PROGRAMMING FOR ARTISTS

DT8114 PhD seminar on the book:

Programming Interactivity by Joshua Noble

Jordi Puig
jordi.puig@q2s.ntnu.no

TABLE OF CONTENTS

openFrameworks

- Created by Zach Lieberman, Theo Watson Arturo Castro and Chris O'Shea
- "It is a framework for artists and designers working with interactive design and media art"
- It is similar to what processing is to java, oF is to c++
- c++ is a very low level language!

EXAMPLES

Hand from Above http://www.openframeworks.cc/gallery/hand-from-above

My secret heart http://vimeo.com/2131989

Multitouch proof of concept http://vimeo.com/5414506

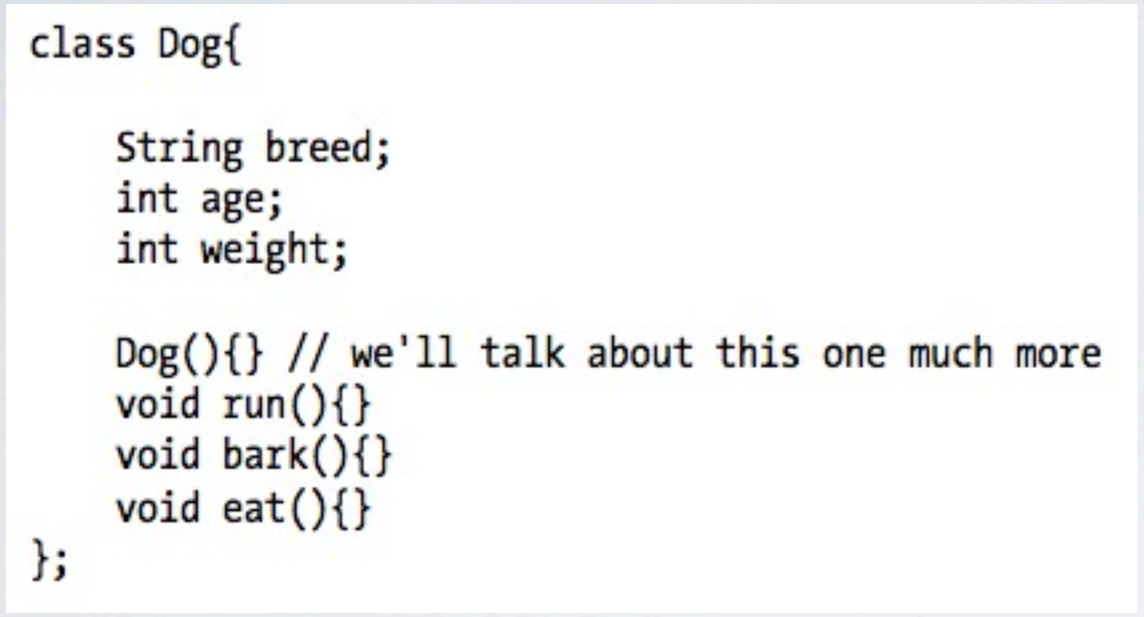
Flick-Flock and LummoBlocks on http://wasawi.com

OBJECT- ORIENTED PROGRAMMING (OOP)

- keeps better understanding of your code
- more organized
- easier to plan and expand
- lots of programing languages are OO

OOP: BASICS OF A CLASS

a Processing class



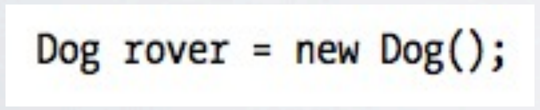
```
class Dog{

    String breed;
    int age;
    int weight;

    Dog(){} // we'll talk about this one much more
    void run(){}
    void bark(){}
    void eat(){}
};
```

the class (noun)

properties (adjective)

methods (verbs)

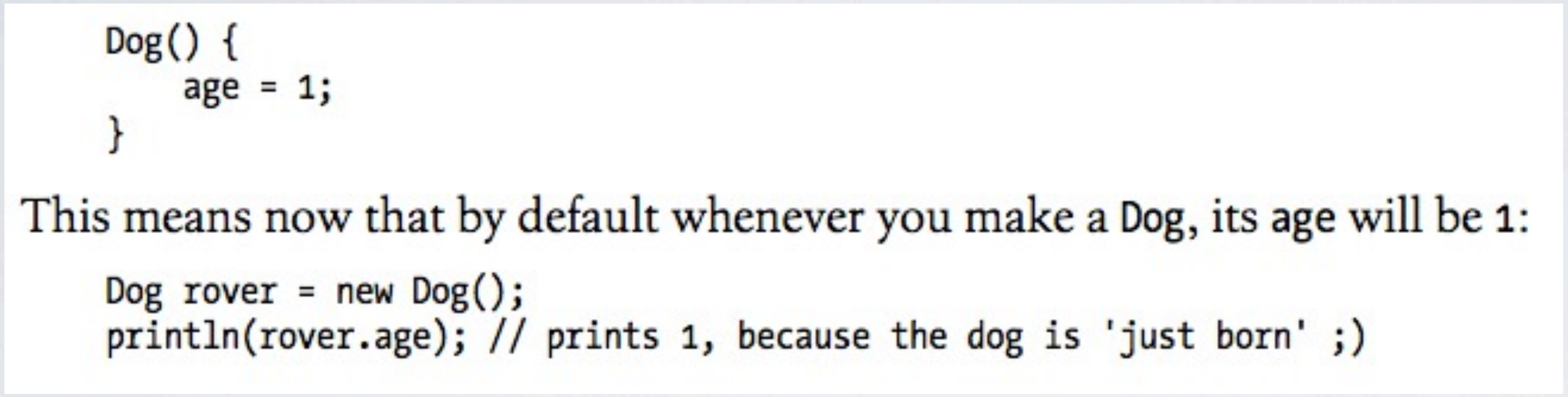
- A class is a grouping of variables and methods into an object that contains and controls access to them all.
- Differentiate properties and actions (methods).



```
Dog rover = new Dog();
```

OOP: THE CONSTRUCTOR

a Processing class



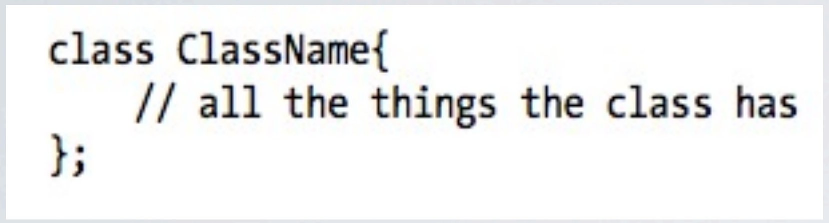
```
    Dog() {
        age = 1;
    }
```

This means now that by default whenever you make a Dog, its age will be 1:

```
    Dog rover = new Dog();
    println(rover.age); // prints 1, because the dog is 'just born' ;)
```

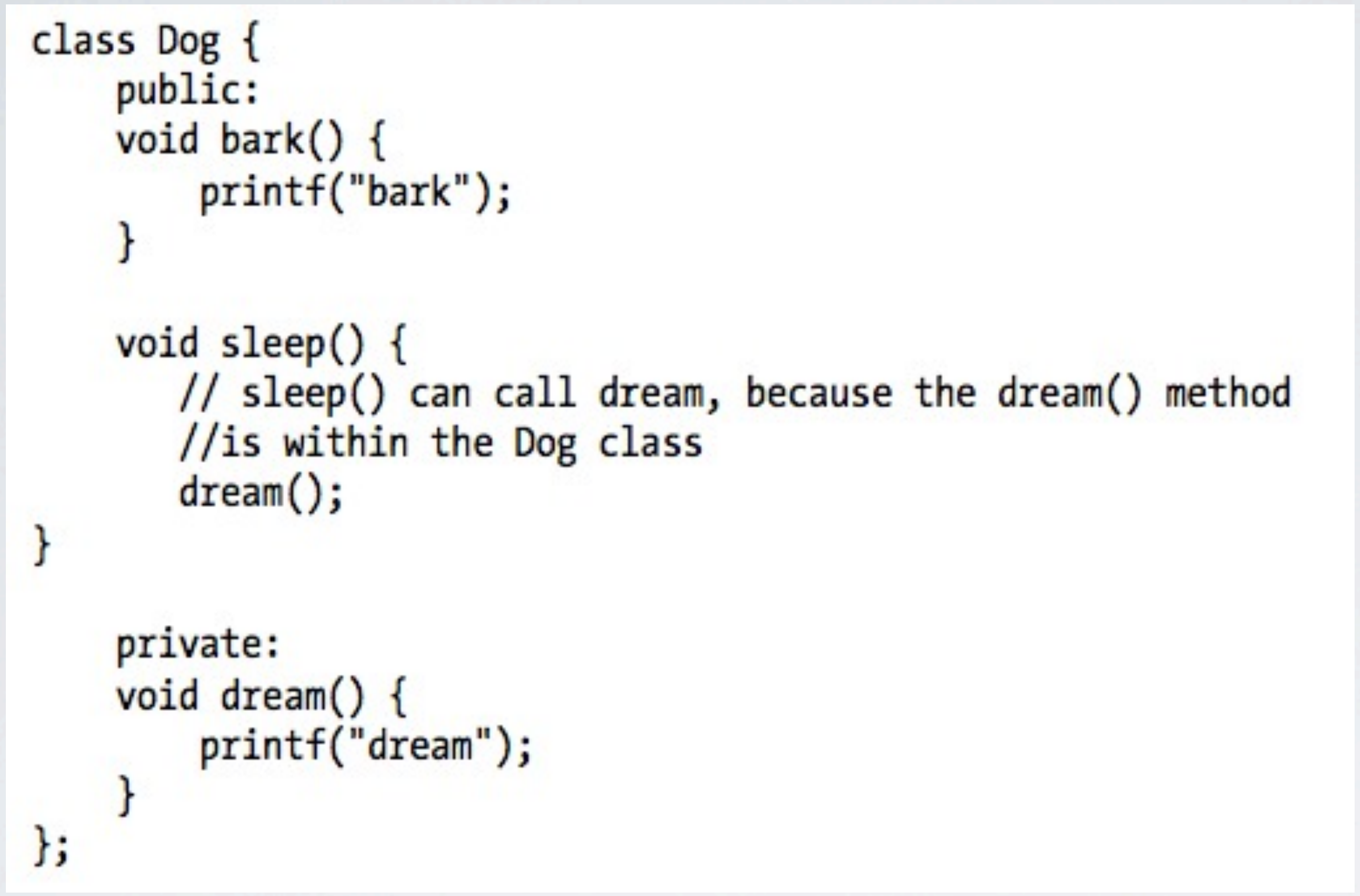
- the constructor performs any actions that you want to perform when the class is first created.
- it runs only one time at the beginning.
- it is mostly used to give value to its properties.

OOP: CLASS RULES


```
class ClassName{
    // all the things the class has
};
```

- they start with UpperCase! (if two words, together)
- should have good method names
- classes should be nouns and methods should be verbs
- For instance, a Dog should have run(), bark(), and eat() methods, but not a paper() method. A method called fetchThePaper() would be far more appropriate, but ultimately, your code is your own, and you can do whatever you like with it.

OOP: PUBLIC - PRIVATE

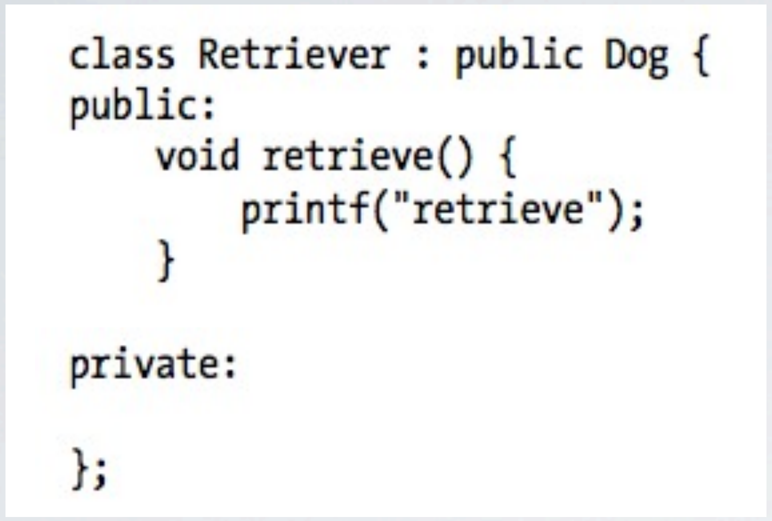


```
class Dog {
    public:
    void bark() {
        printf("bark");
    }

    void sleep() {
       // sleep() can call dream, because the dream() method
       //is within the Dog class
       dream();
}

    private:
    void dream() {
        printf("dream");
    }
};
```

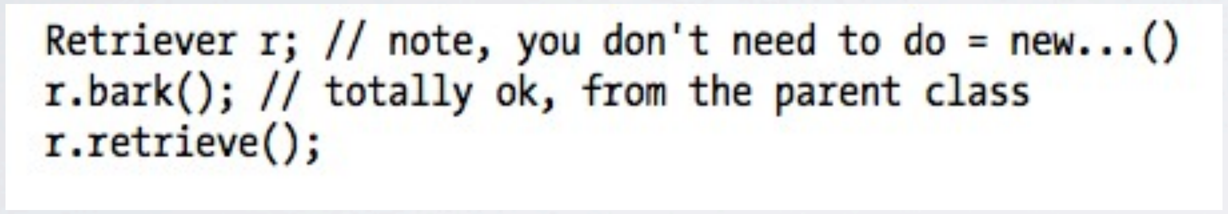
- Public ones are available to the outside world, which means that other classes can use those properties and methods.

- Private properties are not available to the outside world, only to methods and variables that are inside the class, which means that other classes cannot use those properties and methods.

OOP: INHERITANCE



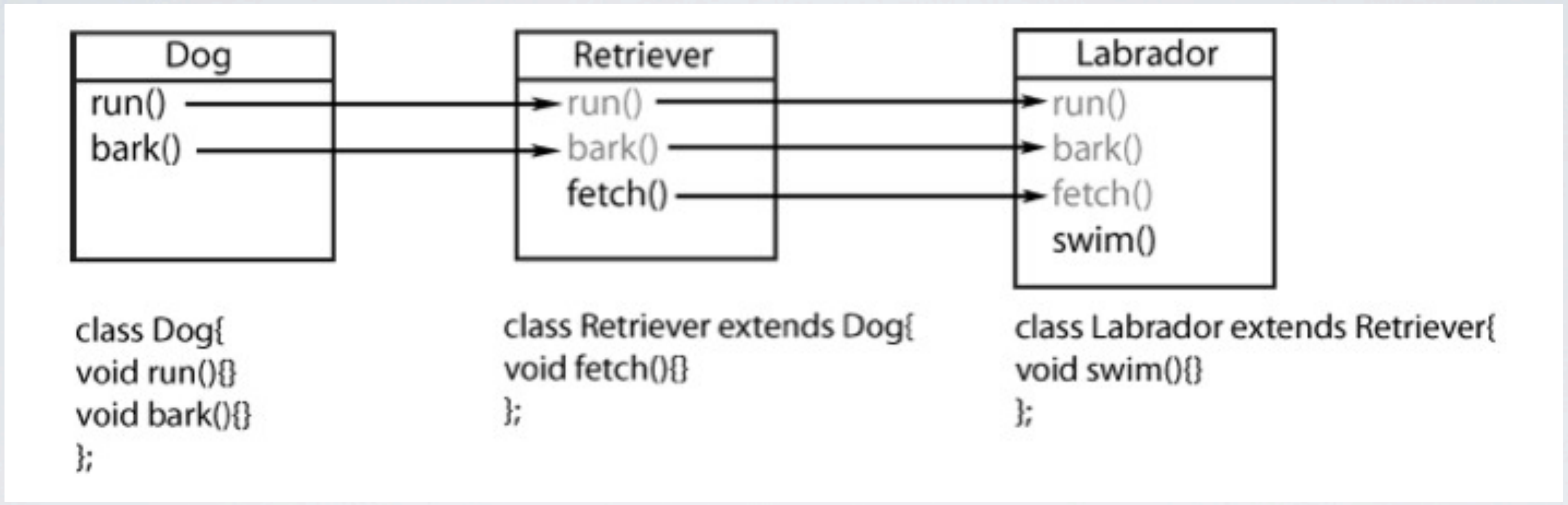
```
class Retriever : public Dog {
public:
    void retrieve() {
        printf("retrieve");
    }

private:

};
```



```
Retriever r; // note, you don't need to do = new...()
r.bark(); // totally ok, from the parent class
r.retrieve();
```

- Inheritance means that a class you're making extends another class, getting all of its variables and methods that have been marked public.

OOP: INHERITANCE



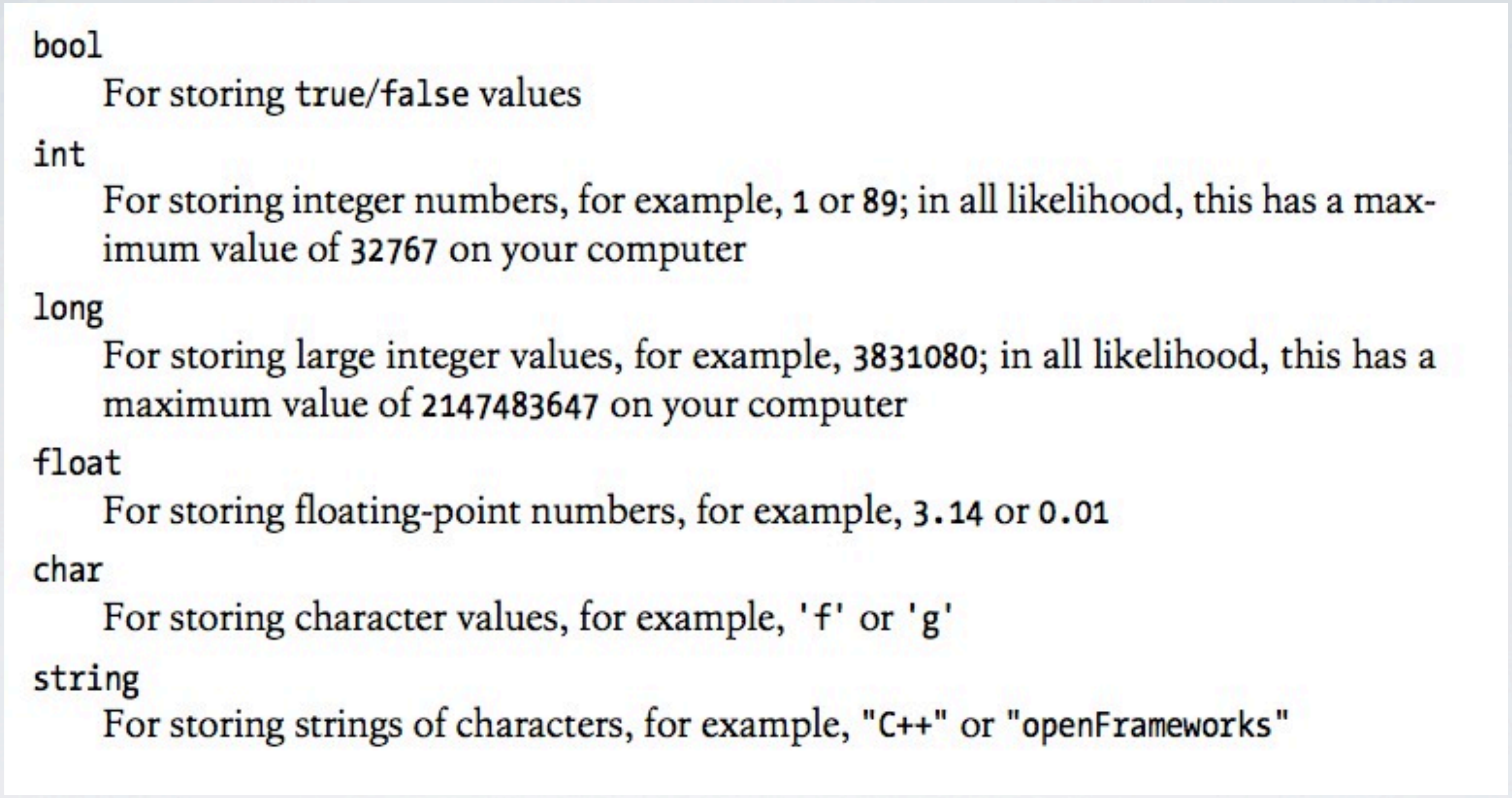
```
class Dog{
void run(){}
void bark(){}
};
```

```
class Retriever extends Dog{
void fetch(){}
};
```

```
class Labrador extends Retriever{
void swim(){}
};
```

- this is a nice way to keep many pieces of code clean and organized, also a good metaphor of our categorization system

C++: VARIABLE TYPES



`bool`
For storing `true`/`false` values

`int`
For storing integer numbers, for example, `1` or `89`; in all likelihood, this has a maximum value of `32767` on your computer

`long`
For storing large integer values, for example, `3831080`; in all likelihood, this has a maximum value of `2147483647` on your computer

`float`
For storing floating-point numbers, for example, `3.14` or `0.01`

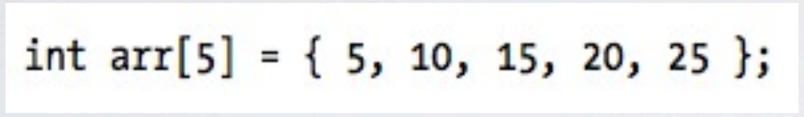
`char`
For storing character values, for example, `'f'` or `'g'`

`string`
For storing strings of characters, for example, `"C++"` or `"openFrameworks"`

- this are the most common types used, there are many others but lets start with few.

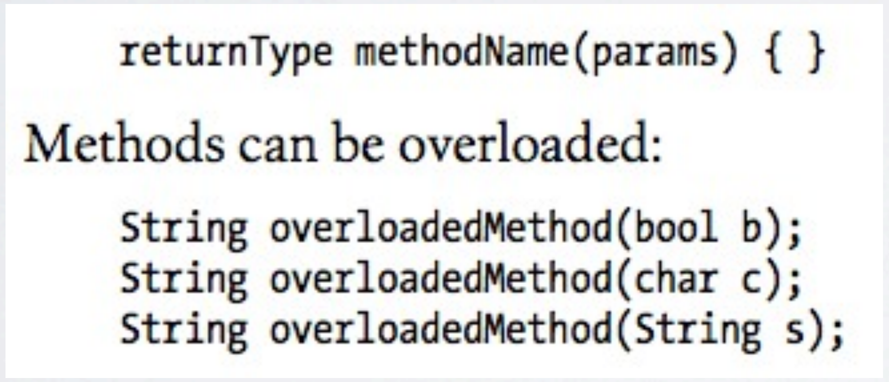
C++: OTHERS

- Arrays



```
int arr[5] = { 5, 10, 15, 20, 25 };
```

- Methods



```
    returnType methodName(params) { }
```

Methods can be overloaded:

```
    String overloadedMethod(bool b);
    String overloadedMethod(char c);
    String overloadedMethod(String s);
```

C++: FILES

the **.cpp** will contain the following:

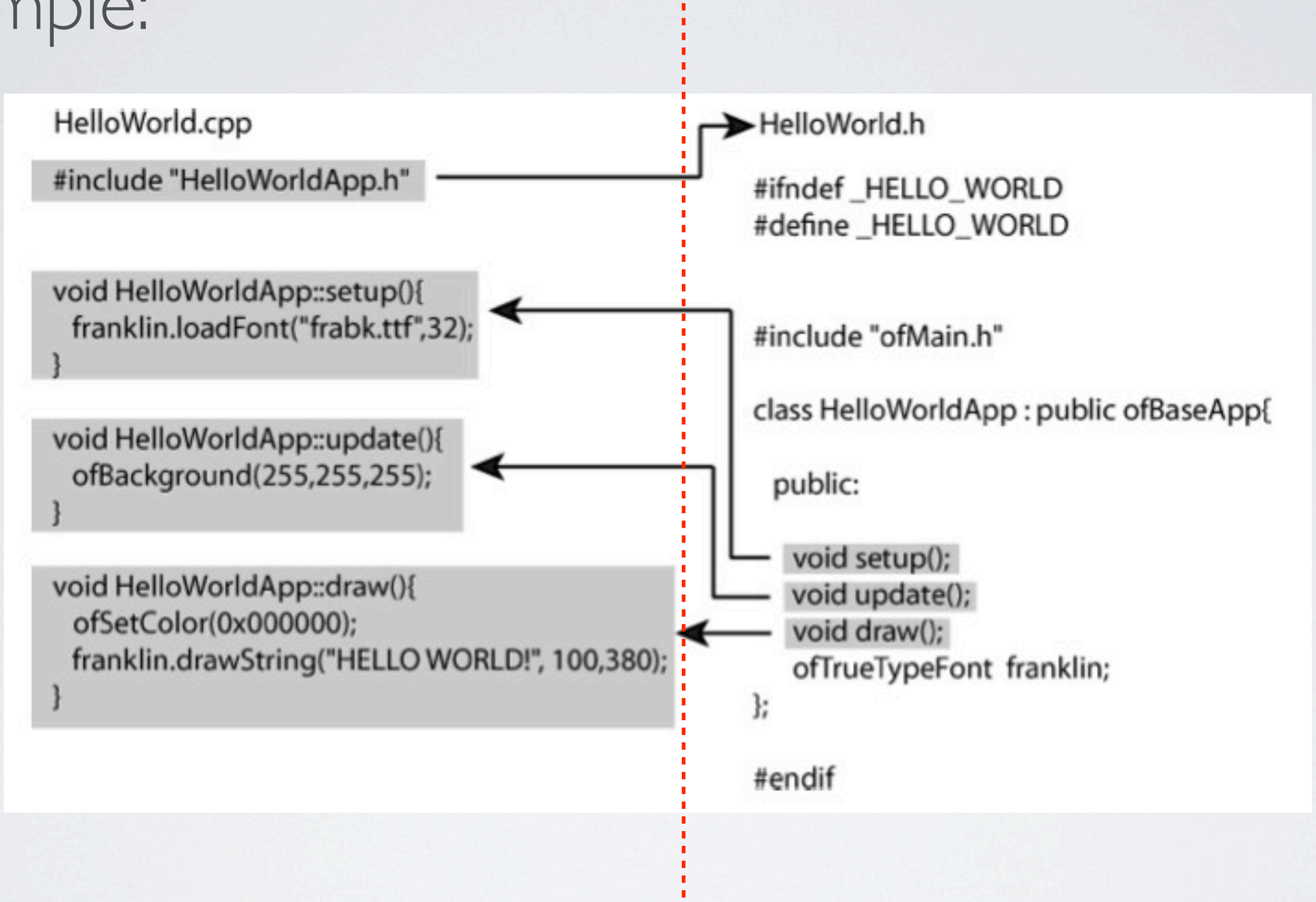
- The actual definition of any methods that the class defines

the **.h** file will contain the following:

- Any import statements that the class needs to make
- The name of the class
- Anything that the class extends (more on this later)
- Declarations of variables that the class defines (sometimes referred to as properties)
- Declarations of methods that the class defines

C++: FILES

Example:



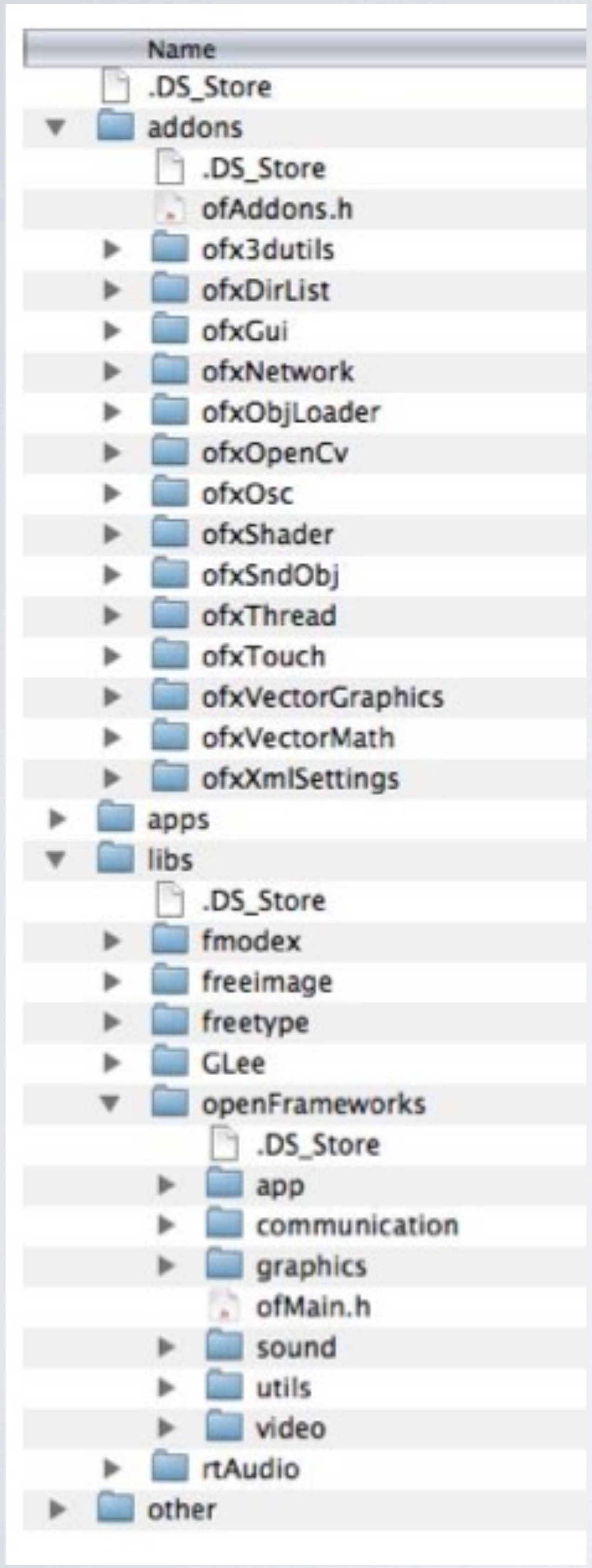
HelloWorld.cpp

```
#include "HelloWorldApp.h"

void HelloWorldApp::setup(){
  franklin.loadFont("frabk.ttf",32);
}

void HelloWorldApp::update(){
  ofBackground(255,255,255);
}

void HelloWorldApp::draw(){
  ofSetColor(0x000000);
  franklin.drawString("HELLO WORLD!", 100,380);
}
```

HelloWorld.h

```
#ifndef _HELLO_WORLD
#define _HELLO_WORLD

#include "ofMain.h"

class HelloWorldApp : public ofBaseApp{

  public:

    void setup();
    void update();
    void draw();
    ofTrueTypeFont  franklin;
};

#endif
```

GETTING STARTED WITH OF



```
Name
.DS_Store
▼ addons
    .DS_Store
    ofAddons.h
  ▶ ofx3dutils
  ▶ ofxDirList
  ▶ ofxGui
  ▶ ofxNetwork
  ▶ ofxObjLoader
  ▶ ofxOpenCv
  ▶ ofxOsc
  ▶ ofxShader
  ▶ ofxSndObj
  ▶ ofxThread
  ▶ ofxTouch
  ▶ ofxVectorGraphics
  ▶ ofxVectorMath
  ▶ ofxXmlSettings
▶ apps
▼ libs
    .DS_Store
  ▶ fmodex
  ▶ freeimage
  ▶ freetype
  ▶ GLee
  ▼ openFrameworks
      .DS_Store
    ▶ app
    ▶ communication
    ▶ graphics
      ofMain.h
    ▶ sound
    ▶ utils
    ▶ video
  ▶ rtAudio
▶ other
```

addons:

Contains all the added-on features for openFrameworks that have been con- tributed by users over the past year or so.

apps:

This is where your programs should be stored. Examples are here too.

libs:

This is where the libraries that oF relies on are stored.

openFrameworks:

Contains the core of the oF framework within six folders.

GETTING STARTED WITH OF

Your computer and OS matters!

c++ is platform dependent so each OS will run a different IDE and a different oF package

Windows:

Code::Blocks or Visual Studio

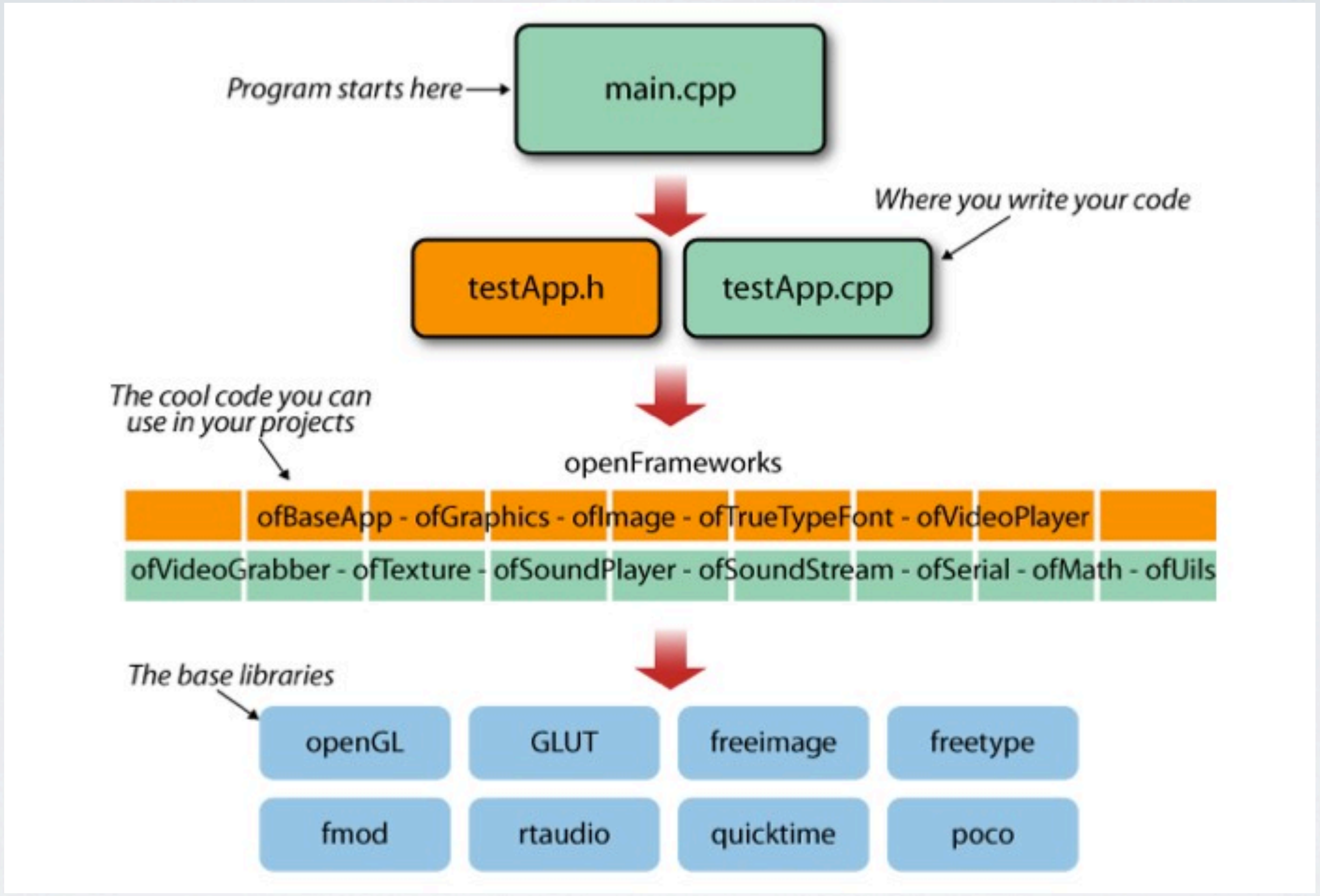
Mac OS X:

Xcode

Linux:

Code::Blocks or makefiles

GETTING STARTED WITH OF


Program starts here
main.cpp
Where you write your code
testApp.h
testApp.cpp
The cool code you can use in your projects
openFrameworks
ofBaseApp - ofGraphics - ofImage - ofTrueTypeFont - ofVideoPlayer
ofVideoGrabber - ofTexture - ofSoundPlayer - ofSoundStream - ofSerial - ofMath - ofUils
The base libraries
openGL
GLUT
freeimage
freetype
fmod
rtaudio
quicktime
poco

GETTING STARTED WITH OF

```
setup(){
    Executed only at the beginning of the program
    Here we will initialize our variables.
}
update(){
    Executed every frame.
    Here we will compute something
}
draw(){
    Executed every frame to draw to screen
}
```

GETTING STARTED WITH OF

Other method callbacks (event handlers):

keyPressed

keyReleased

mouseMoved

mouseDragged

mousePressed

mouseReleased

windowResized

...

Lets start coding!

OVERVIEW LANGUAGES FOR ARTISTS

	oF	Processing	Max	VVVV
processing speed	10	5	8	9
graphics	good but slow to write	good but slow to run	good	the best
video	good but slow to write	not so good	good and fast to work	not so good
speed to code	very slow	fast	very fast	very fast
trendy	10	7	4	9
big apps	the best	not so good	not so good	good enough
platform	not-multi platform but supported	multi-platform	mac and win	only windows
license	GNU	GNU GPL	commercial	free only for private use
community	interactive art computer graphics	computational design	sound, theater, electroacustic music	traditionally VJ