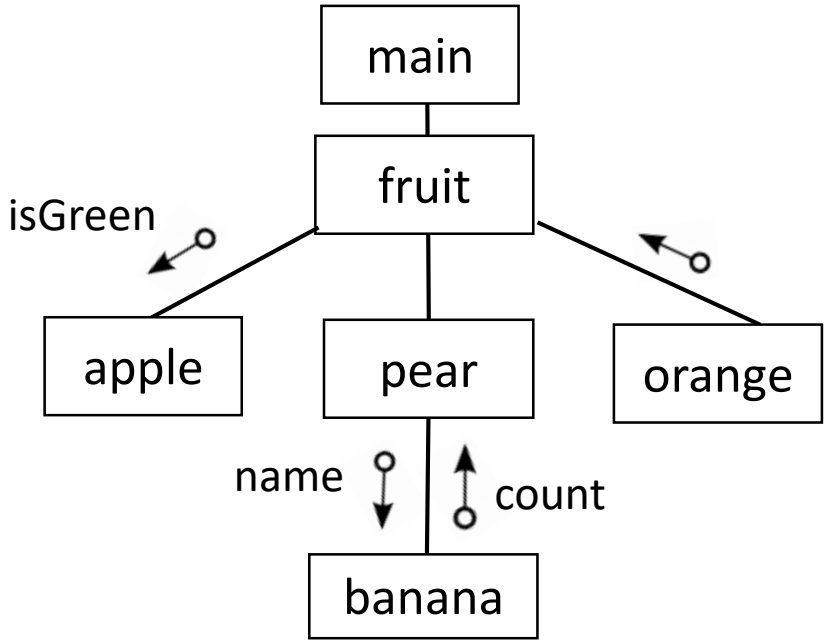


_____ _____ _____(_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for pear.

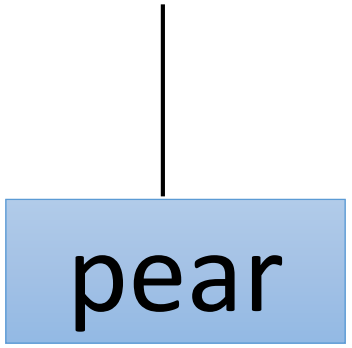


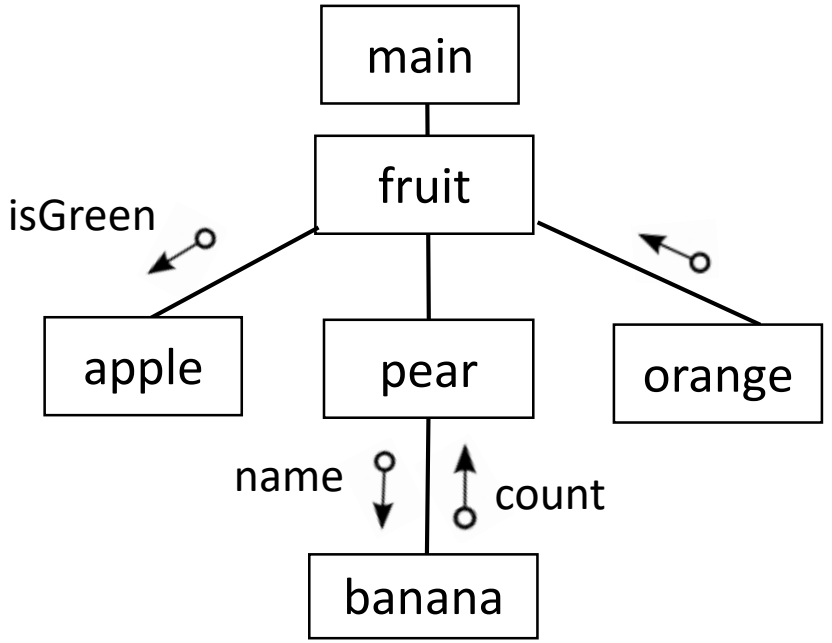


public _____ _____ _____(_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for pear.

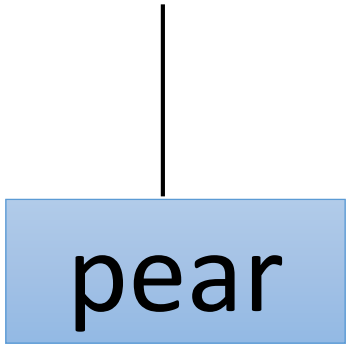


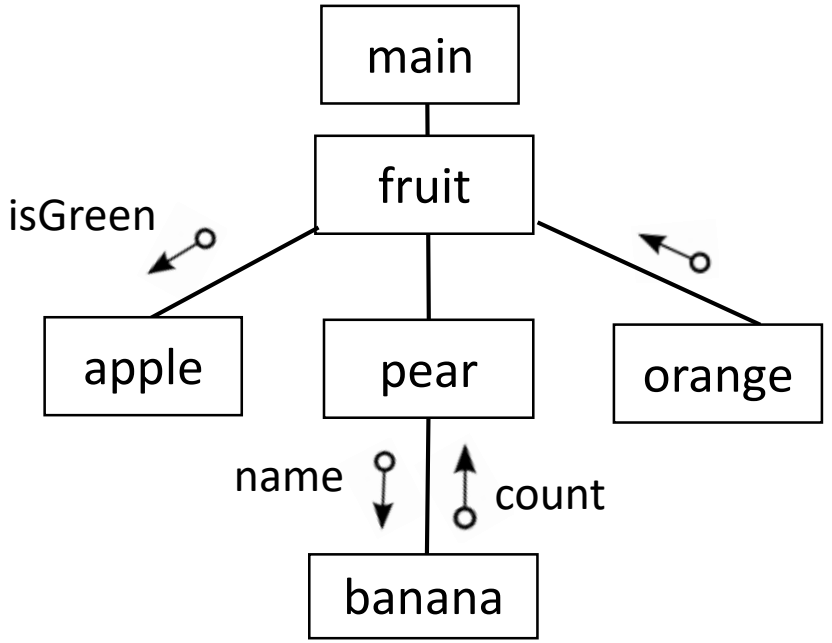


public _____ void _____ _____ (_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for pear.

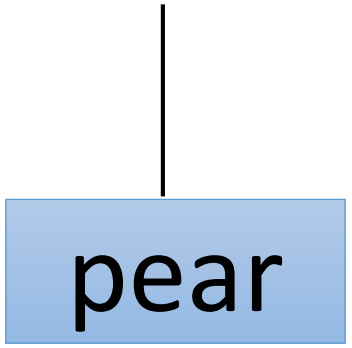


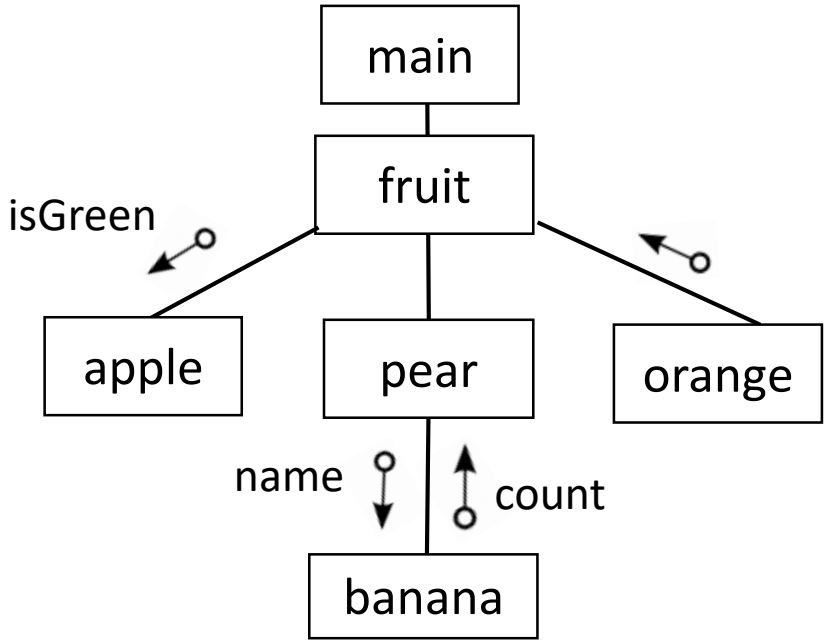


public _____ void _____ pear _____ (_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for pear.

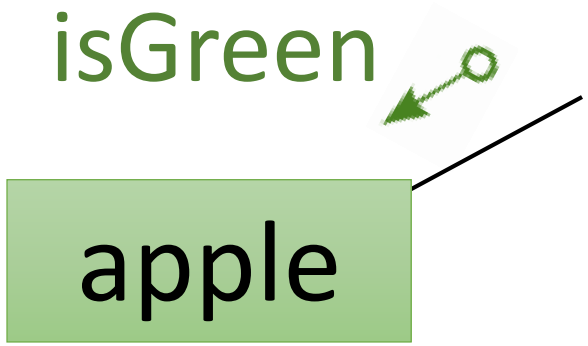


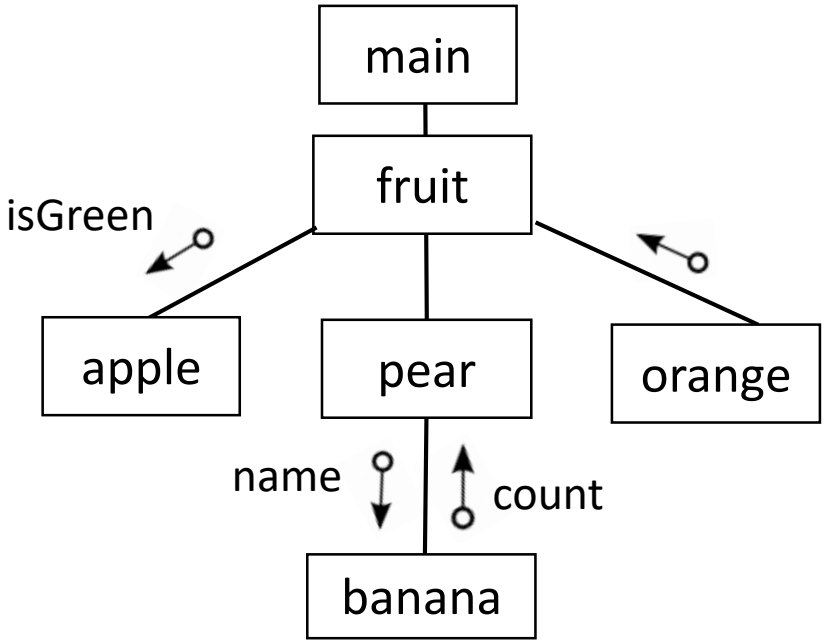


public _____ void _____ pear _____ ([blank] _____ [blank] _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for isGreen.





_____ _____ _____(_____ _____){

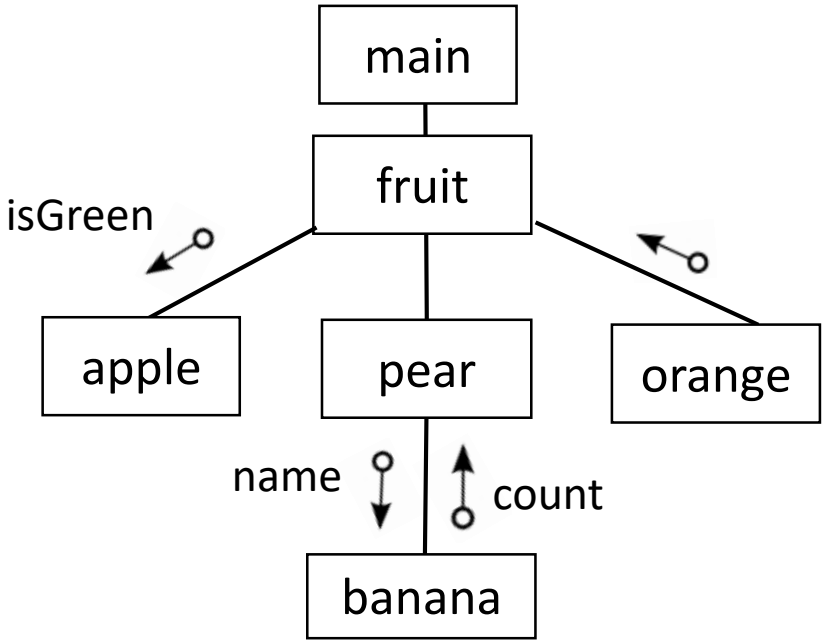
start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for isGreen.

"is" in front means boolean

isGreen

apple



public _____ _____ _____(_____ _____){

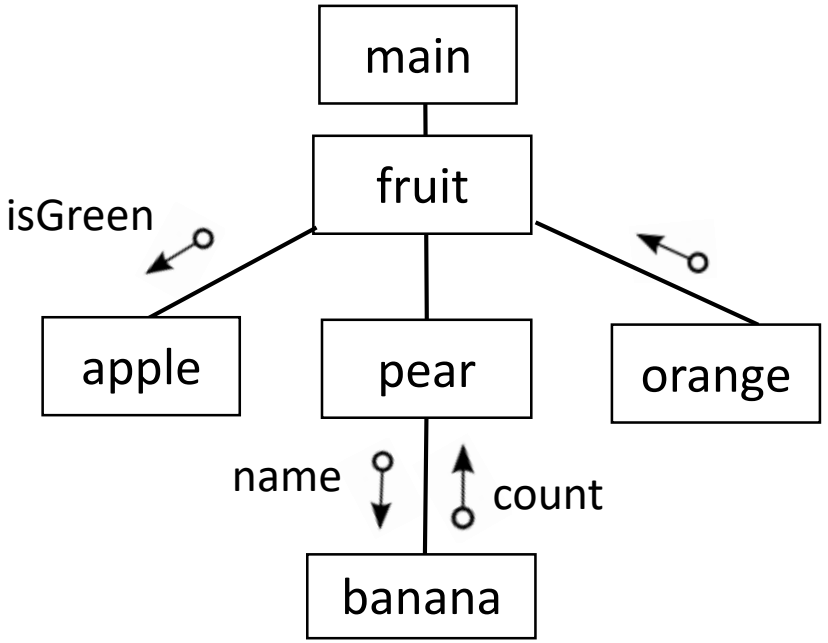
start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for isGreen.

"is" in front means boolean

isGreen

apple



public _____ void _____ _____ (_____ _____){

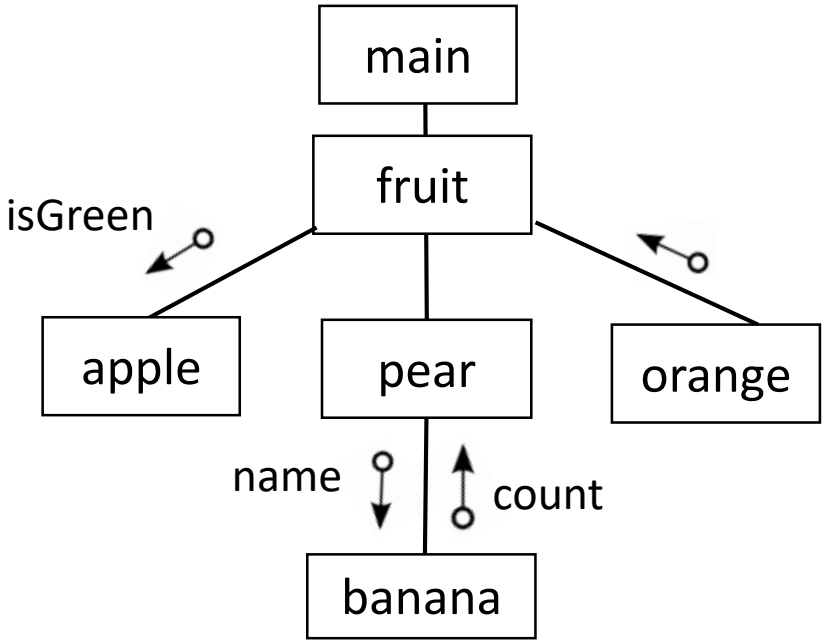
start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for isGreen.

"is" in front means boolean

isGreen

apple



public _____ void _____ apple _____ (_____ _____){

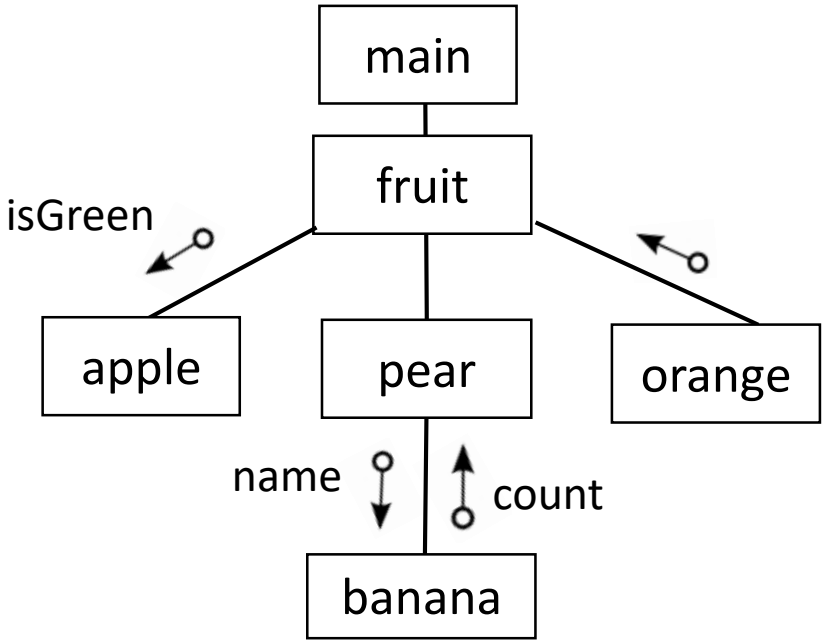
start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for isGreen.

"is" in front means boolean

isGreen

apple



public _____ void _____ apple _____ (boolean _____ _____){

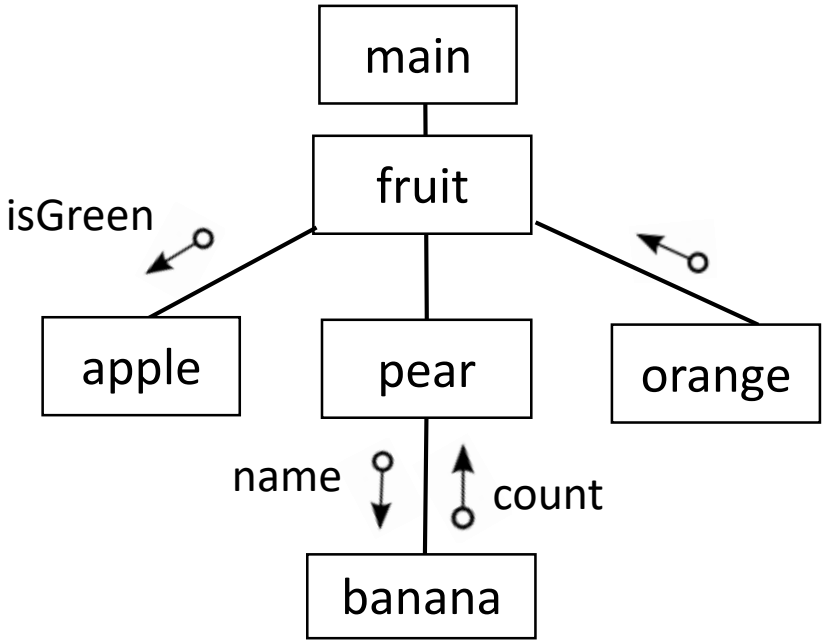
start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for isGreen.

"is" in front means boolean

isGreen

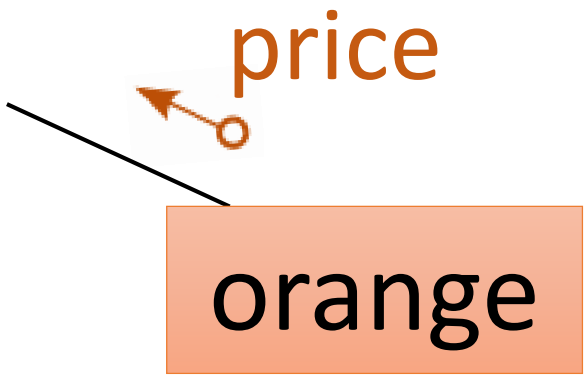
apple

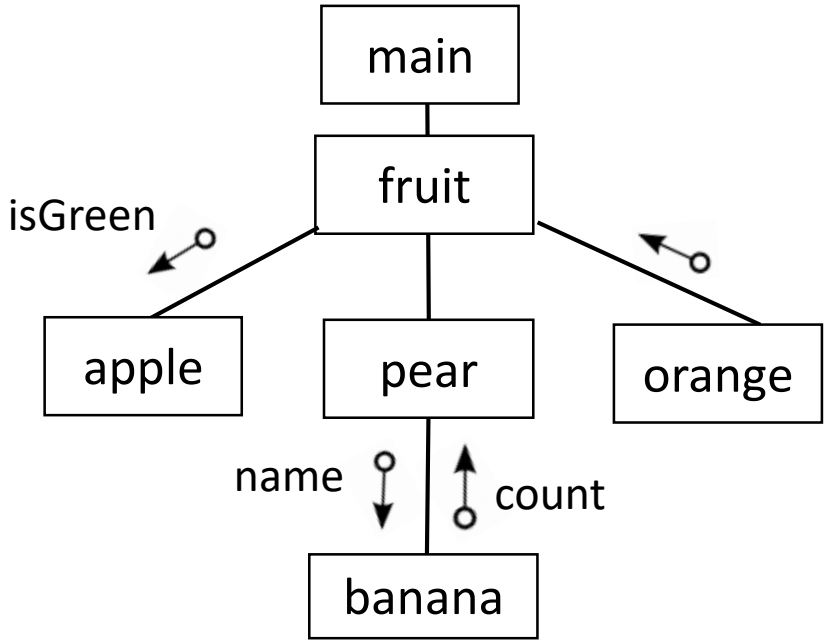


public _____ void _____ apple _____ (boolean _____ isGreen _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for orange.

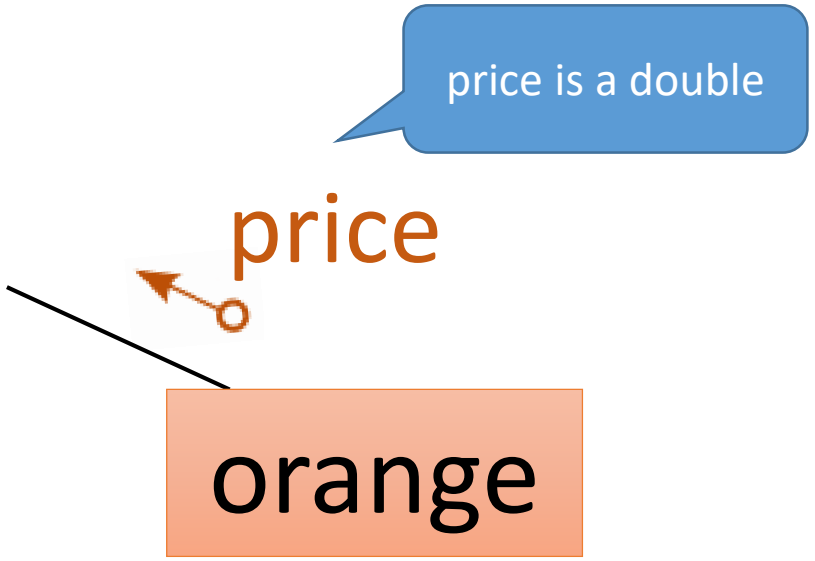


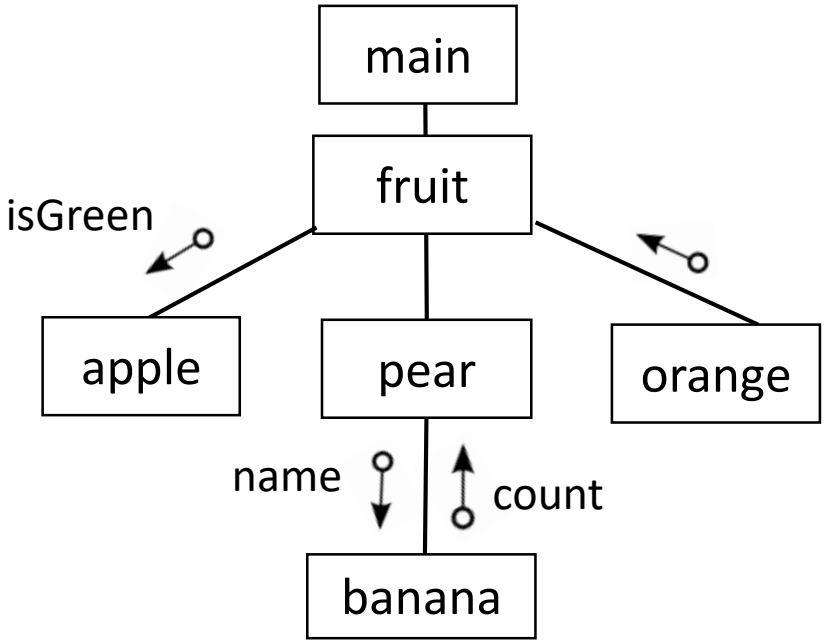


_____ _____ _____(_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for orange.

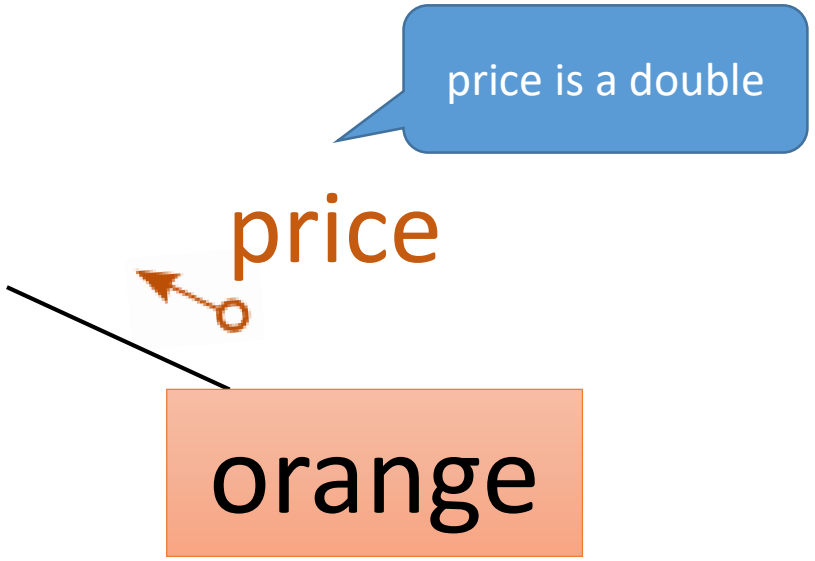


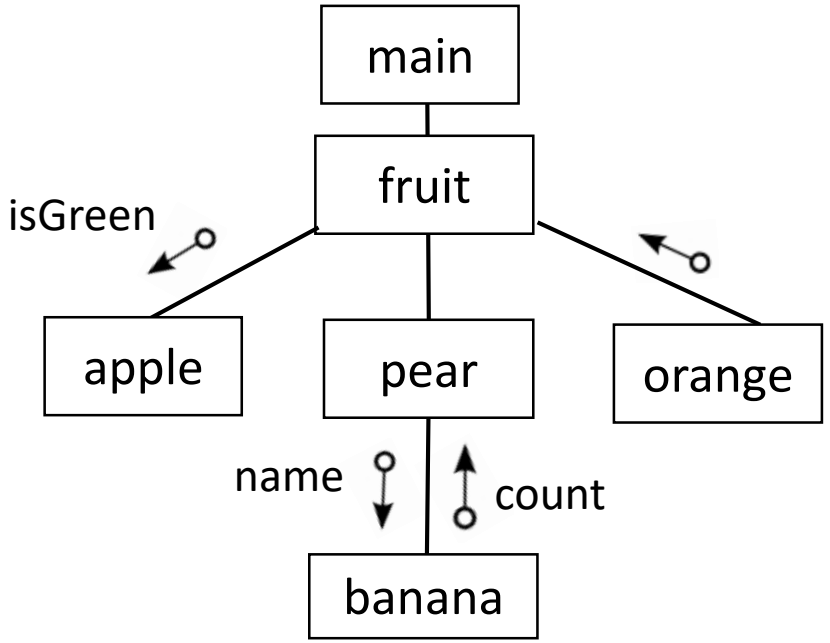


public _____ _____ _____(_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for orange.

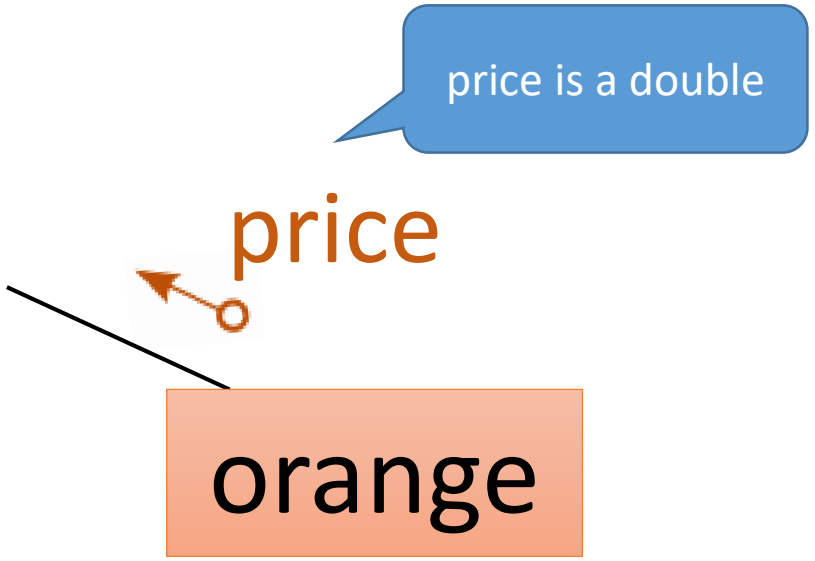


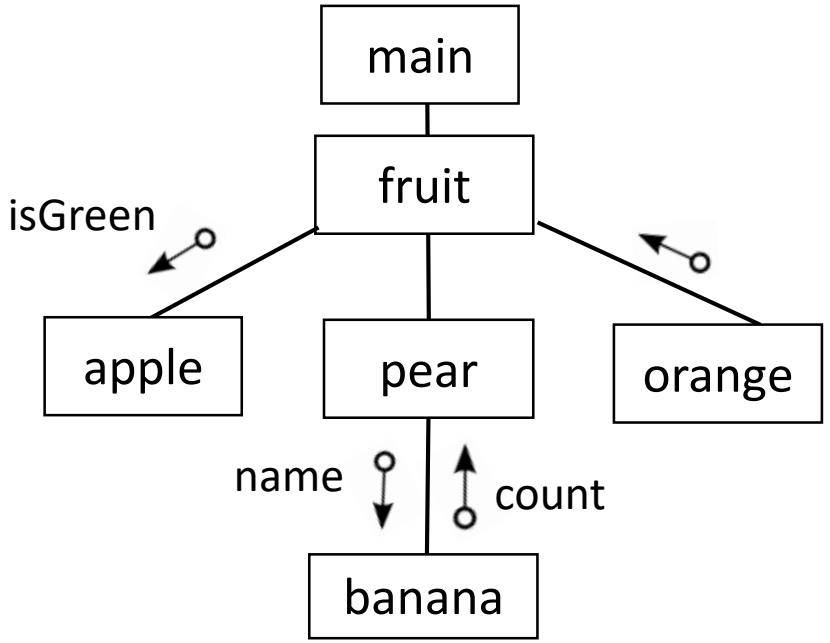


public _____ double _____ _____(_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for orange.





public _____ double _____ orange _____ (_____ _____){

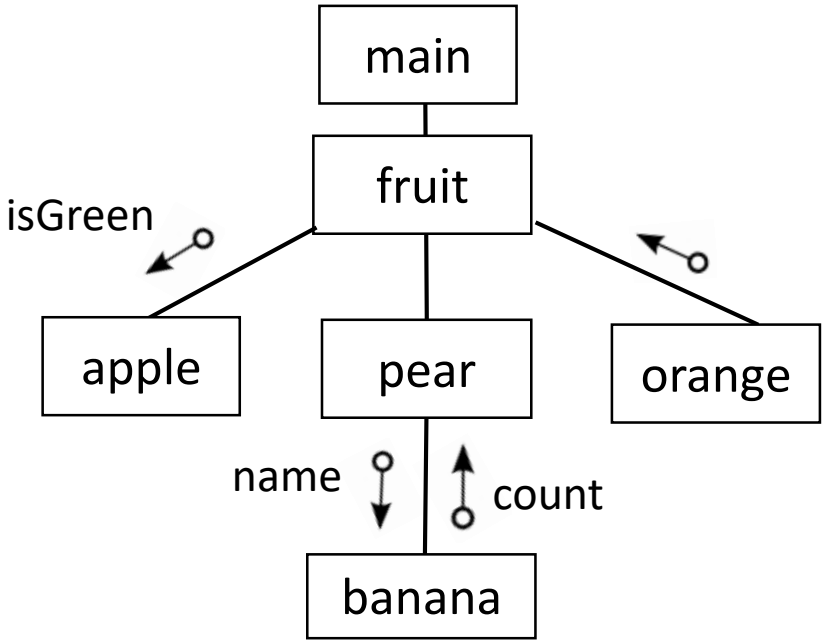
start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for orange.

price is a double

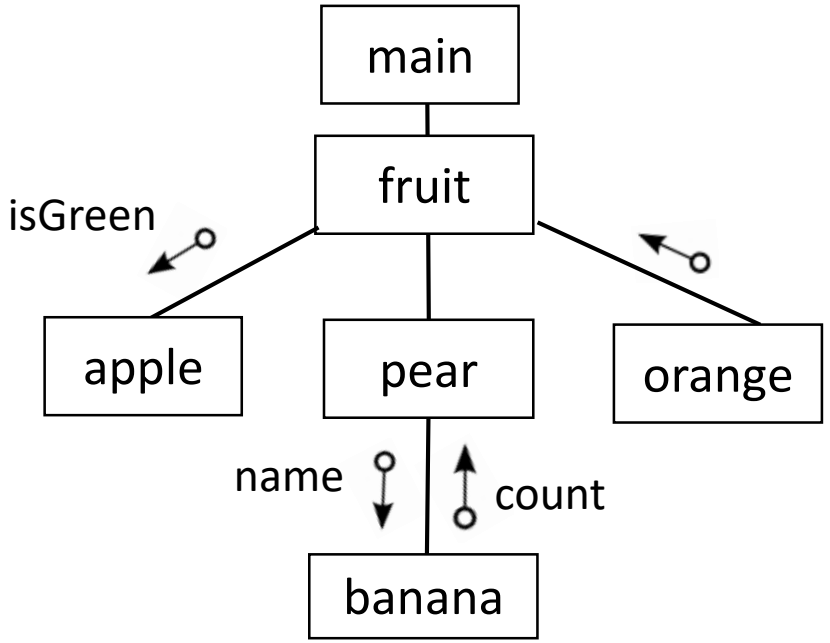
price

orange



public	double	orange	([blank]	[blank]	){
start word	*return type (out)*	*method name*	*param type*	*name (in)*	

Let's build the method signature for banana.



_____ _____ _____(_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for banana.

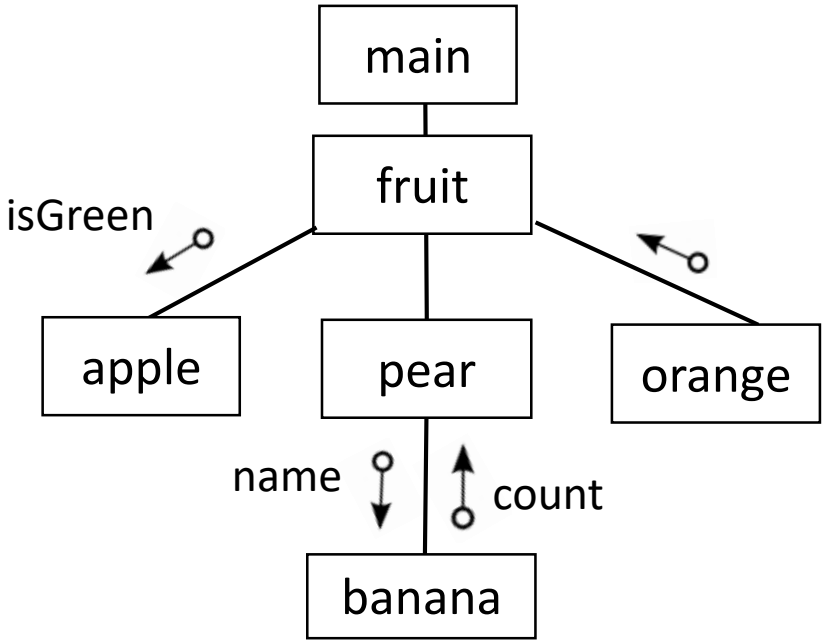
String

int

name

count

banana



public _____ _____ _____(_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for banana.

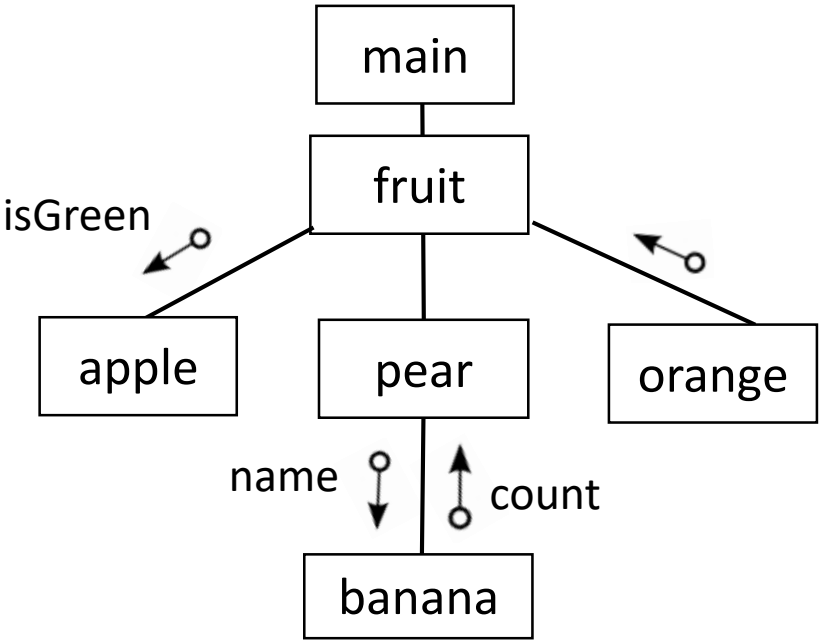
String

name

int

count

banana



public _____ int _____ _____ (_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for banana.

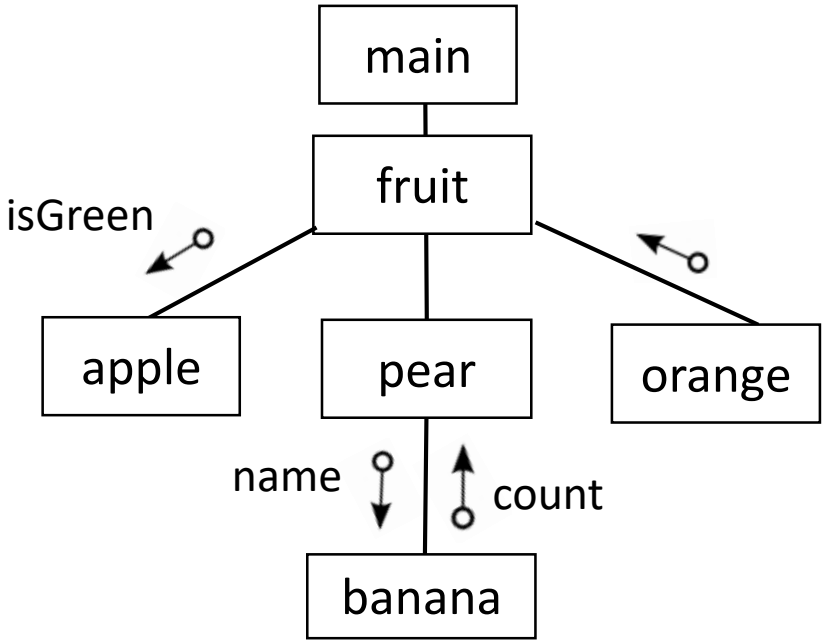
String

name

count

int

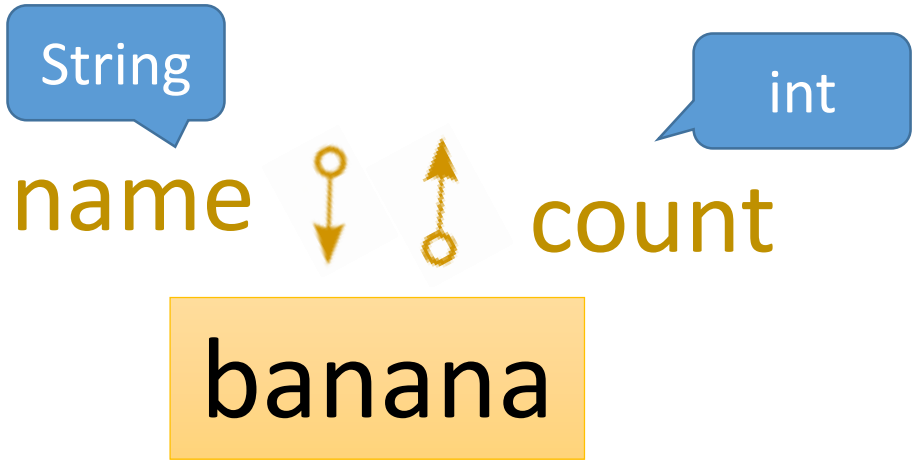
banana

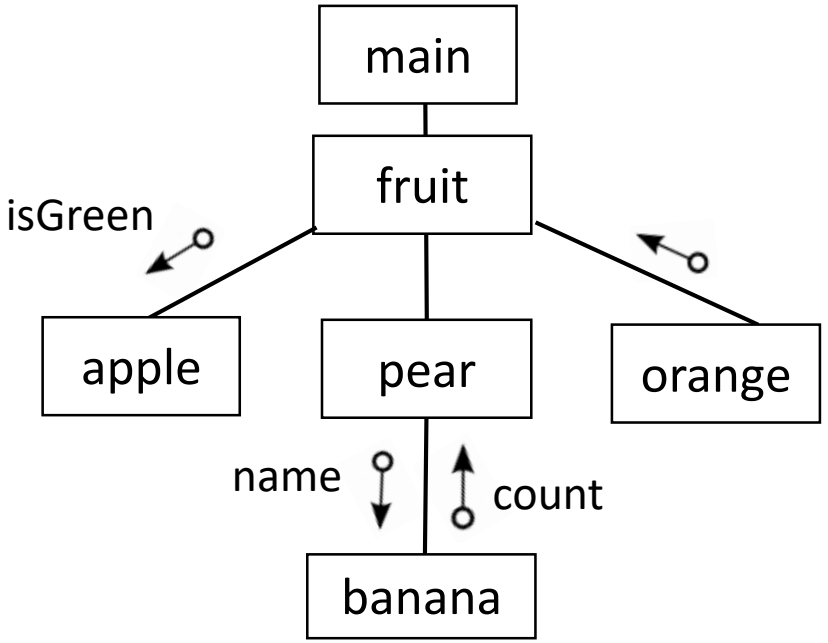


public _____ int _____ banana _____ (_____ _____){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for banana.





public int banana (String){

start word *return type (out)* *method name* *param type* *name (in)*

Let's build the method signature for banana.

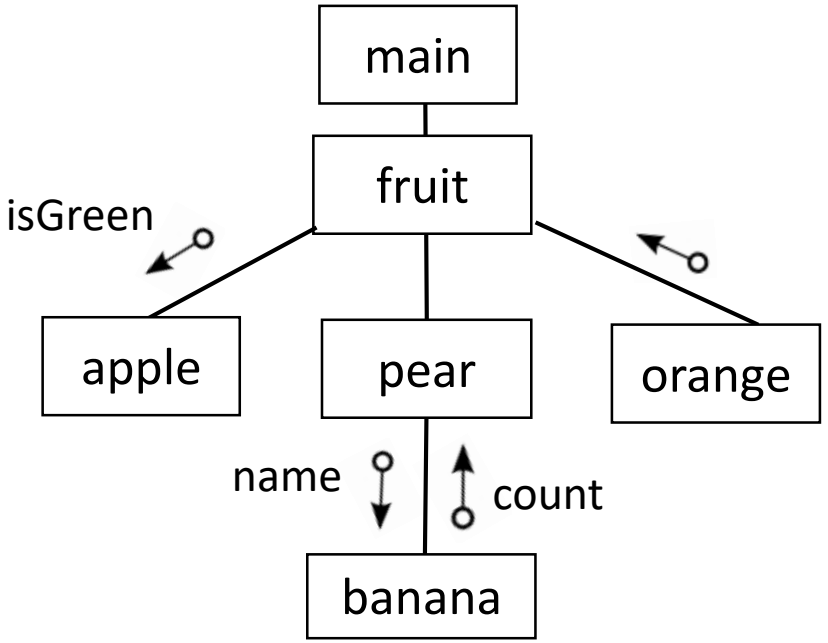
String

int

name

count

banana

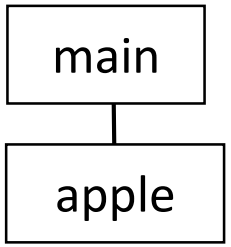


public	int	banana	(String	name	){
start word	*return type (out)*	*method name*	*param type*	*name (in)*	

```
public class apple
{
    public static void main (String args[])
    {  new apple ();
    }

    public apple ()
    {   System.out.println ("Welcome to apples!");
        seeds ();
    }

    public void seeds ()
    {   int amount = newtree ();
        printnewforest (amount);
    }

    public int newtree ()
       {    return (int) Math.random () * 9;
       }

       public void printnewforest (int amount)
       {  for (int i = 0 ; i < amount ; i++)
                System.out.print ("*");
       }
}
```

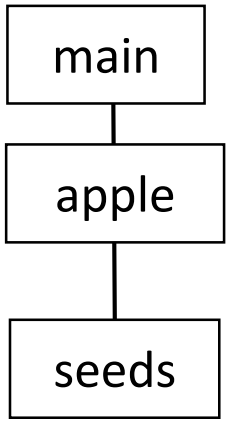
Draw a structure chart for this class.

```
public class apple
{
    public static void main (String args[])
    {  new apple ();
    }

    public apple ()
    {   System.out.println ("Welcome to apples!");
        seeds ();
    }

    public void seeds ()
    {   int amount = newtree ();
        printnewforest (amount);
    }

    public int newtree ()
       {    return (int) Math.random () * 9;
       }

       public void printnewforest (int amount)
       {  for (int i = 0 ; i < amount ; i++)
                System.out.print ("*");
       }
}
```



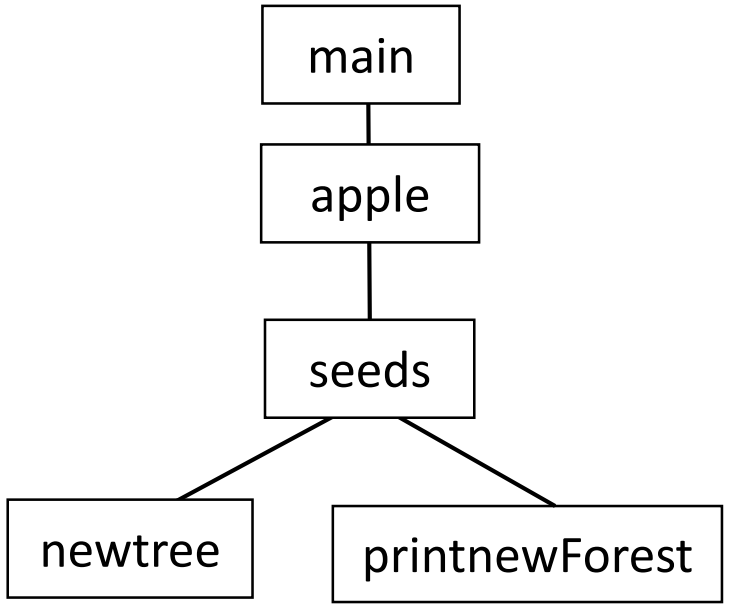
Draw a structure chart for this class.

```
public class apple
{
    public static void main (String args[])
    {  new apple ();
    }

    public apple ()
    {   System.out.println ("Welcome to apples!");
        seeds ();
    }

    public void seeds ()
    {   int amount = newtree ();
        printnewforest (amount);
    }

    public int newtree ()
       {    return (int) Math.random () * 9;
       }

       public void printnewforest (int amount)
       {  for (int i = 0 ; i < amount ; i++)
                System.out.print ("*");
       }
}
```



Draw a structure chart for this class.

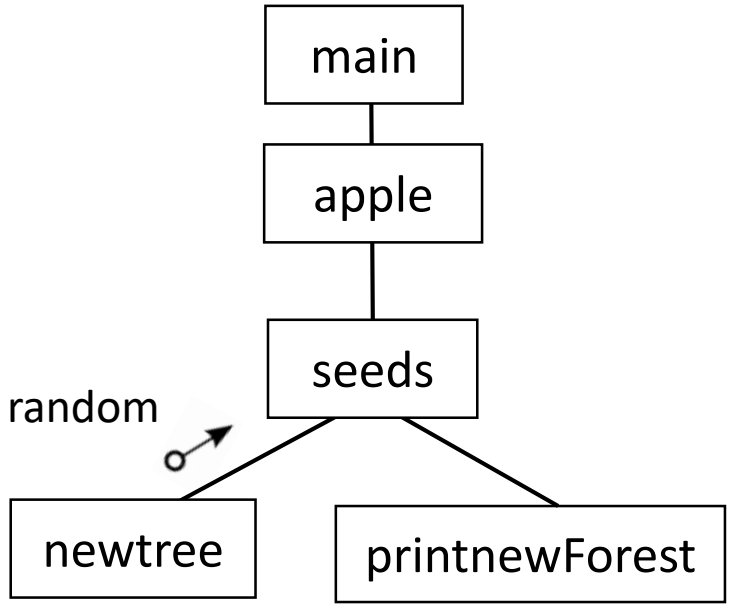
```
public class apple
{
    public static void main (String args[])
    {  new apple ();
    }

    public apple ()
    {   System.out.println ("Welcome to apples!");
        seeds ();
    }

    public void seeds ()
    {   int amount = newtree ();
        printnewforest (amount);
    }
```



Draw a structure chart for this class.

```
    public int newtree ()
    {    return (int) Math.random () * 9;
    }

    public void printnewforest (int amount)
    {  for (int i = 0 ; i < amount ; i++)
            System.out.print ("*");
    }
}
```

```
public class apple
{
    public static void main (String args[])
    {  new apple ();
    }

    public apple ()
    {   System.out.println ("Welcome to apples!");
        seeds ();
    }

    public void seeds ()
    {   int amount = newtree ();
        printnewforest (amount);
    }
```



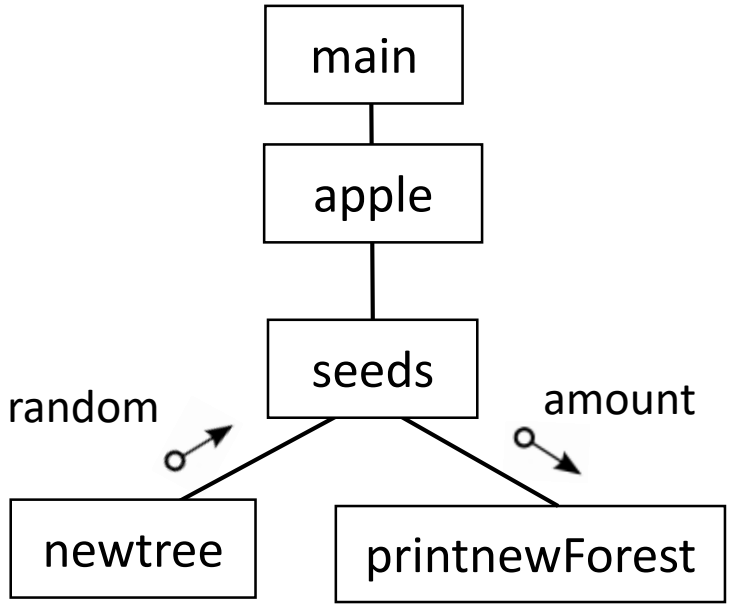
Draw a structure chart for this class.

int out

Nothing in

```
    public int newtree ()
        {    return (int) Math.random () * 9;
        }

        public void printnewforest (int amount)
        {  for (int i = 0 ; i < amount ; i++)
                System.out.print ("*");
        }
}
```

```
public class apple
{
    public static void main (String args[])
    {  new apple ();
    }

    public apple ()
    {   System.out.println ("Welcome to apples!");
        seeds ();
    }

    public void seeds ()
    {   int amount = newtree ();
        printnewforest (amount);
    }
```



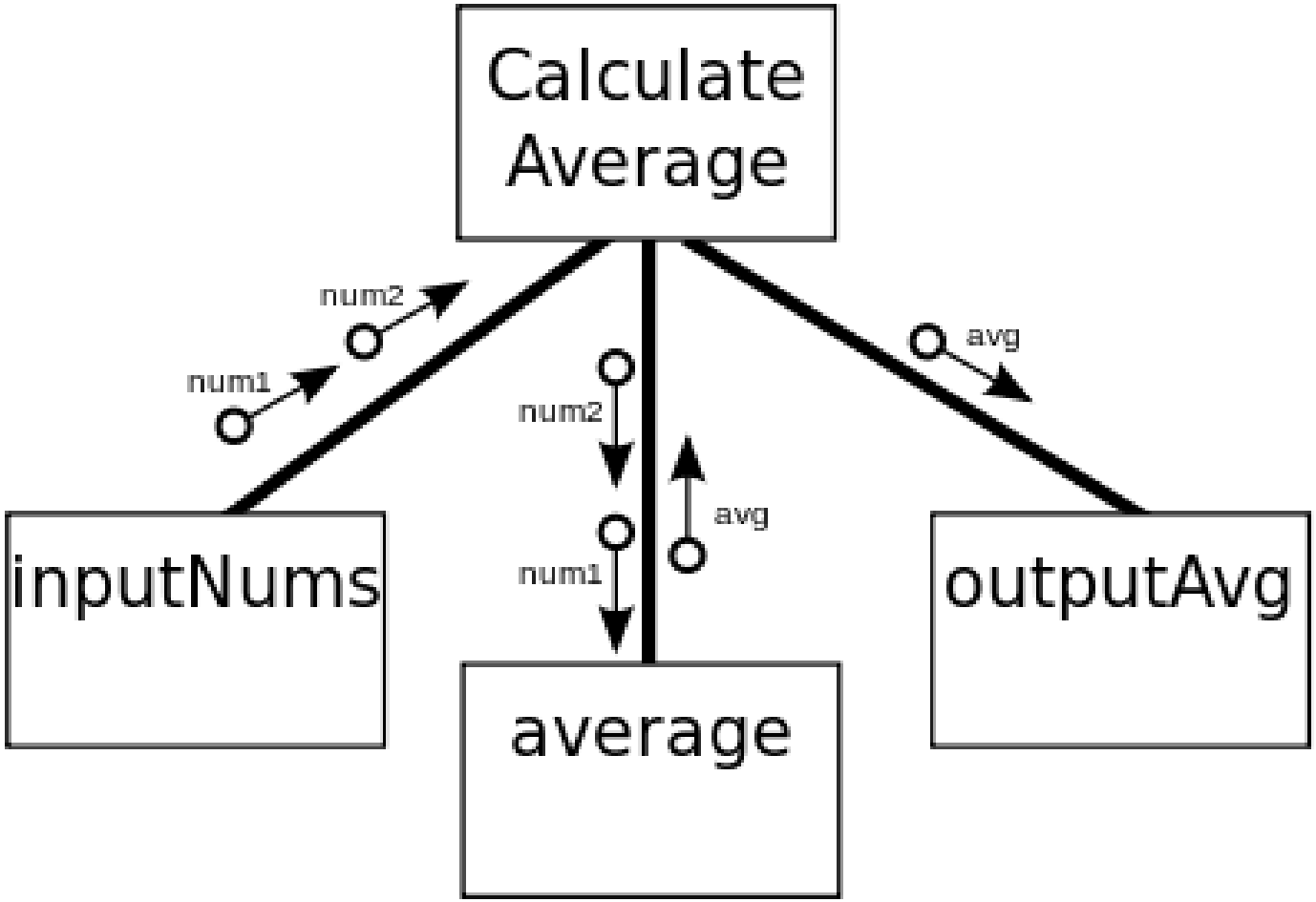
Draw a structure chart for this class.

```
    public int newtree ()
    {    return (int) Math.random () * 9;
    }

    public void printnewforest (int amount)
    {  for (int i = 0 ; i < amount ; i++)
            System.out.print ("*");
    }
}
```

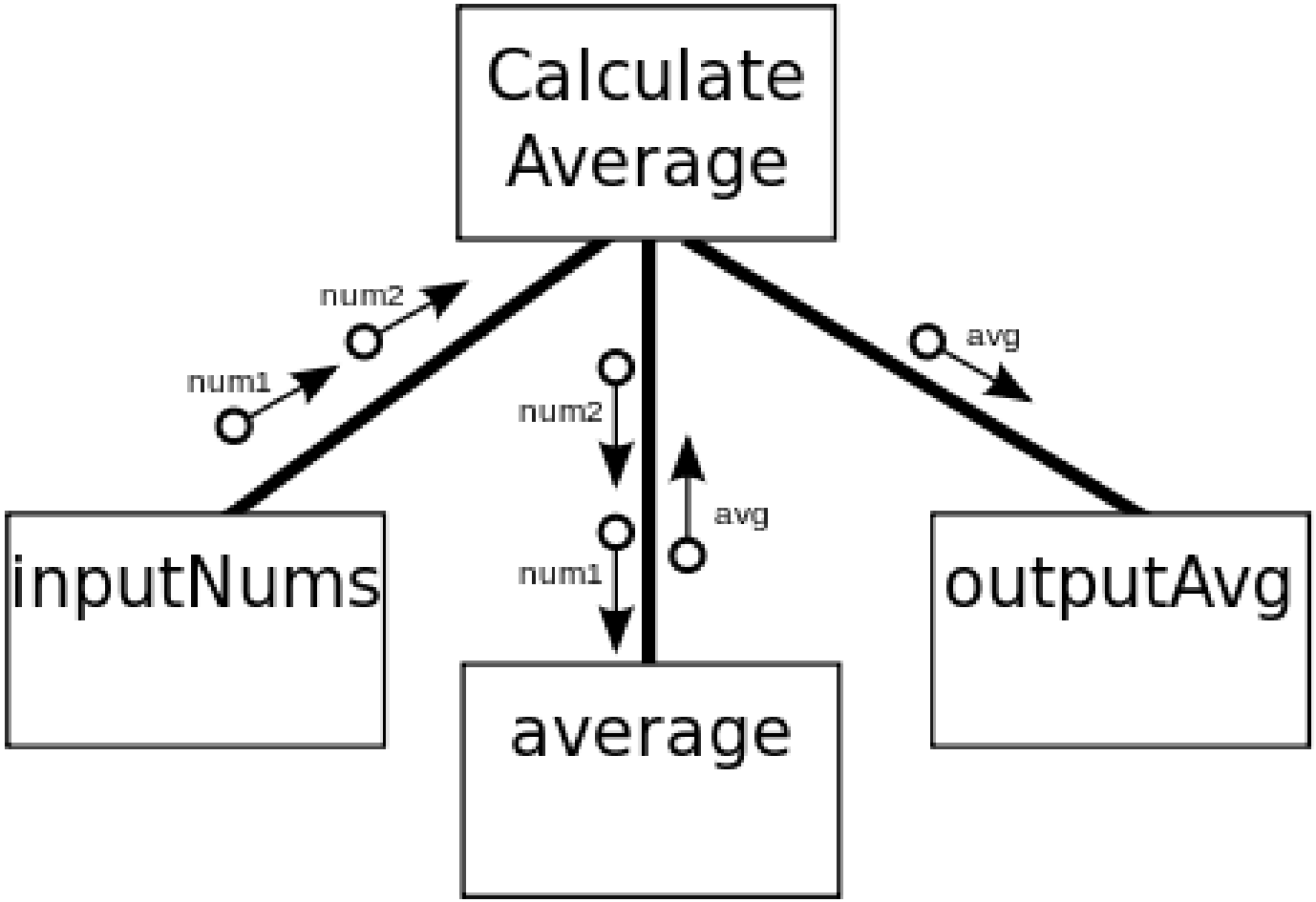
Nothing out

Amount in

More Examples…

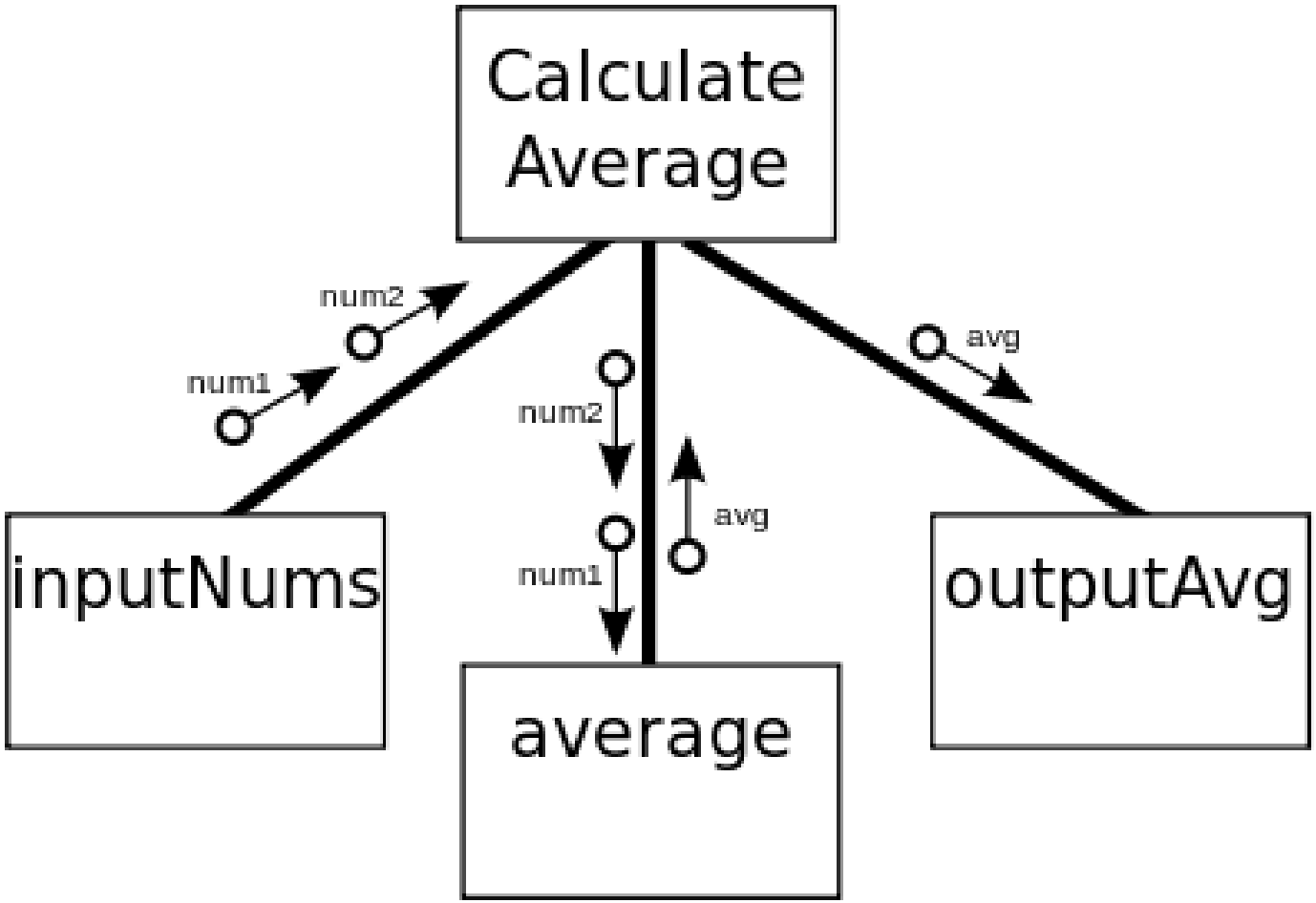


How many methods?



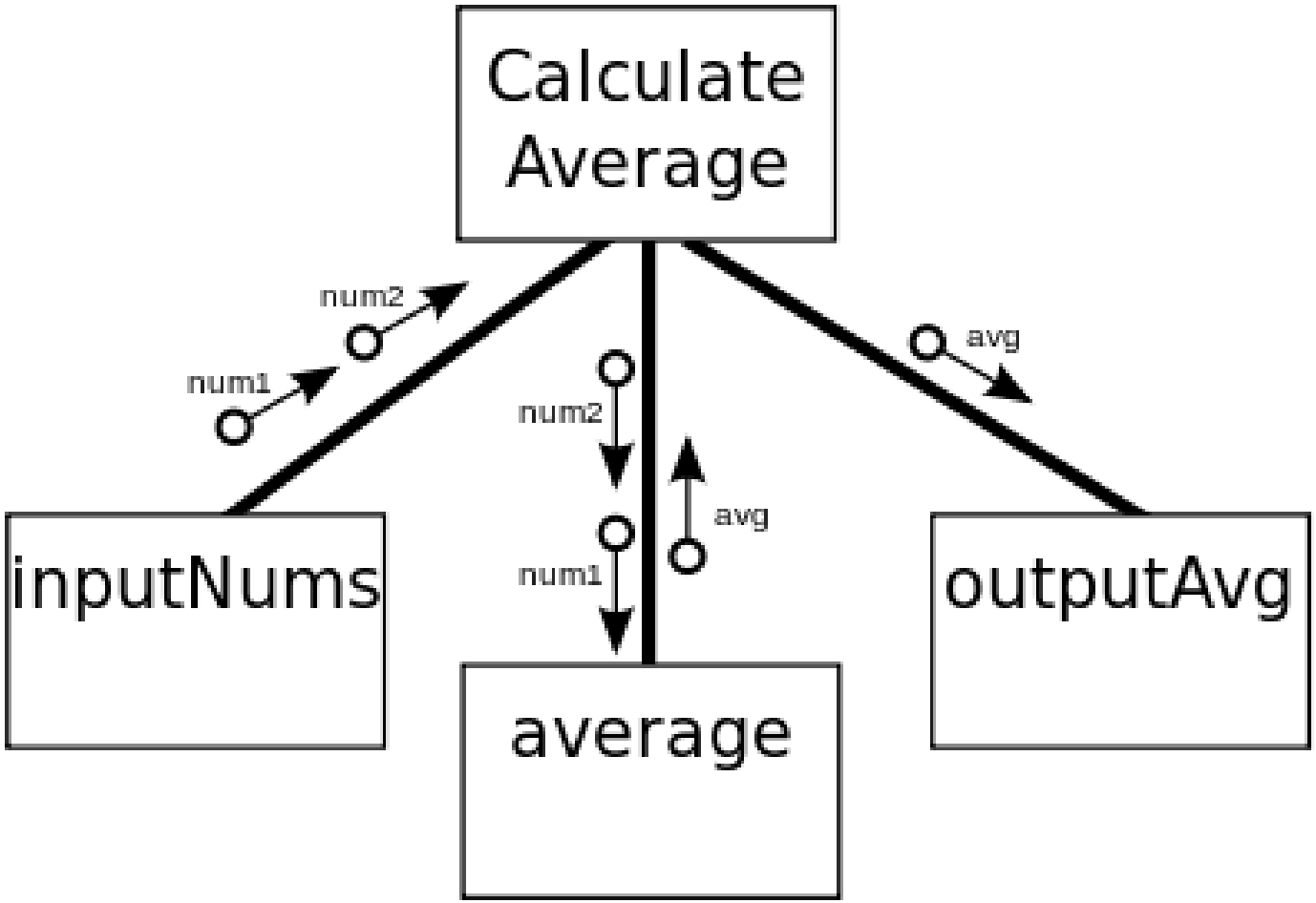
4

What is returned from average?



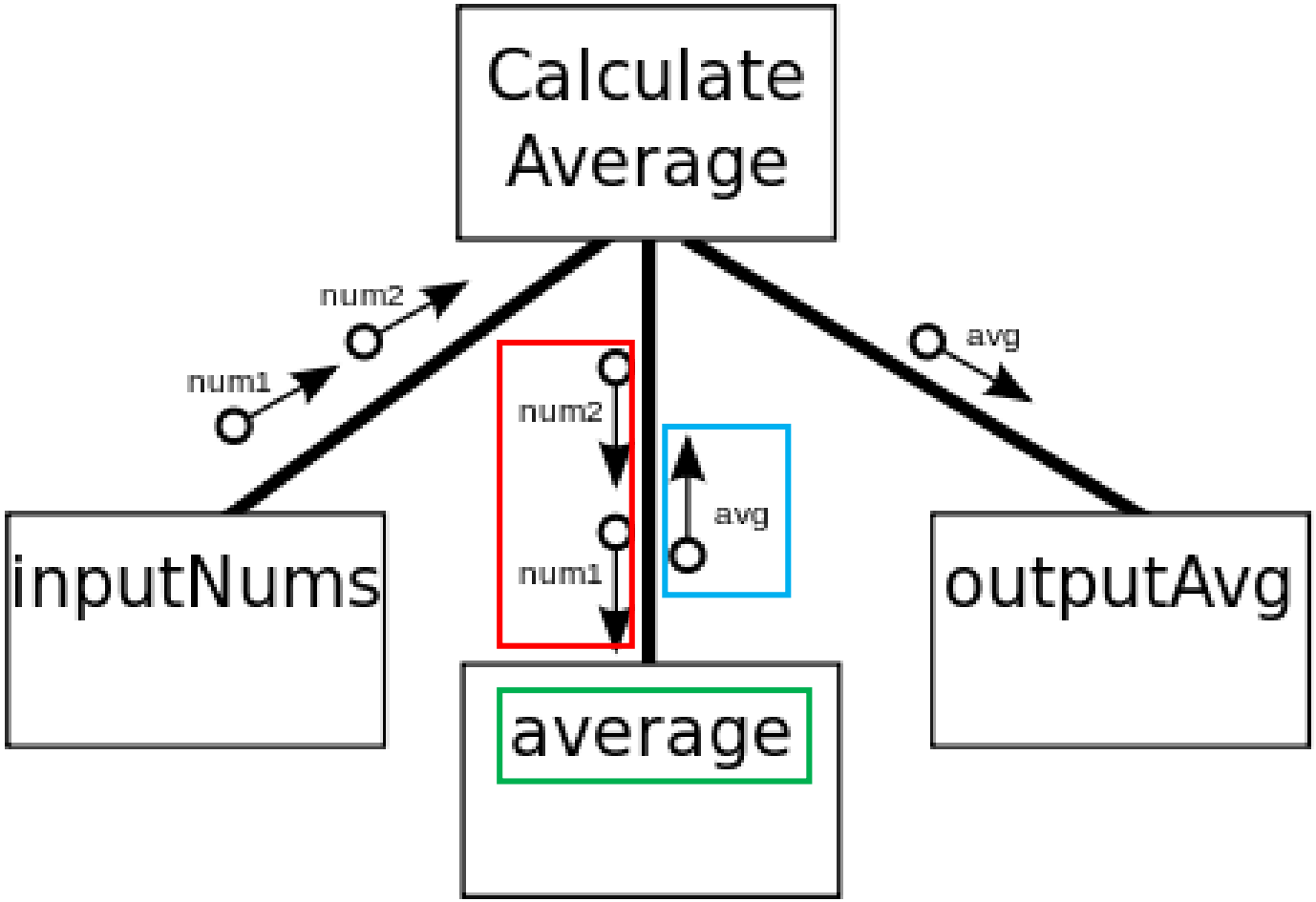
avg

What is sent in to avergae?

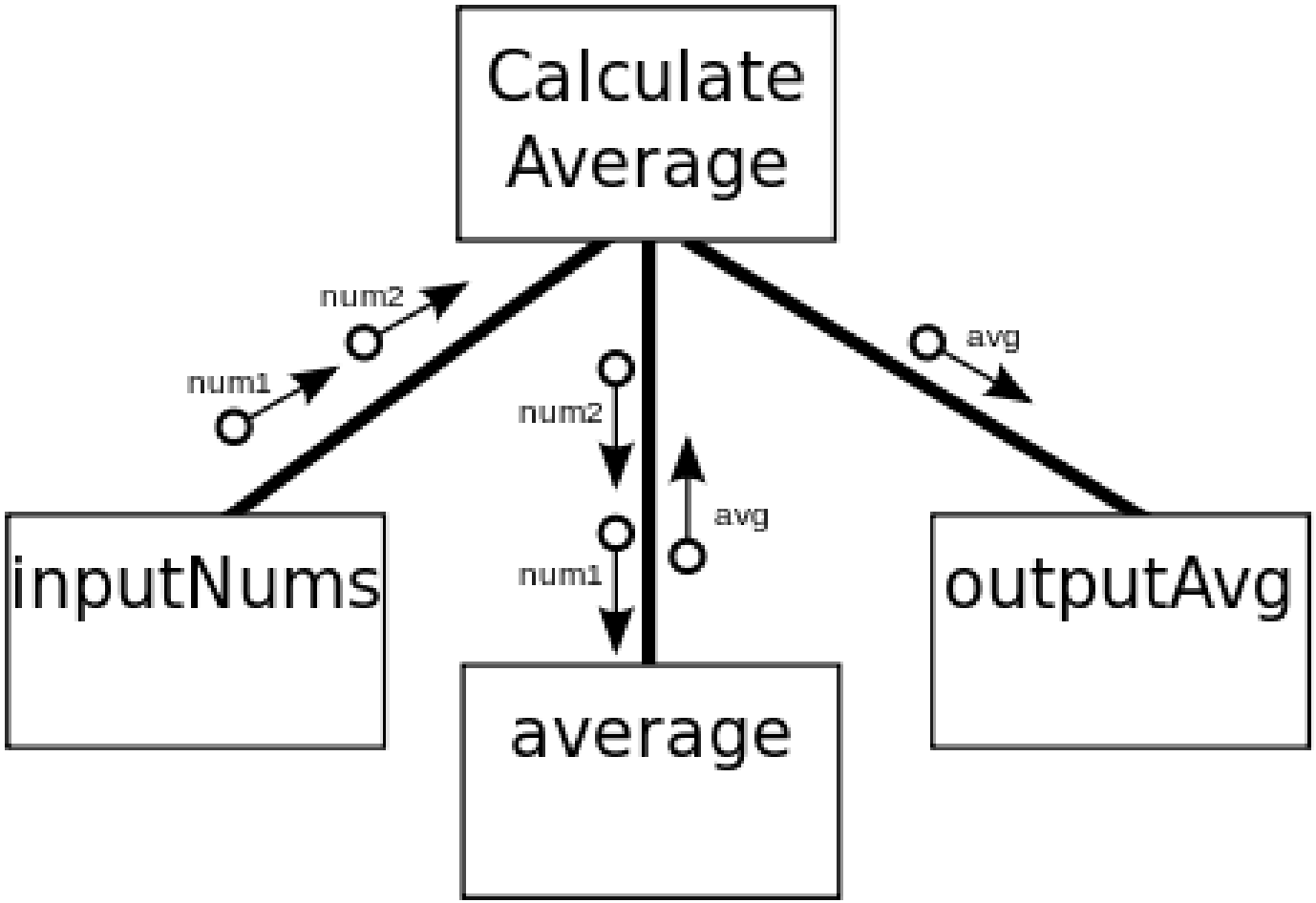


num1, num2

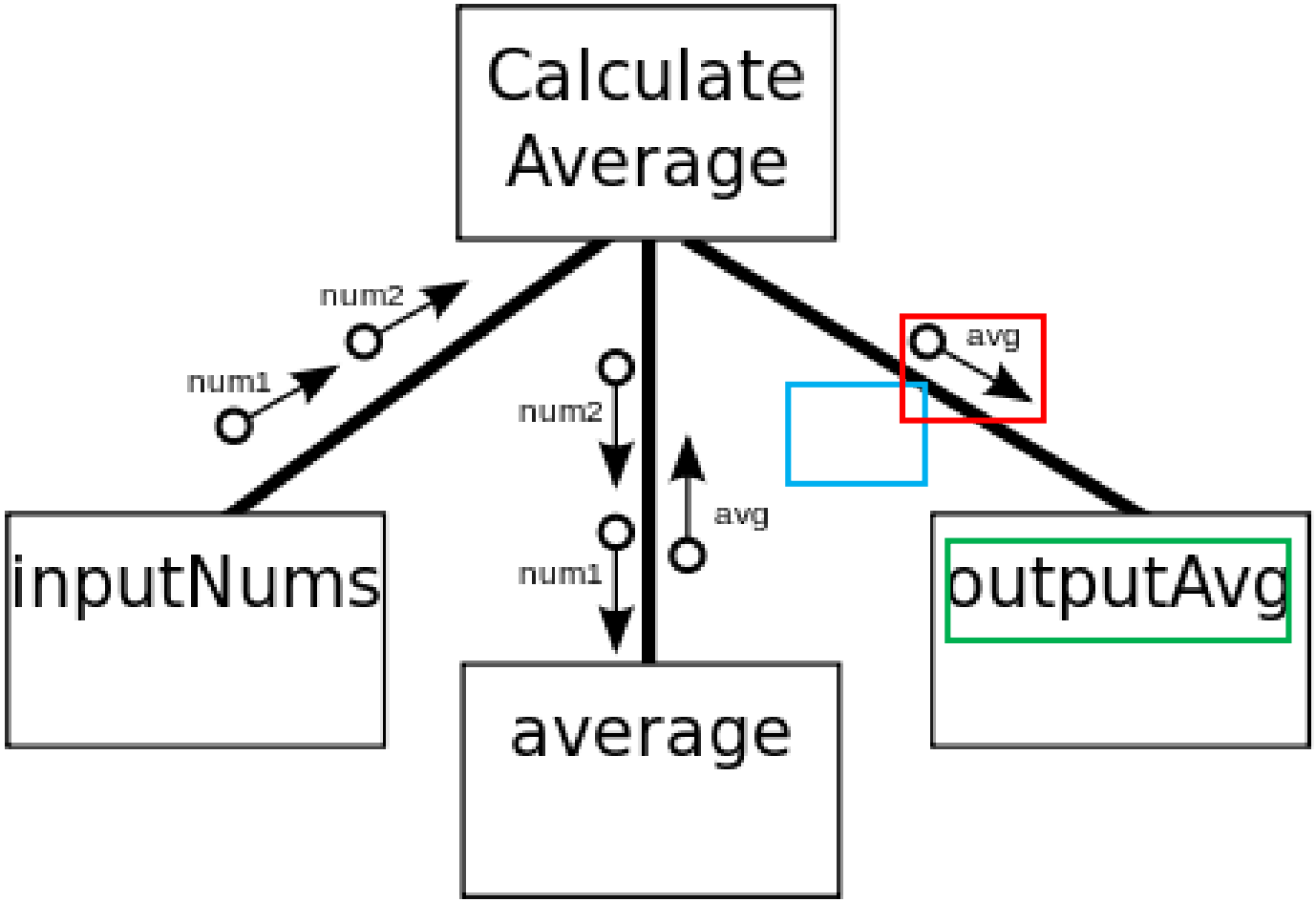
Write inputNum's method signature.




```
public int average(int num1, int num2)
```



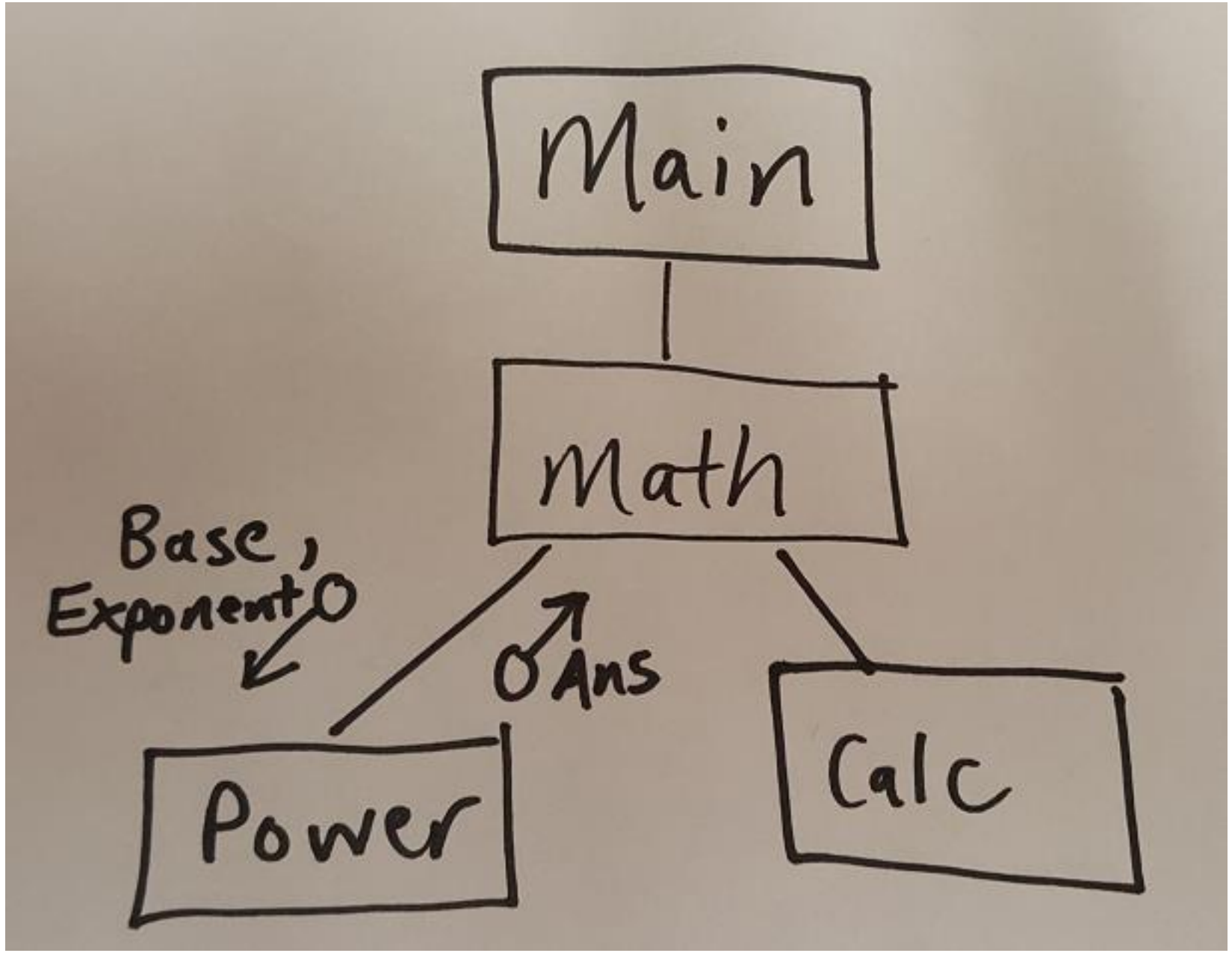
Write outputAvg's method signature.

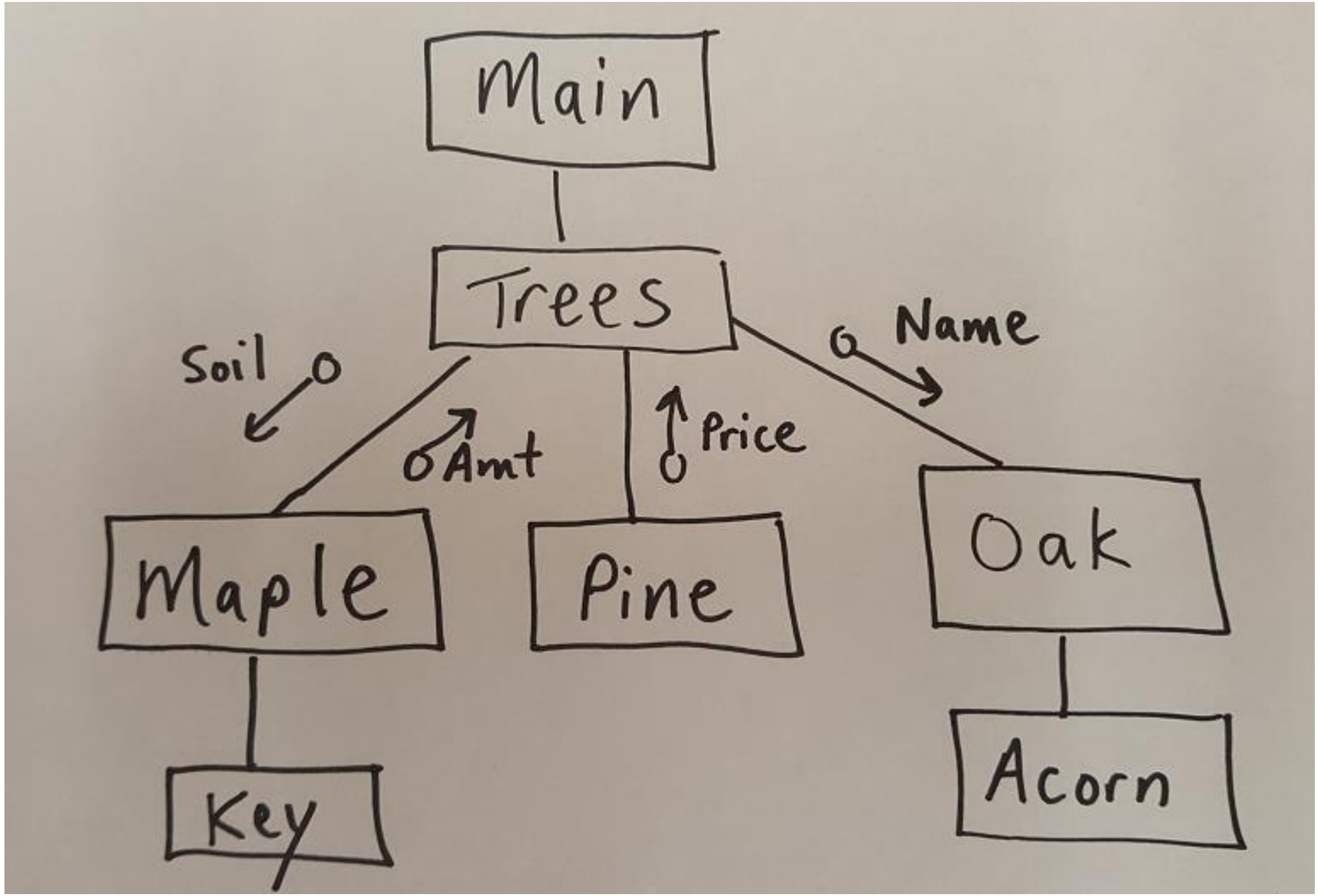


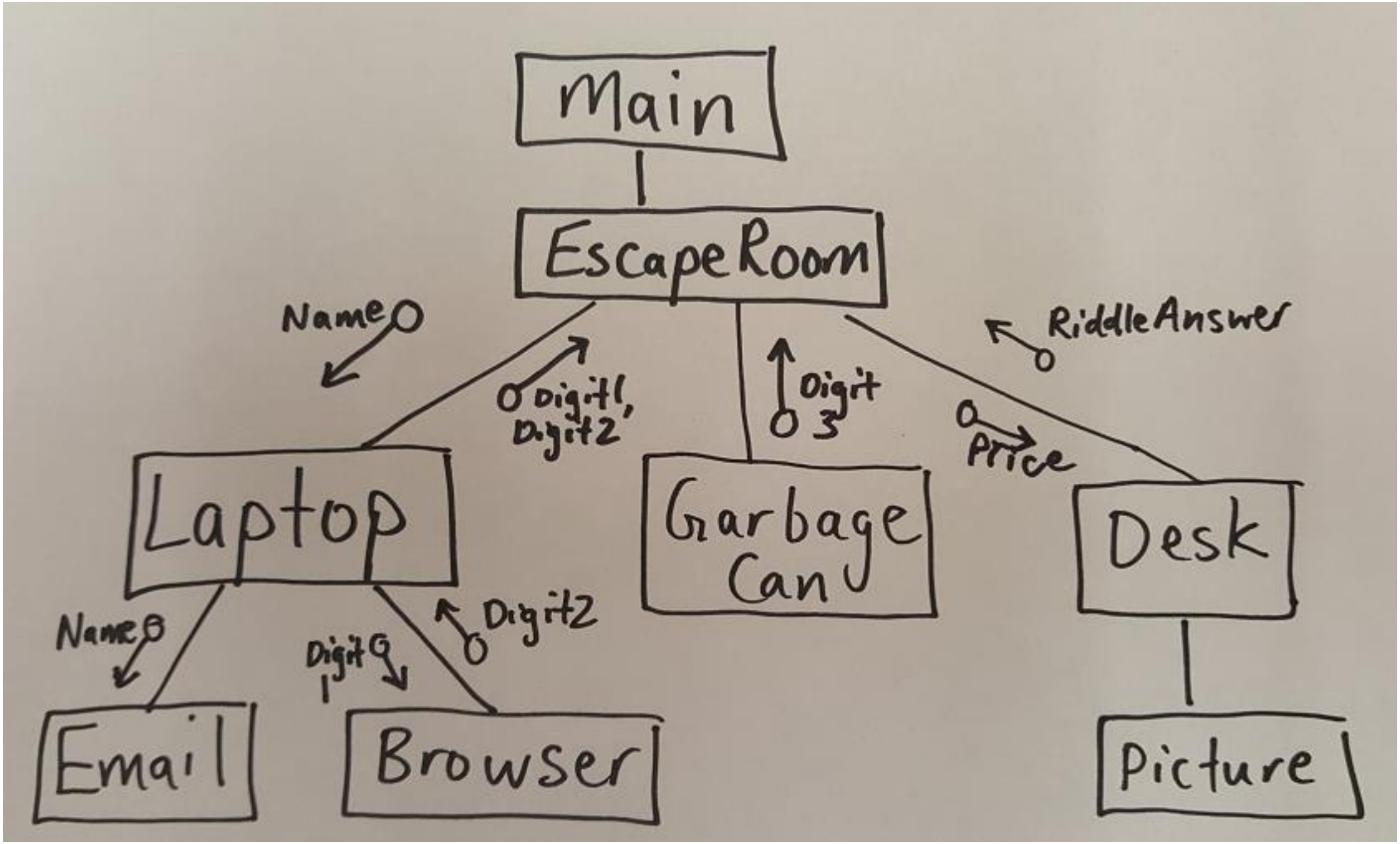

```
public void outputAvg(int num1, int num2)
```

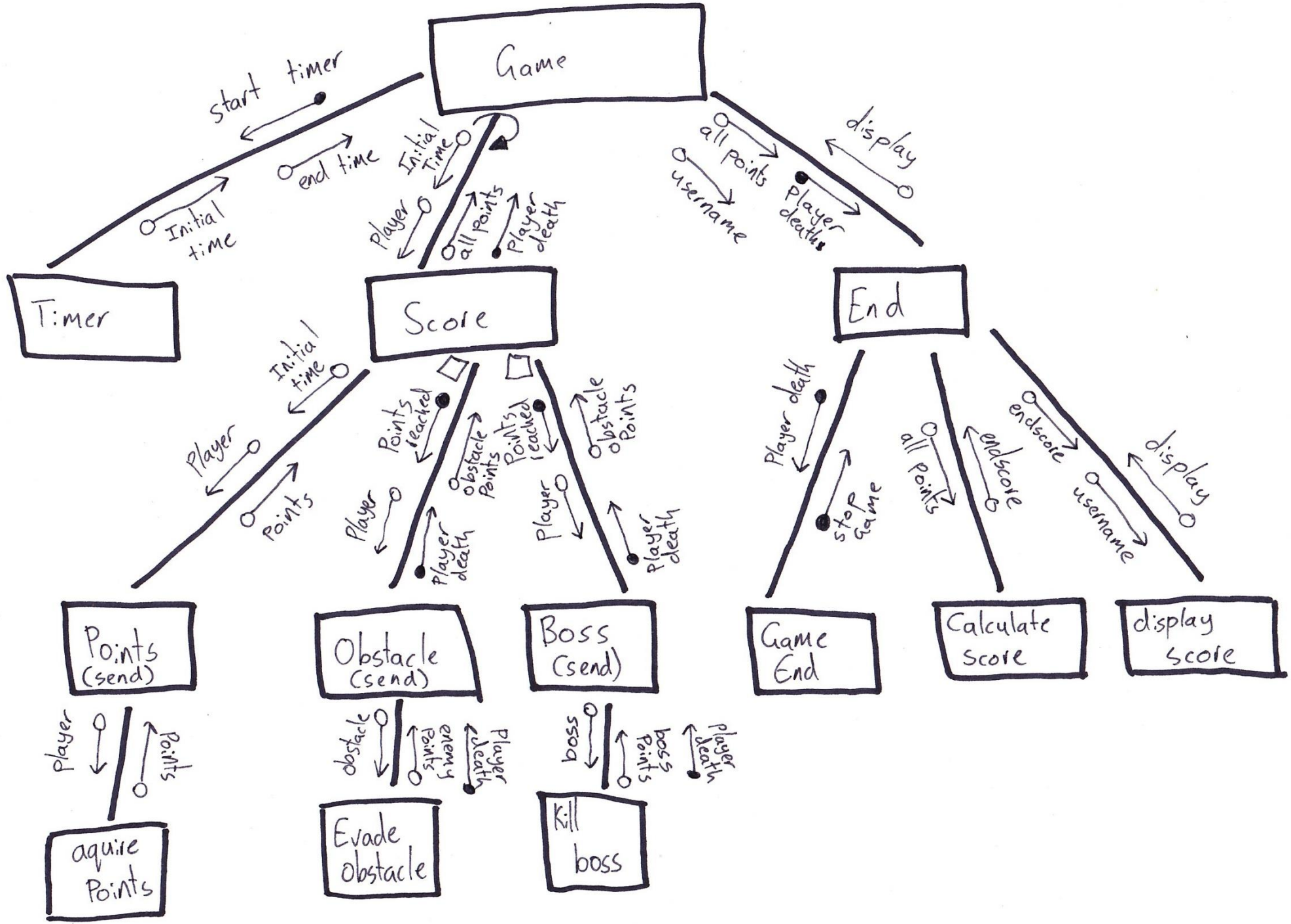
```
public class hello
{
  public static void main (String args[])
  {   new hello ();
  }

  public hello ()
  {   greetings ();
      char letter = IO.inputChar ("Letter? ");
      alphabet (letter);
  }

  public void greetings ()
  {   System.out.println ("Hello");
  }
```

```
public void alphabet (char letter)
   {   if (isAlpha (letter))
           System.out.println ("Ascii");
   }

   public boolean isAlpha (char letter)
   {   int num = (int) letter;
       if (num >= 65 && num <= 122)
           return true;
       else
           return false;
   }}
```


Main
Math
Base,
Exponent
Ans
Power
Calc


Main
Trees
Soil
Amt
Price
Name
Maple
Pine
Oak
Key
Acorn


Main
EscapeRoom
Name0
Digit1,
Digit2
Digit
3
RiddleAnswer
Price
Laptop
Garbage
Can
Desk
Name0
Digit
1
Digit2
Email
Browser
Picture


Game
start timer
end time
Initial time
Initial Time
Player
all Points
Player death
all Points
username
Player death
display
Timer
Score
End
Initial time
Player
Points
Points reached
obstacle Points
Points reached
obstacle Points
Player
Player death
Player
Player death
Player death
stop game
all Points
endscore
endscore
username
display
Points (send)
Obstacle (send)
Boss (send)
Game End
Calculate score
display score
player
Points
obstacle
enemy Points
Player death
boss
boss Points
Player death
aquire Points
Evade obstacle
Kill boss