

Running an app results in invoking its `main` method.

When a method is invoked, a block of memory is allocated to store the values of the parameters and variables of the method.

Main method

```
public static void main(String[] args)
```

Question

How many parameters does the `main` method have?

¹We will come back to this type later in the course.


Main method

```
public static void main(String[] args)
```

Question

How many parameters does the `main` method have?

Answer: one.

¹We will come back to this type later in the course. 

Main method

```
public static void main(String[] args)
```

Question

How many parameters does the `main` method have?

Answer: one.

Question

What is the name of the parameter?

¹We will come back to this type later in the course.

Main method

```
public static void main(String[] args)
```

Question

How many parameters does the main method have?

Answer: one.

Question

What is the name of the parameter?

Answer: args.

¹We will come back to this type later in the course.

Main method

```
public static void main(String[] args)
```

Question

How many parameters does the main method have?

Answer: one.

Question

What is the name of the parameter?

Answer: args.

Question

What is the type of the parameter?

¹We will come back to this type later in the course.

Main method

```
public static void main(String[] args)
```

Question

How many parameters does the main method have?

Answer: one.

Question


What is the name of the parameter?

Answer: args.

Question

What is the type of the parameter?

Answer: String[].¹

¹We will come back to this type later in the course. 

Main method

In the first half of this course, we will not use the parameter of the main method. Therefore, we will **not** include the parameter of the main method in our memory diagrams (for now).

Price of gold

Simplified version of body of the main method:

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD)
```

Question

What are the names of the variables in the above main method?

Price of gold

Simplified version of body of the main method:

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD)
```

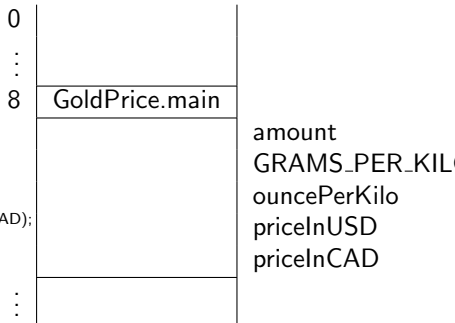
Question

What are the names of the variables in the above main method?

Answer: amount, GRAMS_PER_KILO, ouncePerKilo, priceInUSD and priceInCAD.

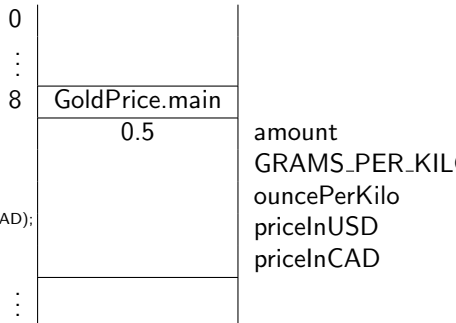
Memory model

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```



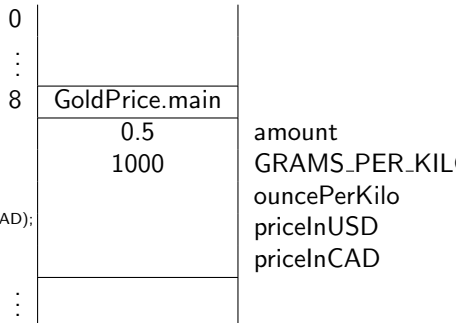
Memory model

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```



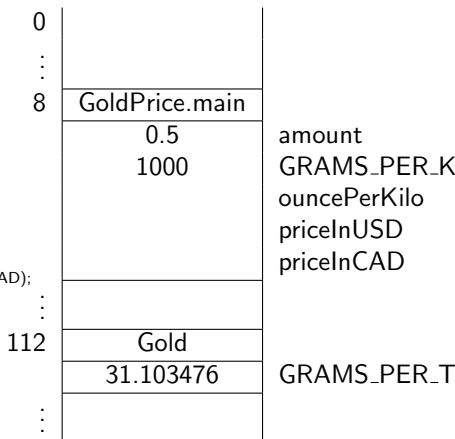
Memory model

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```



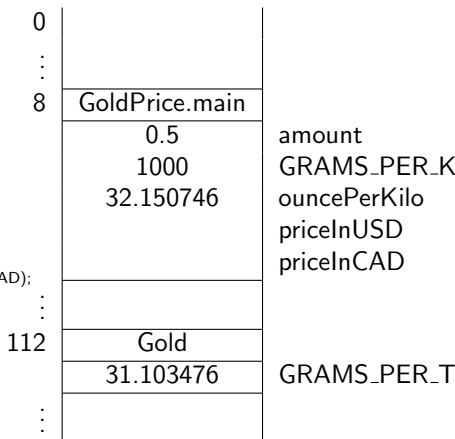
Memory model

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```



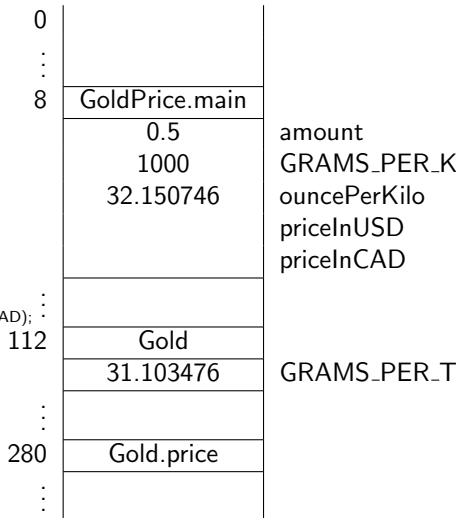
Memory model

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```



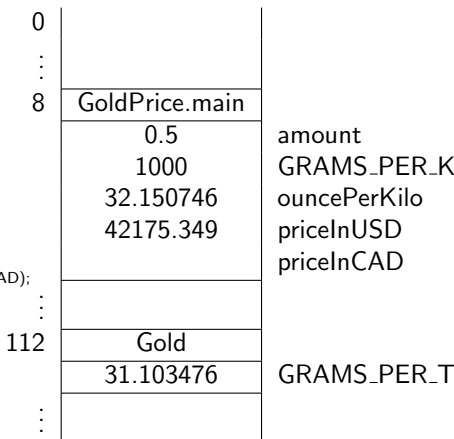
Memory model

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```



Memory model

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```



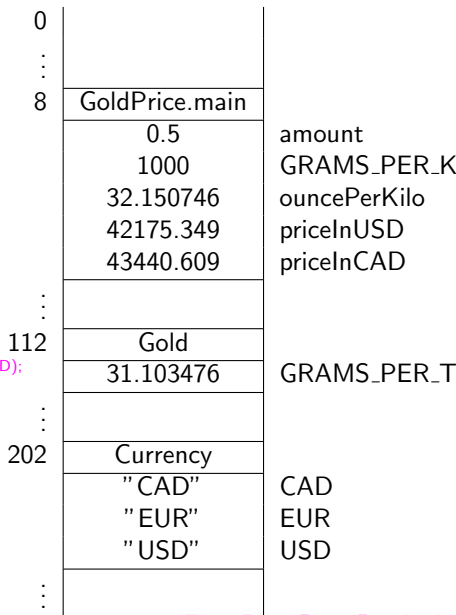
Memory model

```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```

8	GoldPrice.main	
	0.5	amount
	1000	GRAMS_PER_KILO
	32.150746	ouncePerKilo
	42175.349	priceInUSD
		priceInCAD
112	Gold	
	31.103476	GRAMS_PER_TROY_OUNCE
202	Currency	
	"CAD"	CAD
	"EUR"	EUR
	"USD"	USD
240	Currency.convert	
	42175.349	amount
	"USD"	from
	"CAD"	to

Memory model

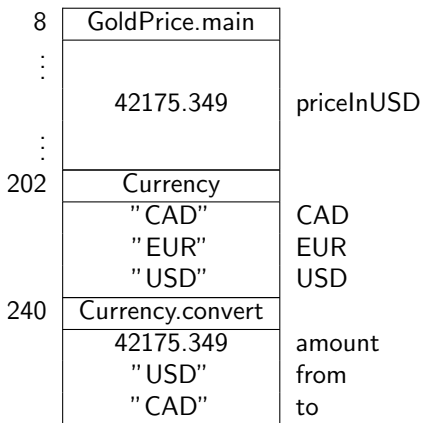
```
double amount = 0.5;
final double GRAMS_PER_KILO = 1000;
double ouncePerKilo =
    GRAMS_PER_KILO / Gold.GRAMS_PER_TROY_OUNCE;
double priceInUSD = amount * ouncePerKilo * Gold.price();
double priceInCAD =
    Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```



Pass-by-value

```
... Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```

The **values** of the arguments are passed in Java (and many other programming languages).



Pass-by-reference

```
... Currency.convert(priceInUSD, Currency.USD, Currency.CAD);
```

The **addresses** of the arguments are passed in some programming languages such as Perl.

	GoldPrice.main	
⋮		
28	42175.349	priceInUSD
⋮		
204	Currency	
	"CAD"	CAD
	"EUR"	EUR
220	"USD"	USD
	Currency.convert	
	28	amount
	220	from
	204	to

Question

```
int value = 2;  
Magic.triple(value);  
output.println(value);
```

What is the output produced by the above snippet?

Question

```
int value = 2;  
Magic.triple(value);  
output.println(value);
```

What is the output produced by the above snippet?

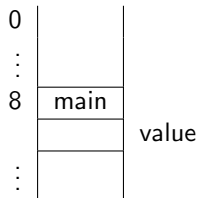
Answer

2

The method `triple` of the class `Magic` gets passed only the value of the variable `value`, not its address.

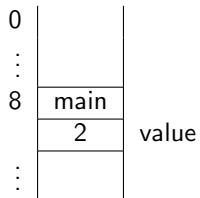
Pass-by-value

```
int value = 2;  
Magic.triple(value);  
output.println(value);
```



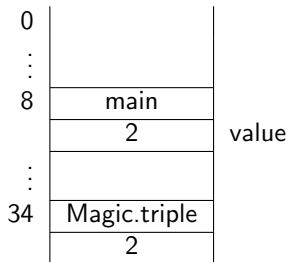
Pass-by-value

```
int value = 2;  
Magic.triple(value);  
output.println(value);
```



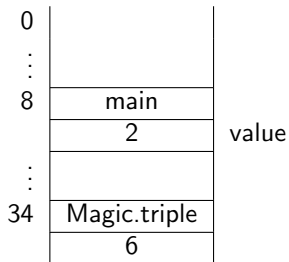
Pass-by-value

```
int value = 2;  
Magic.triple(value);  
output.println (value);
```



Pass-by-value

```
int value = 2;  
Magic.triple(value);  
output.println (value);
```



Question

```
int value = 2;  
Magic.triple(value);  
output.println(value);
```

If Java were to use pass-by-reference, what would the output produced by the above snippet be?

Question

```
int value = 2;  
Magic.triple(value);  
output.println(value);
```

If Java were to use pass-by-reference, what would the output produced by the above snippet be?

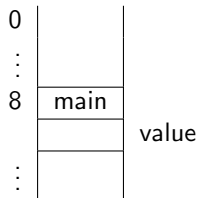
Answer

6 or any other integer.

The method `triple` of the class `Magic` gets passed the address of the variable `value` and, hence, can change its value.

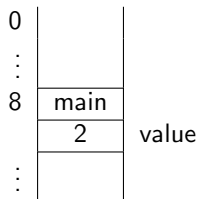
Pass-by-reference

```
int value = 2;  
Magic.triple(value);  
output.println(value);
```



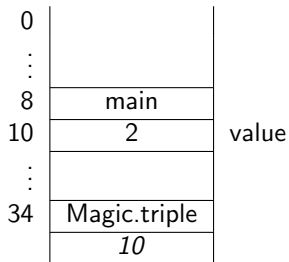
Pass-by-reference

```
int value = 2;  
Magic.triple(value);  
output.println(value);
```



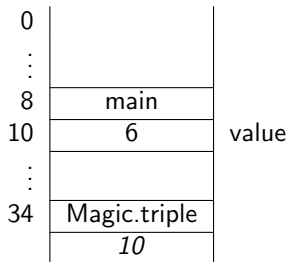
Pass-by-reference

```
int value = 2;  
Magic.triple(value);  
output.println (value);
```



Pass-by-reference

```
int value = 2;  
Magic.triple(value);  
output.println (value);
```



The signature of a method is unique in its class.

Terminology

Two methods in the same class with the same name are said to be **overloaded**.

Example

In the class `PrintStream`, the method `println` is overloaded.

When the compiler encounters the invocation

$C.m(a_1, \dots, a_n)$

it must determine which method to invoke. This process is known as **early binding**. It consists of the following three steps.

- 1 Find the class C .
- 2 Find the compatible methods m in class C .
- 3 Select the most specific compatible method m in class C .

Early binding (step 1)

Early binding of $C.m(a_1, \dots, a_n)$.

Question

How can “find the class C” fail?

Early binding (step 1)

Early binding of $C.m(a_1, \dots, a_n)$.

Question

How can “find the class C” fail?

Answer

The class is missing, since it has not been imported, it is not part of the classpath, or its name has been misspelled.

Early binding (step 2)

Early binding of $C.m(a_1, \dots, a_n)$.

Question

When is a method m in class C **compatible** with invocation $C.m(a_1, \dots, a_n)$?

Answer

The types of the arguments a_1, \dots, a_n are **compatible** with the types of the parameters of the method m .

Question

Which methods in class `PrintStream` are compatible with invocation `output.println(1)`?

Early binding (step 2)

Question

Which methods in class `PrintStream` are compatible with invocation `output.println(1)`?

Answer

```
println (double)
println (float )
println (int )
println (long)
```


Early binding (step 2)

Early binding of $C.m(a_1, \dots, a_n)$.

Question

How can “find compatible methods m in class C ” fail?

Early binding (step 2)

Early binding of $C.m(a_1, \dots, a_n)$.

Question

How can “find compatible methods m in class C ” fail?

Answer

The method is missing, since it simply does not exist or its name has been misspelled.

Question

Which of the methods

`println(double)`

`println(float)`

`println(int)`

`println(long)`

in class `PrintStream` is most specific to invocation
`output.println(1)`?

Early binding (step 3)

Question

Which of the methods

`println(double)`

`println(float)`

`println(int)`

`println(long)`

in class `PrintStream` is most specific to invocation
`output.println(1)`?

Answer

`println(int)` since the argument 1 is of type `int`.

Question

Which of the methods

`println(double)`

`println(float)`

`println(int)`

`println(long)`

in class `PrintStream` is most specific to invocation
`output.println(1L)`?

Early binding (step 3)

Question

Which of the methods

`println(double)`

`println(float)`

`println(int)`

`println(long)`

in class `PrintStream` is most specific to invocation
`output.println(1L)`?

Answer

`println(long)` since the argument `1L` is of type `long`.

Question

Which of the methods

`println(double)`

`println(float)`

`println(int)`

`println(long)`

in class `PrintStream` is most specific to invocation
`output.println('1')`?

Early binding (step 3)

Question

Which of the methods

`println(double)`

`println(float)`

`println(int)`

`println(long)`

in class `PrintStream` is most specific to invocation `output.println('1')`?

Answer

`println(int)` since the argument `'1'` is of type `char` and converting it to an `int` requires the least amount of promotion.

Early binding (step 3)

Early binding of $C.m(a_1, \dots, a_n)$.

Question

How can “select the most specific compatible method m in class C ” fail?

Early binding (step 3)

Early binding of $C.m(a_1, \dots, a_n)$.

Question

How can “select the most specific compatible method m in class C ” fail?

Answer

Consider the class C with methods

$m(\text{int}, \text{double})$

$m(\text{double}, \text{int})$

and the invocation $C.m(1, 2)$. Note that both $m(\text{int}, \text{double})$ and $m(\text{double}, \text{int})$ are compatible with $C.m(1, 2)$. However, both require the same amount of promotion, namely promoting an `int` to a `double`. Hence, one is not more specific than the other and therefore we cannot select the most specific one.

The Four Steps

- 1 Solve the problem.
- 2 Write the app.
- 3 Compile the app.
- 4 Run the app.

- 1 Analysis (define the problem)
- 2 Design (solve the problem)
- 3 Implementation (write and compile the app)
- 4 Testing (run the app)
- 5 Deployment

We will come back to this in Chapter 7.

Is your mouse faster than Usain Bolt?



comicvine.com and greatrun.org

Is your mouse faster than Usain Bolt?



meraneed.com and greatrun.org

Is your mouse faster than Usain Bolt?

As we have seen before, the average speed of Usain Bolt when he ran his 100 meter world record was 23.35 miles per hour.

Problem

Determine the average speed of your mouse cursor in miles per hour.

Part of the analysis phase.

- Is any input needed? If so, how is it provided? Is any validation of the input needed?
- Is there any output? If so, how should the output be provided?

Problem

Print on the console

Move your mouse immediately after entering the width of the screen in centimeters:

Compute the average speed of the mouse during 0.1 seconds in miles per hour. Print on the console the average speed with two digits precision.

To solve the problem, we can use components that

- return x-coordinate of the mouse cursor
- return y-coordinate of the mouse cursor
- return the maximal x-coordinate (minimum is zero)
- return the maximal y-coordinate (minimum is zero)
- pause the execution by n milliseconds

Question

How do we solve the problem?

Each component consists of

- a jar (Java archive) file and
- an API.

To use the component,

- download the jar file and add it to the classpath and
- study the API.

Add a jar file to your classpath

- Download the jar file and save it in the directory ???/???/???.
- In the directory with your code, create a file named, say `begin.bat`, with content

```
set CLASSPATH=.;???/???/???/franck.jar;%CLASSPATH%
```
- Open the command prompt and go the directory containing your code. Before running `javac` and `java`, run `begin`.

Add a jar file to your classpath

Another way:

- Download the jar file and save it in the folder
C:\Program Files\Java\jdk1.8.0_??\jre\lib\ext

Study the APIs of

- [franck.cse5910.Mouse](#)
- [franck.cse5910.Timing](#)