E0-245: ASP

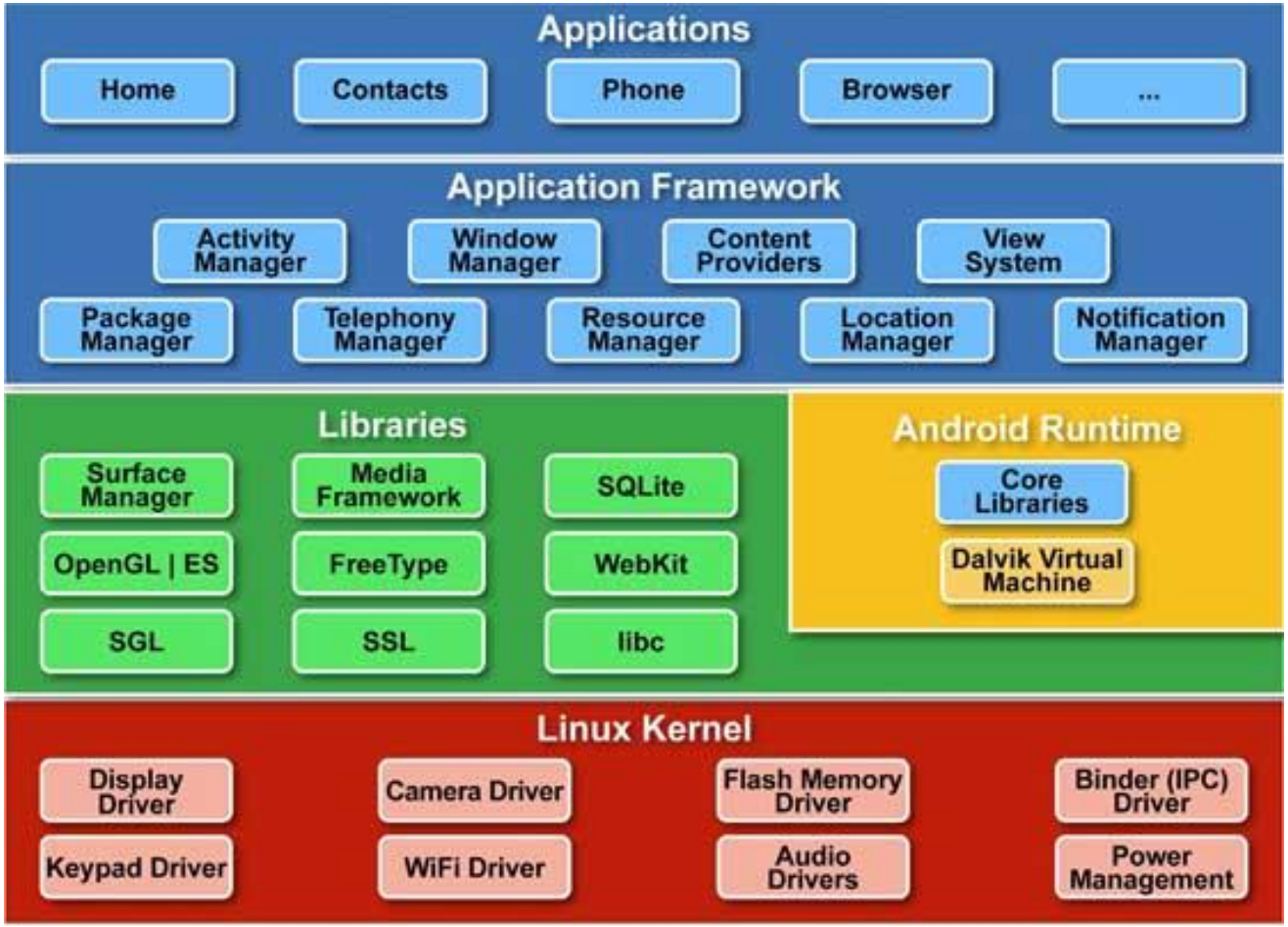
Lecture 5: OOPs Classes and Objects

Dipanjan Gope

Questions from Lecture 3

- Intent is asynchronous
- Sending notification a service?
- SensorEventListener component
- Why does BroadcastReciever not extend service?
- Can 2 activities use same screen space?
- Lifecycle: what is the limit to #of processes?

Android Stack



Ref: http://www.tutorialspoint.com/android/android_architecture.htm

Module 1: OOPs and DS

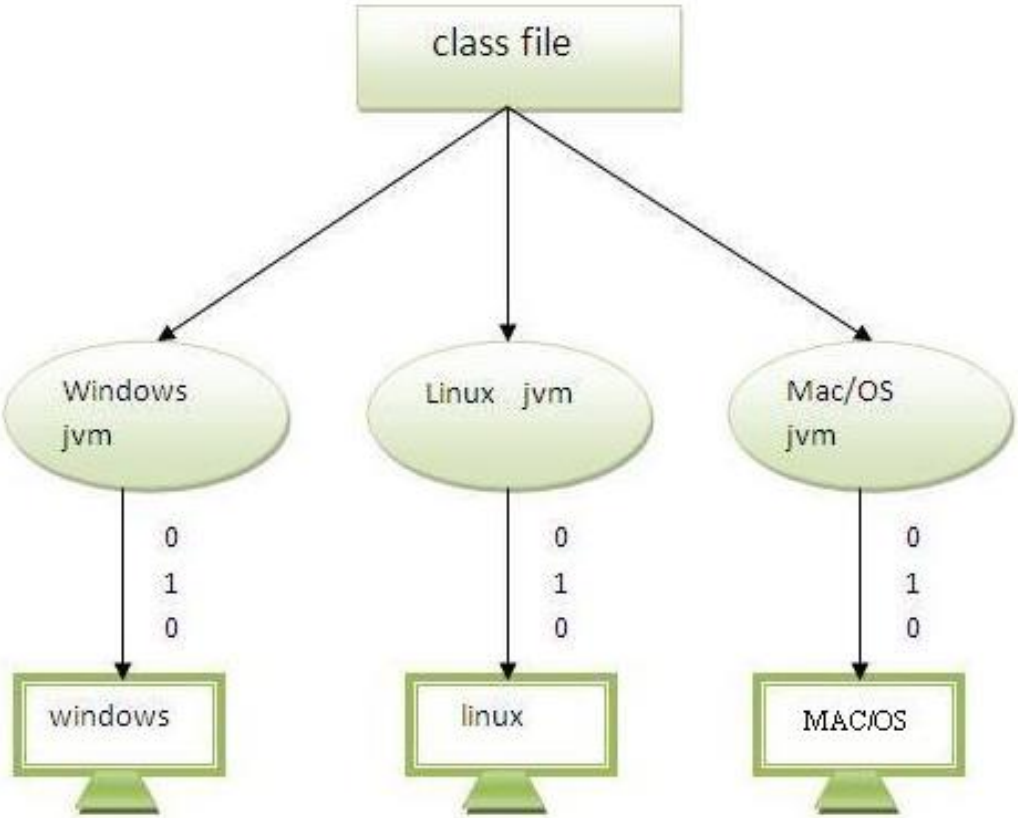
JAVA and C++ mainly: (others C# and Objective-C)

- Object oriented programing: Classes and objects
- Inheritance, polymorphism, abstract class, const
- Templates and generics
- Data structures
- Standard library, JCF, STL
- Complexity analysis
- Multithreading and synchronization
- Good programming styles

Code Conversion

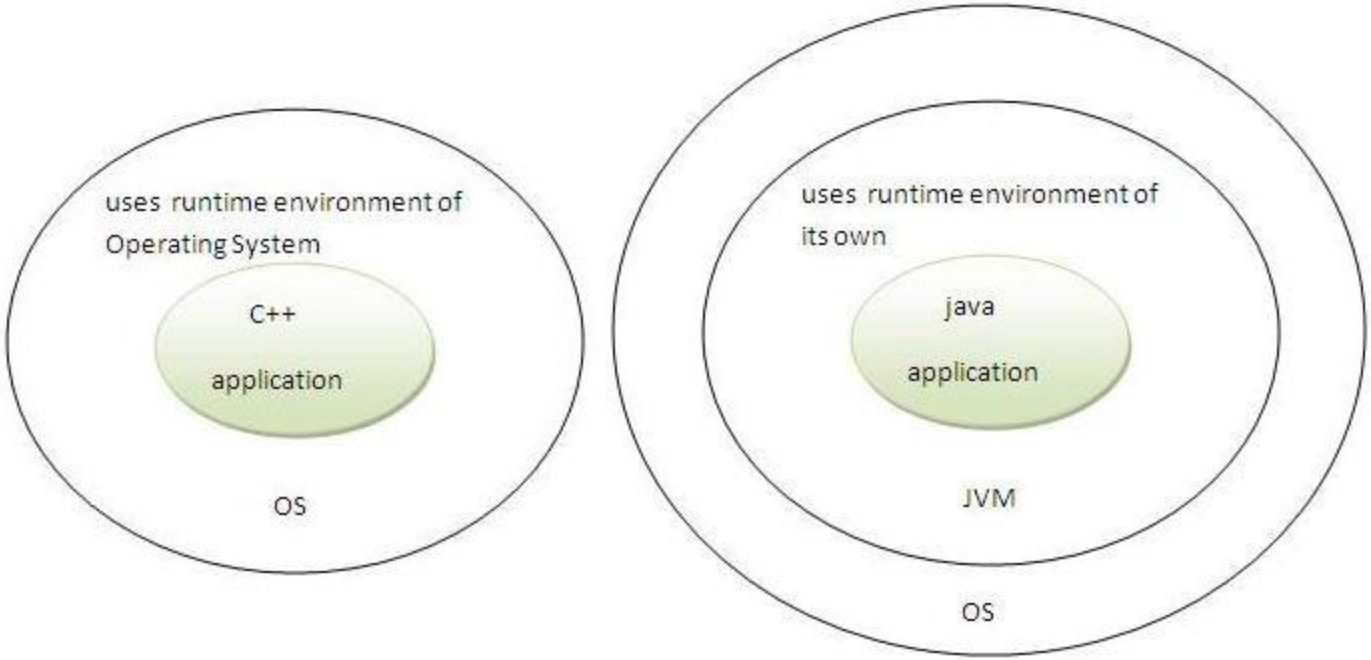
- Assembler
- Compiler
- Interpreter
- Just-in-time Compiler

JAVA VM



http://www.javatpoint.com/features-of-java

JAVA VM



http://www.javatpoint.com/features-of-java

Fun Facts

- First Programming Language
  - Fortran
  - 1957
  - IBM
- First OOPs Language
  - Simula
  - 1967
  - Norwegian Computing Center

Fun Facts

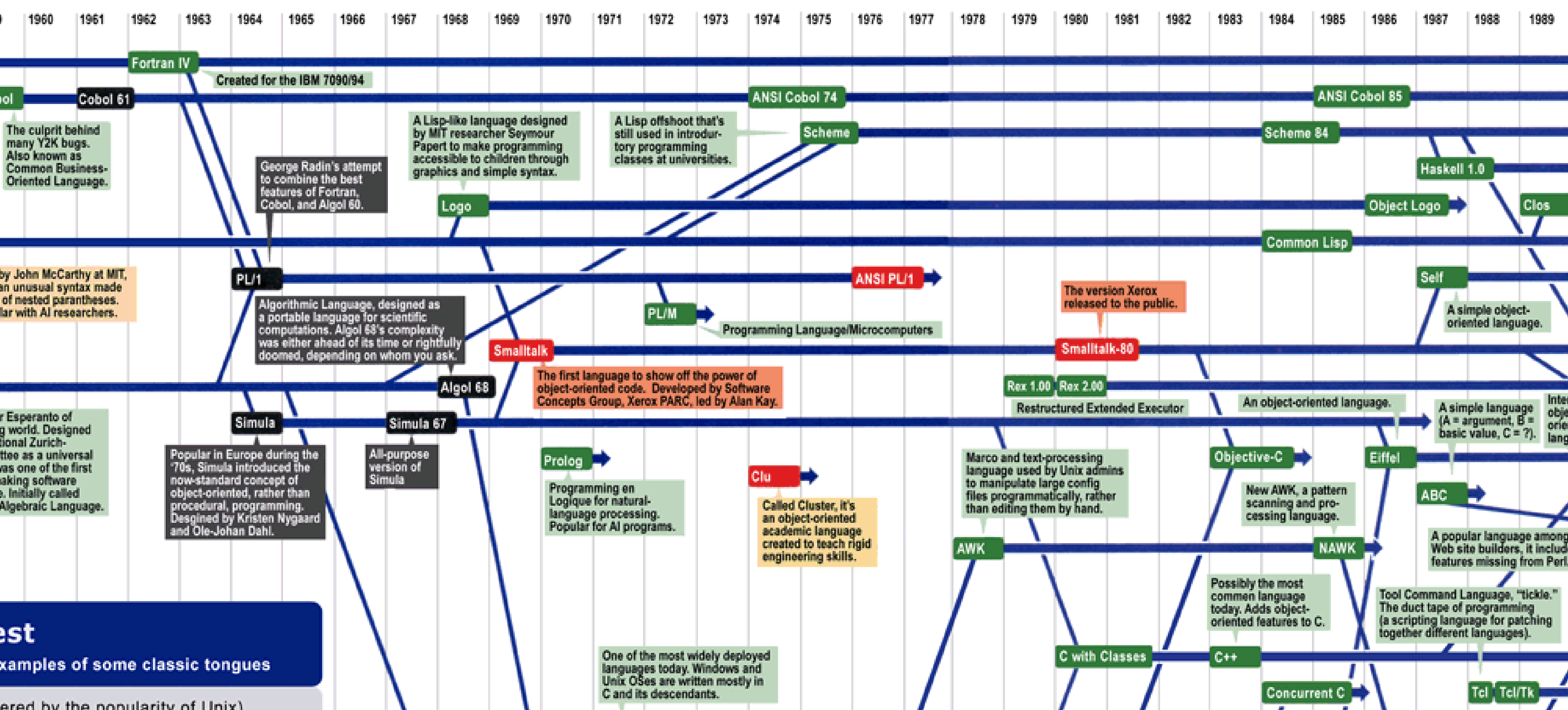
- Java
  - 1994
  - James Gosling, Sun Microsystems
  - Licensed by Netscape for navigator

- C language
  - 1972
  - Dennis Ritchie
  - Bell Labs

Evolution of Programming Languages

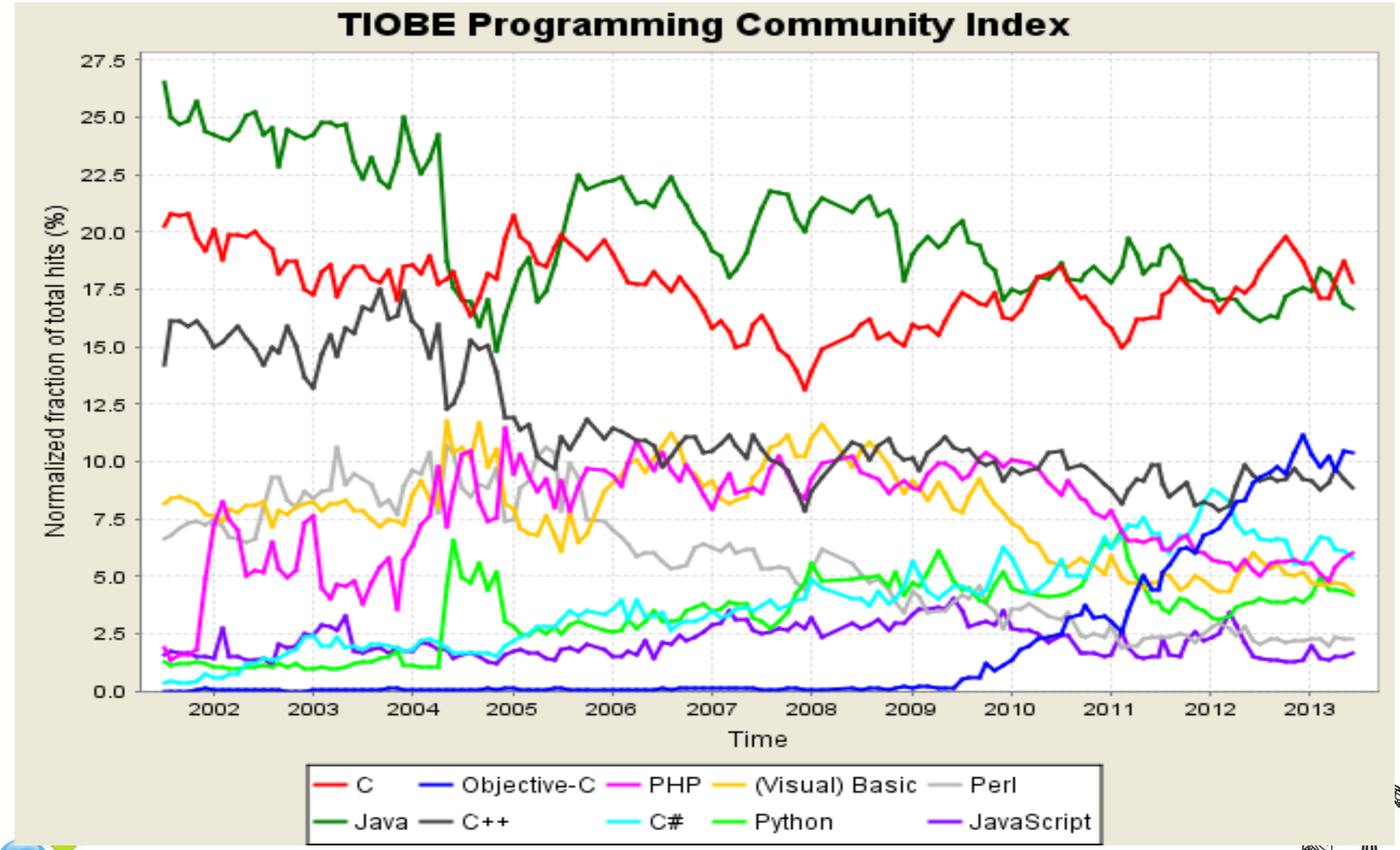
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.
An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working w... History Musuem in Silicon Valley to record and, in some cases, maintain lan... new compilers so our ever-changing hardware can grok the code. Why both... us about the state of software practice, the minds of their inventors, and th... and economic forces that shaped history at the time," Booch explains. "Th... raw material for software archaeologists, historians, and developers to lear... what was brilliant, and what was an utter failure." Here's a peek at the stro... of programming's family tree. For a nearly exhaustive rundown, check out ... at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html*. - Micha...



Paul Boutin and Bret Hailpern at IBM Research and Todd Proebsting at Microsoft, The Retrocomputing Museum, and Gio Wiederhold at Stanford University.

Top 10 Programming Languages



http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Recap: Pointers and References

int a; # how many bytes?

int *pA; # how many bytes?

int &rA = a; # how many bytes?

Recap: Pointers vs References

	Pointers	References
NULL	Yes	No
Point changed to a different object	Yes	No
Initialization	Anytime	On creation

When to use pointers?

- Pointer arithmetic
- NULL initialization

Object Oriented Programming System

- Class
- Object
- Polymorphism
- Inheritance
- Abstraction
- Encapsulation
- Overloading

Advantages of OOPs

- Code Reuse and Recycling
- Encapsulation: Hide details, prevent tampering
- Design benefits
- Software maintenance

https://www.cs.drexel.edu/~introcs/Fa12/notes/06.1_OOP/Advantages.html?CurrentSlide=3

- Run-time errors become compiler errors

Class

```
class Rectangle
{
        # Methods
Public:
Private:
Protected:
        # Field: Member data
Public:
Private:
Protected:
}
```

Fields: Member data
Local Variables: Variables in piece of code
Parameters: Variables in function declaration

Public, Protected, Private

JAVA Modifiers

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

http://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

C++ Access Specifier

Specifier	Class	Project	Sub-class
public	Y	Y	Y
protected	Y	N	Y
no specifier	Same as private		
private	Y	N	N

Object

- Instance of a class
- Just like: int i;
- Occupies memory in RAM;

- E.g. Rectangle rectangle1;

Object: Example

```
class Student
{
private:
        float CGPA;
        int rollNumber;
        double height;
        int schoolID; // string schoolName;
}
```

Static

- Static member data:
  - common to all objects of the class
  - single memory location for all classes
  - e.g. static string schoolName

- Static member function:
  - belongs to class rather than object
  - may be invoked without object instance
  - can access static variables

Constructor

- Special member function
- Same name as the class
- No return type (not even void)
- Used for:

  - variable initialization

  - memory allocation (perhaps)

  - called on “new” or instantiation

Constructor

- Constructor overloading
  - Multiple forms of constructors (polymorph)
  - different input arguments
- Special types of constructors:
  - Default constructor (different for diff compilers)
  - Copy constructor

Constructor

JAVA reference-based

```
Rectangle rectangle1 = new Rectangle;

Rectangle rectangle2 = new Rectangle(1,1.5);
```

C++ pointer-based

```
Rectangle rectangle1;
Rectangle* pRectangle2 = new Rectangle;

Rectangle rectangle1(1,1.5);
Rectangle* pRectangle2 = new Rectangle(1,1.5);
```

Destructor (C++)

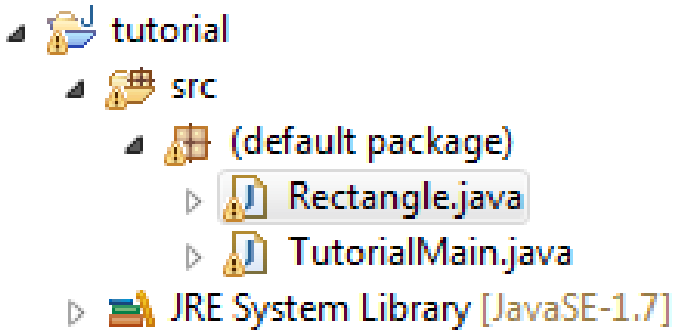
- Special member function
- Same name as the class with ~
- No return type (not even void)
- Used for:

  - memory creation

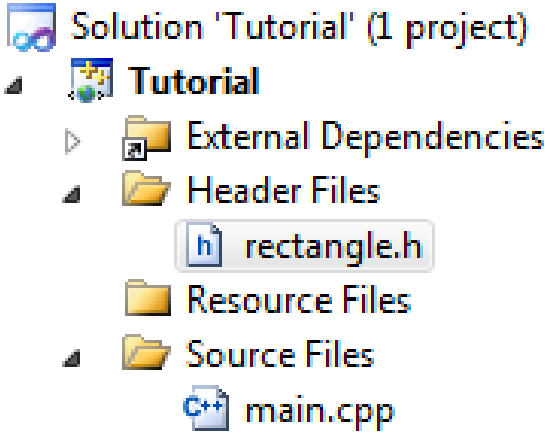
  - called on “out of scope” or “delete”
- JAVA: Garbage collector

Project Organization

JAVA



C++



Code Organization

JAVA

```java
public class Rectangle
{
    public Rectangle() {;}

    private int numNodes;
}
```

C++

```cpp
#include <iostream>

class Rectangle
{
public:
    Rectangle::Rectangle() {};
    Rectangle::~Rectangle() {};

protected:
    int numNodes;
};
```

Rectangle Class

```java
public class TutorialMain
{

    /**
     * @param args
     */
    public static void main(String[] args)
    {
        Rectangle rectangle1 = new Rectangle();
        System.out.println("Done");
    }
}
```

```cpp
#include <iostream>
#include "rectangle.h"


int main()
{
    Rectangle rectangle1;

    std::cout << "Done" << std::endl;
}
```

Main