

# **Complete Java Masterclass @Udemy.com**

**CareerDevs Classroom Presentation  
December 18, 2017  
@GeekyCoderr**

# **Brief Overview Section 5 & 6**

---

**The view from 10,000 feet up.**

**We started this tutorial a week ago.**

**How far did you get?**

**That's a rhetorical question. You don't have to tell me.**

Section: 1 1 / 1  
Introduction

Section: 2 8 / 10  
Setup

Section: 3 3 / 3  
First Steps

Section: 4 2 / 8  
Variables, Datatypes and Operators

Section: 5 0 / 7  
Java Tutorial: Expressions, Statements, Code blocks, Methods and more

Section: 6 0 / 4  
Control Flow Statements

Section: 7 3 / 9  
OOP Part 1 - Classes, Constructors and Inheritance



Resources available

# COMPLETE JAVA MASTERCLASS

Any Questions?  
Ask them in the course forum.

with Tim Buchalka  
@timbuchalka



**Before you can get to the fun stuff like  
OOP...**





**Just like a  
pastry chef has  
to learn how to  
measure and  
calculate  
before they  
make great  
yummy cakes...**



# We have to learn the boring stuff like Keywords, Expressions, Statements, Code blocks, Methods and more...

```
public class Methods {  
  
    public static void main(String[] args) {  
  
        int num = 5;  
  
        double pi = 3.14;  
  
        System.out.println(num);  
  
        System.out.println(pi);  
  
        otherMethod();  
    }  
  
    public static void otherMethod() {  
  
        double num2 = 6.28;  
  
        System.out.println(num2);  
  
    }  
}
```





WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

[Interaction](#)

[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

[Tools](#)

[Article](#)

[Talk](#)

[Read](#)

[Edit](#)

[View history](#)



[Not logged in](#) [Talk](#) [Contributions](#) [Create account](#) [Log in](#)

# List of Java keywords

From Wikipedia, the free encyclopedia

In the [Java programming language](#), a **keyword** is one of **53 reserved words**<sup>[1]</sup> that have a predefined meaning in the language; because of this, programmers cannot use keywords as names for [variables](#), [methods](#), [classes](#), or as any other [identifier](#).<sup>[2]</sup> Due to their special functions in the language, most [integrated development environments](#) for Java use [syntax highlighting](#) to display keywords in a different colour for easy identification.

## Contents [\[hide\]](#)

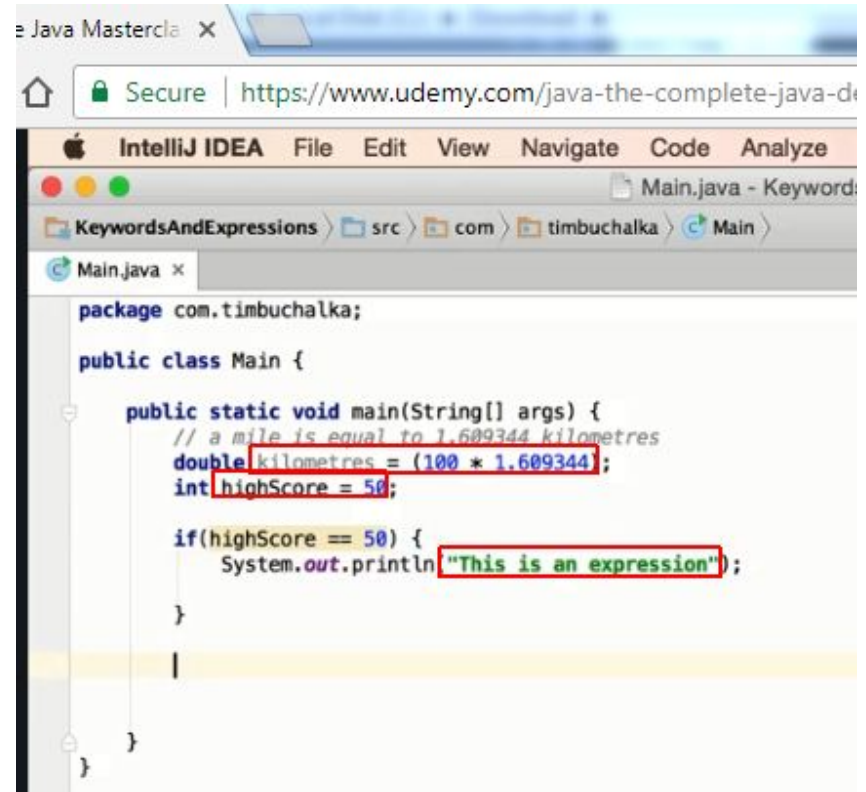
- [List](#)
- [Reserved words for literal values](#)
- [Unused](#)
- [See also](#)
- [References](#)

```
public void processData()  
{  
    do  
    {  
        int data = getData();  
        if(data < 0)  
            performOperation1(data);  
        else  
            performOperation2(data);  
    }  
    while(hasMoreData());  
}
```

A snippet of Java code with keywords highlighted in blue and bold font

# Expressions

- Between the datatype and semi-colon  
ex. double `E = mc^2`;
- Prefix and postfix operators  
ex. `a++`, `b--`, `++c`, `--d`
- Inside the parentheses of a method  
`println: System.out.println("expression")`
- Object instantiation  
ex. `Coffee myCoffee = new Coffee()`;
- Method calls with or w/out return value  
ex. `boolean isSweet = addSugar(2)`;
- Ternary operators  
ex. `myCoffee = Hazelnut ? iced : hot`;



The screenshot shows the IntelliJ IDEA IDE with a Java file named `Main.java` open. The code is as follows:

```
package com.timbuchalka;

public class Main {

    public static void main(String[] args) {
        // a mile is equal to 1.609344 kilometres
        double kilometres = (100 * 1.609344);
        int highScore = 50;

        if(highScore == 50) {
            System.out.println("This is an expression");
        }
    }
}
```

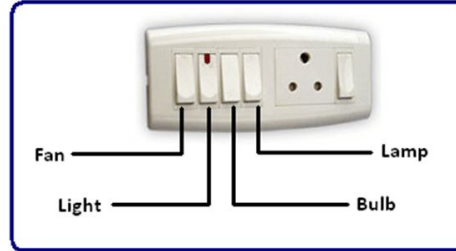
Red boxes highlight the following expressions in the code:

- `E = mc^2` in the example text.
- `"expression"` in the `println` statement.
- `myCoffee = new Coffee()` in the example text.
- `(100 * 1.609344)` in the `double kilometres` assignment.
- `50` in the `int highScore` assignment.
- `"This is an expression"` in the `println` statement.
- `isSweet = addSugar(2)` in the example text.



# Control Flow Statements

```
1 import java.util.*;
2 public class hello {
3     private static Scanner input;
4     public static void main(String[] args) {
5         Random rnd=new Random();
6         int z=rnd.nextInt(5);
7         int x=99;
8         while(x!=z) {
9             System.out.println("輸入數字:(0-4)")
10            input = new Scanner(System.in);
11            x= input.nextInt();
12            switch (x) {
13                case 1:
14                    System.out.println("ONE");
15                    break;
16                case 2:
17                    System.out.println("TWO");
18                    break;
19                case 3:
20                    System.out.println("THREE");
21                    break;
22                default:
23                    System.out.println("其他");
24            }
25        }
26        System.out.println("z="+z);
27    }
28 }
```



Switch - syntax

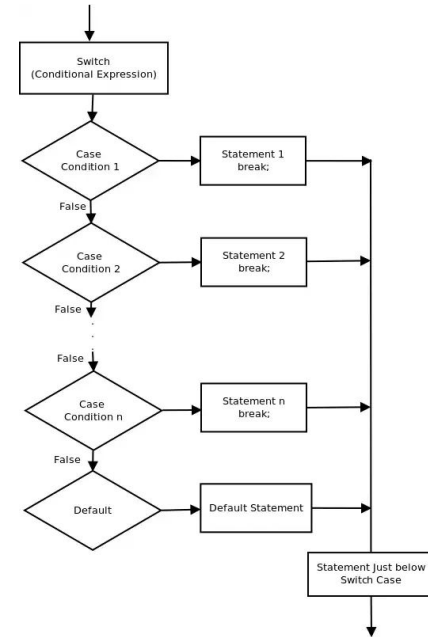
- The general syntax of a switch statement is:

switch  
and  
case  
are  
reserved  
words

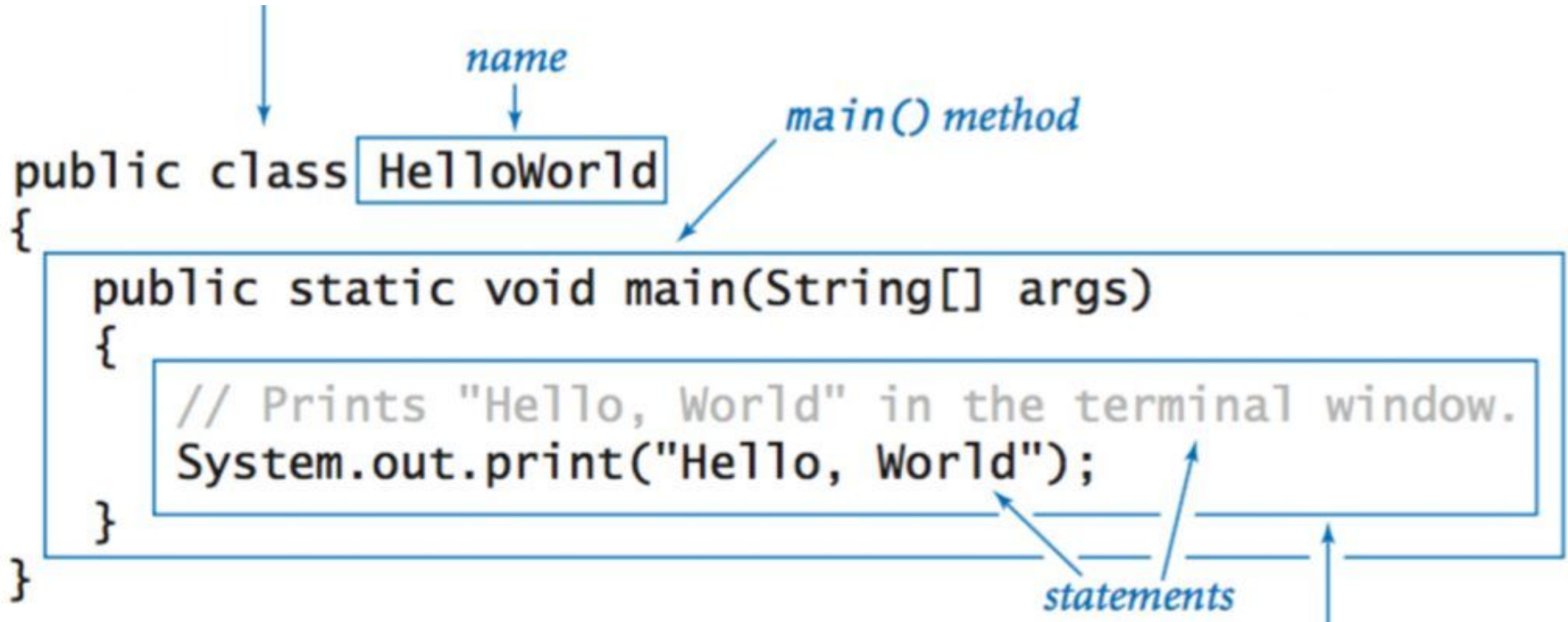
```
switch ( expression ){
    case value1 :
        statement-list1
    case value2 :
        statement-list2
    case value3 :
        statement-list3
    case ...
}
```

If expression  
matches value3,  
control jumps  
to here

```
public void processData()
{
    do
    {
        int data = getData();
        if(data < 0)
            performOperation1(data);
        else
            performOperation2(data);
    }
    while(hasMoreData());
}
```



# Statements, Code Blocks, Methods and more...



The diagram illustrates the structure of a Java program with the following code and annotations:

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        // Prints "Hello, World" in the terminal window.
        System.out.print("Hello, World");
    }
}
```

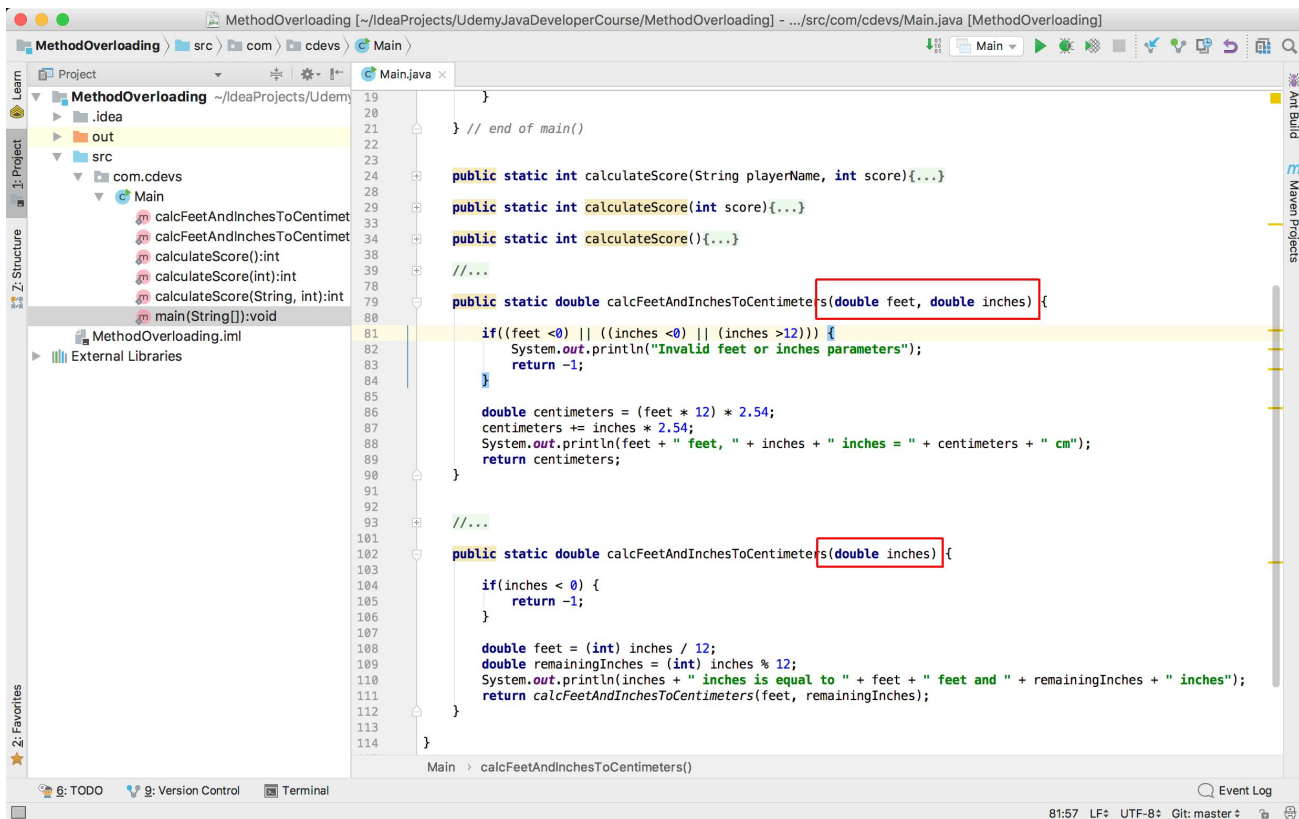
Annotations and their targets:

- An arrow points to the `public class` keyword.
- An arrow labeled *name* points to the `HelloWorld` class name, which is enclosed in a blue box.
- An arrow labeled *main() method* points to the `main` method signature.
- A large blue box encloses the entire `main` method body.
- Inside the `main` method body, a smaller blue box encloses the two statements: `// Prints "Hello, World" in the terminal window.` and `System.out.print("Hello, World");`.
- An arrow labeled *statements* points to the two lines of code within the inner box.
- Another arrow points to the closing curly brace of the `main` method.

# Methods and Method Overloading

Overloaded Methods:

- MUST have different number and/or type of parameters.
- CAN have different return type
- CAN have different access modifier



```
MethodOverloading [~/IdeaProjects/UdemyJavaDeveloperCourse/MethodOverloading] - .../src/com/cdevs/Main.java [MethodOverloading]
MethodOverloading > src > com > cdevs > Main
Main.java x
Project: MethodOverloading
  .idea
  out
  src
    com.cdevs
      Main
        calcFeetAndInchesToCentimeters(double, double)
        calcFeetAndInchesToCentimeters(double, double)
        calculateScore():int
        calculateScore(int):int
        calculateScore(String, int):int
        main(String[]):void
MethodOverloading.iml
External Libraries

19 }
20 } // end of main()
21
22
23
24 public static int calculateScore(String playerName, int score){...}
25
26 public static int calculateScore(int score){...}
27
28 public static int calculateScore(){...}
29
30 //...
31
32 public static double calcFeetAndInchesToCentimeters(double feet, double inches) {
33
34     if((feet < 0) || ((inches < 0) || (inches > 12))) {
35         System.out.println("Invalid feet or inches parameters");
36         return -1;
37     }
38
39     double centimeters = (feet * 12) * 2.54;
40     centimeters += inches * 2.54;
41     System.out.println(feet + " feet, " + inches + " inches = " + centimeters + " cm");
42     return centimeters;
43 }
44
45 //...
46
47 public static double calcFeetAndInchesToCentimeters(double inches) {
48
49     if(inches < 0) {
50         return -1;
51     }
52
53     double feet = (int) inches / 12;
54     double remainingInches = (int) inches % 12;
55     System.out.println(inches + " inches is equal to " + feet + " feet and " + remainingInches + " inches");
56     return calcFeetAndInchesToCentimeters(feet, remainingInches);
57 }
58
59 }
60
61 Main > calcFeetAndInchesToCentimeters()
```

# Novice Java programmers often confuse **Method Overloading vs. Overriding**

<https://www.programcreek.com/2009/02/overriding-and-overloading-in-java-with-examples/>

## Overloading

```
class Dog{  
    public void bark(){  
        System.out.println("woof ");  
    }  
  
    //overloading method  
    public void bark(int num){  
        for(int i=0; i<num; i++)  
            System.out.println("woof ");  
    }  
}
```

Same Method Name,  
Different Parameter

## Overriding

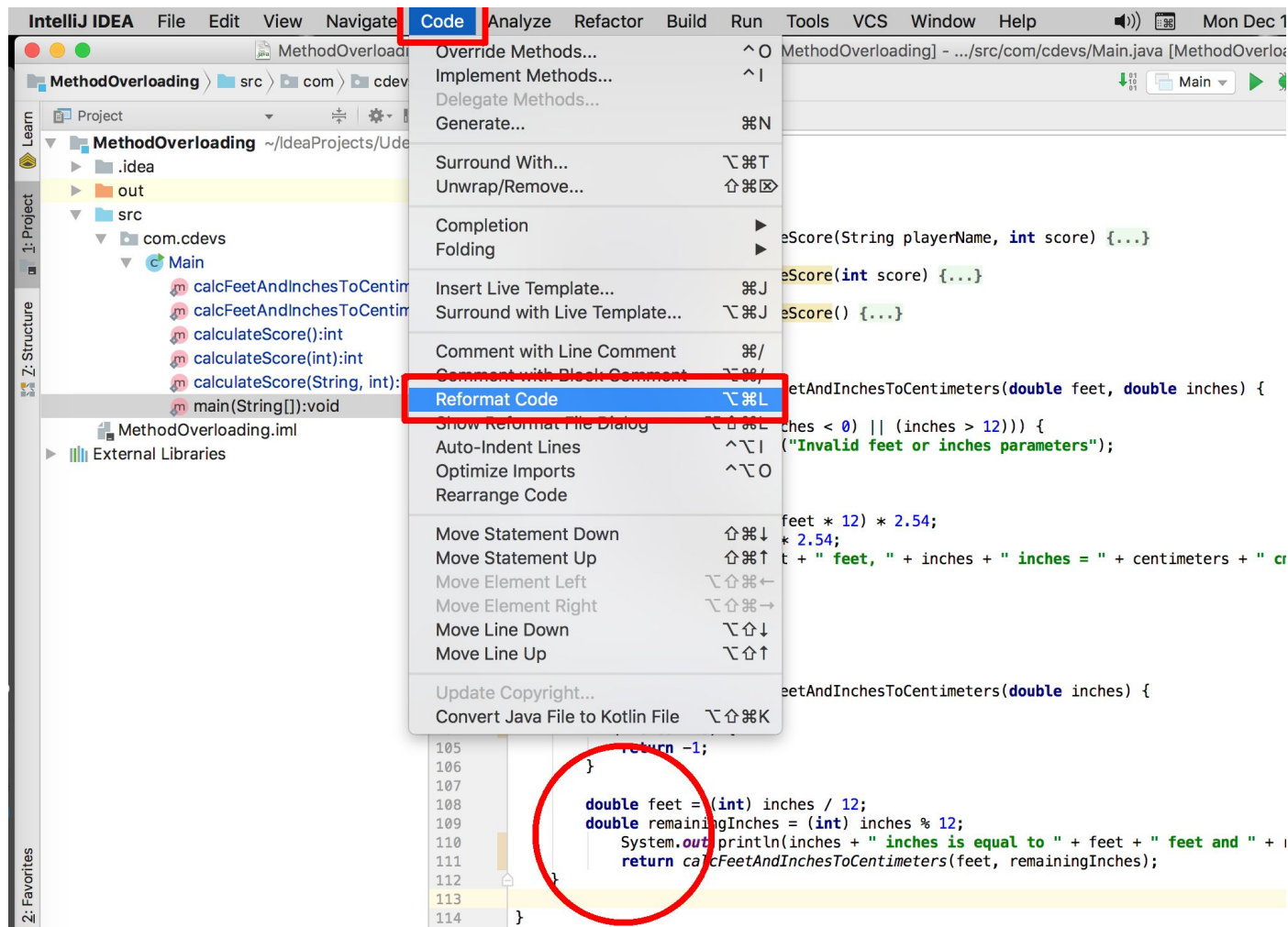
```
class Dog{  
    public void bark(){  
        System.out.println("woof ");  
    }  
}  
  
class Hound extends Dog{  
    public void sniff(){  
        System.out.println("sniff ");  
    }  
  
    public void bark(){  
        System.out.println("bowl");  
    }  
}
```

Same Method Name,  
Same parameter



# Reformat Code

for when  
your typing  
gets out of  
control and  
you want to  
tidy up a bit  
before you  
commit and  
push to  
GitHub





Hex



## Assembler



```
main()
{
    char ch,*text;
    int bit;
    int data;
    FILE *f;
    struct
    {
        int
        int
    }
}
```

C

Fortran

C++

A man with a beard and long hair, wearing a simple brown tunic, is shown in profile, walking and carrying a large, complex mechanical device. The device has a long, thin, curved blade or arm extending from its side. The man's expression is one of concentration or effort. The background is a plain, light-colored surface.

# Java



Ruby