EECS 1710
Programming for Digital Media

Lecture 2 :: Java Anatomy (Basics)

YORK UNIVERSITÉ UNIVERSITY

Announcements

- Course website (moodle + duplicate now available at):
    - https://www.eecs.yorku.ca/course_archive/2018-19/F/1710/

- New Lab Section (both sections run at same time)
    - LAB04 - Tuesday (LAS 1004)
    - Working on Lab 0 (on your own). Thursday group will start Lab 1.
    - Lab 1 will be posted before next lecture.

- Please try to read/work through tutorial on setting up a virtual machine (to use at home/remote access) in addition to Lab 0.

- Lab Schedule (so that you can find timeslots to enter & use the lab outside of your lab time):
    - http://eecs.lassonde.yorku.ca/wp-content/uploads/Labs/Schedules/prism.pdf

Introduction to Programming

Topics

- Anatomy of a program
- The declaration statement
- The assignment statement

Source code of examples will be posted online

Quick Tour

Example Program – Area.java

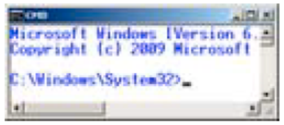
Figure 1.1

```
import java.lang.System;

public class Area
{
   public static void main(String[] args)
   {
      int width;
      width = 8;
      int height = 3;
      int area = width * height;
      System.out.println(area);
   }
}
```

Imports
Classes
Methods
Style

Demo using eclipse

Demo using



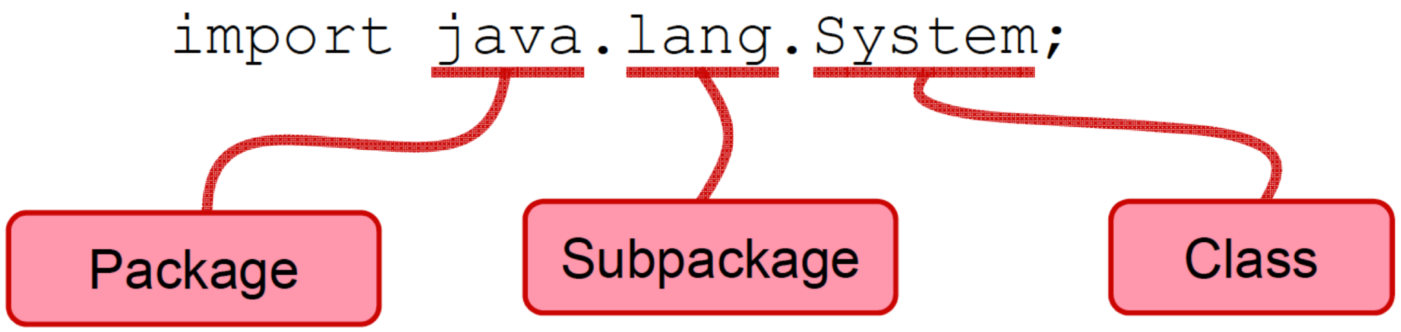
```
Microsoft Windows [Version 6.
Copyright (c) 2009 Microsoft

C:\Windows\System32>_
```

```java
import java.lang.System;

public class Area
{
   public static void main(String[] args)
   {
      int width;
      width = 8;
      int height = 3;
      int area = width * height;
      System.out.println(area);
   }
}
```

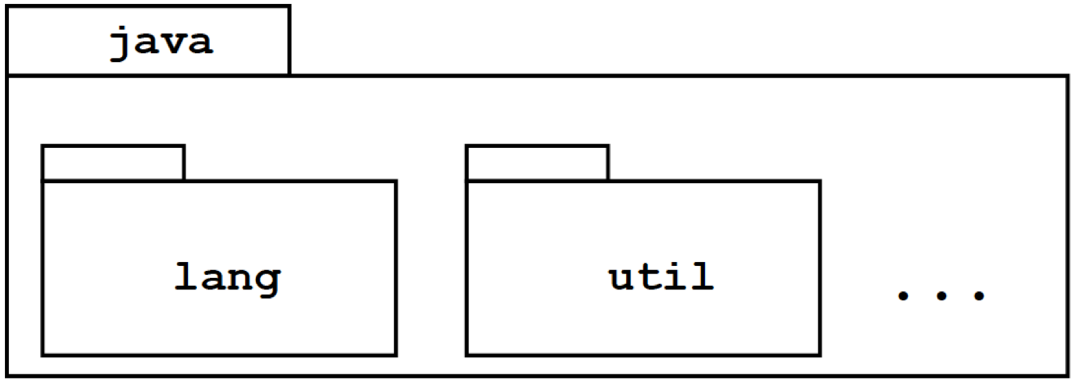
Imports

```java
import java.lang.System;

public class Area
{
   public static void main(String[] args)
   {
      int width;
      width = 8;
      int height = 3;
      int area = width * height;
      System.out.println(area);
   }
}
```

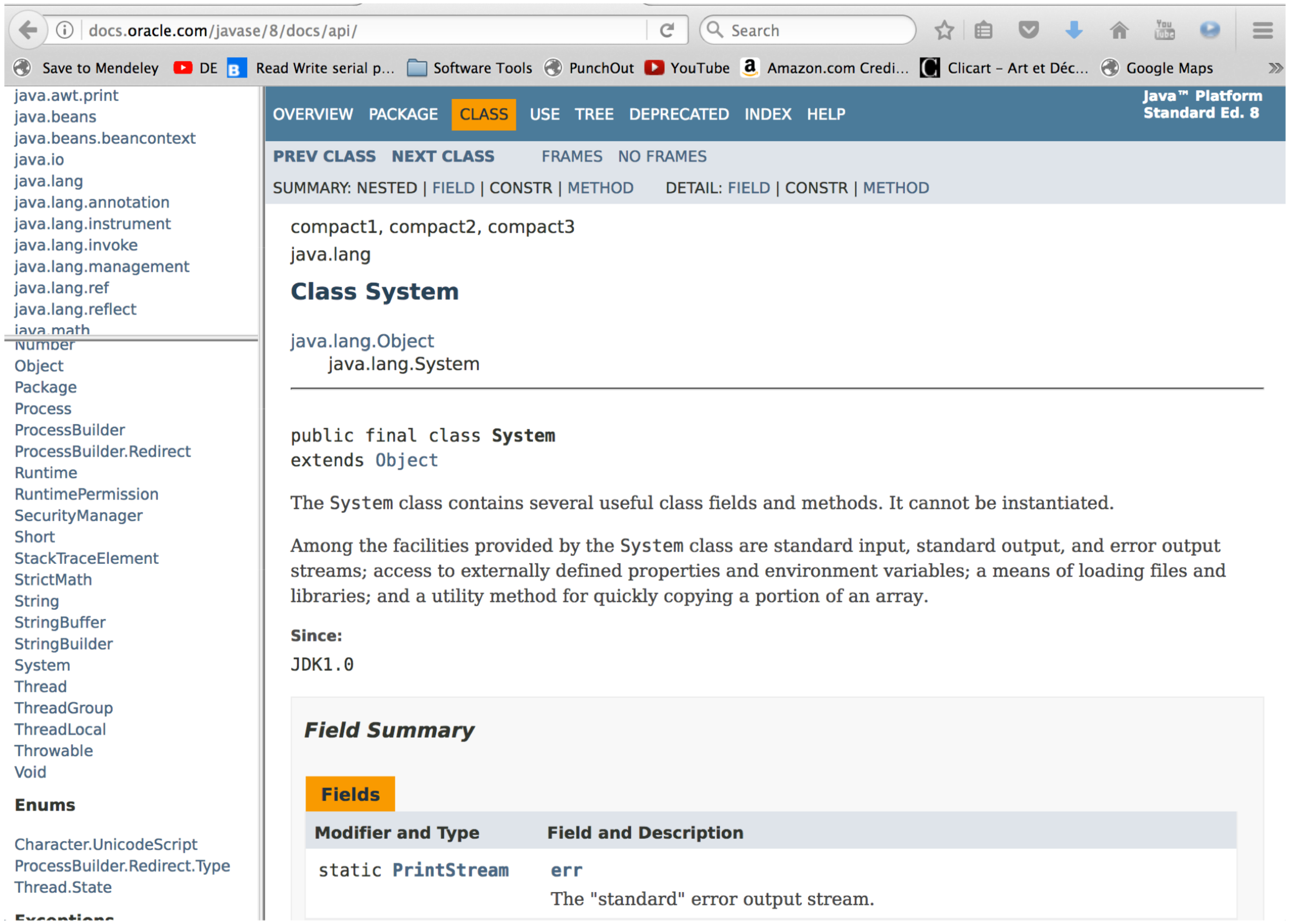
Imported Class
(= delegation)



```
import java.lang.System;
```



System Class API



java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.reflect
java.math

Number
Object
Package
Process
ProcessBuilder
ProcessBuilder.Redirect
Runtime
RuntimePermission
SecurityManager
Short
StackTraceElement
StrictMath
String
StringBuffer
StringBuilder
System
Thread
ThreadGroup
ThreadLocal
Throwable
Void

Enums

Character.UnicodeScript
ProcessBuilder.Redirect.Type
Thread.State

Exceptions

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

Java™ Platform Standard Ed. 8

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3
java.lang

Class System

java.lang.Object
    java.lang.System

```
public final class System
extends Object
```

The System class contains several useful class fields and methods. It cannot be instantiated.

Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

Since:
JDK1.0

Field Summary

Fields

Modifier and Type	Field and Description
static PrintStream	err The "standard" error output stream.

http://docs.oracle.com/javase/8/docs/api/

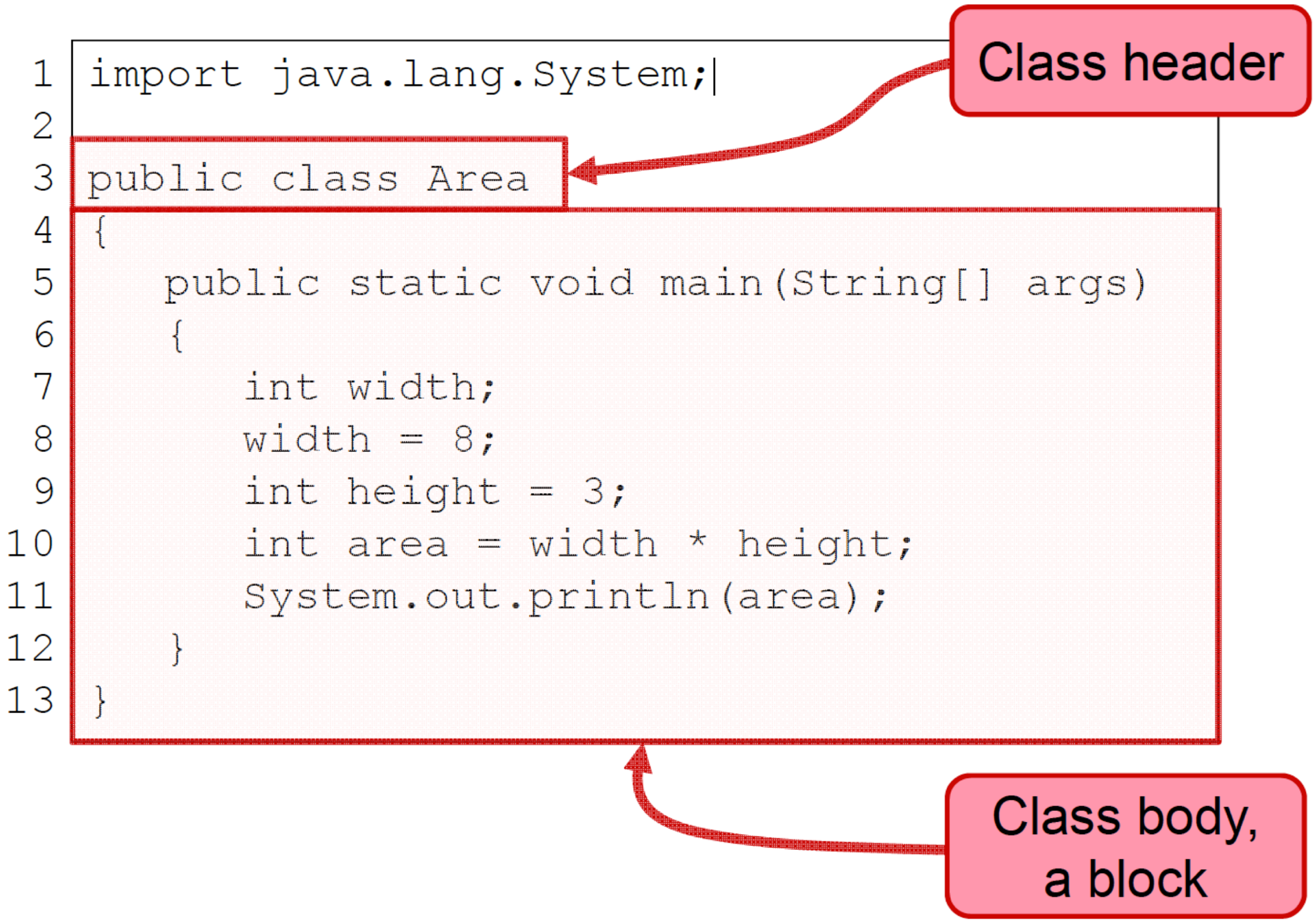
- Instead of

```
import java.lang.System;
```

  you can use an asterisk as a wildcard:

```
import java.lang.*;
```

- In this case, all classes in `java.lang` are available to be used in your program
- *Note*: `java.lang` is so important, that the above import is automatic (not so for other packages or subpackages)



```java
import java.lang.System;

public class Area
{
    public static void main(String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

```
import java.lang.System;

public class Area
{
    public static void main(String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

Method header

Method body, a block

Style

- Class naming convention:
  - Use title case unless an acronym
  - E.g., Math , UTL , StringTokenizer
- Method naming convention:
  - Use lowercase letters, except…
  - For multi-word names, capitalize the first letter of each subsequent word (no spaces)
  - E.g., main , equals , toString , isLeapYear
- Block layout:
  - Braces must align vertically and the all statements must
  - be left justified and indented by one tab position

Language Elements (key to a Java program):

Keywords	The keywords are the reserved words plus the literals `true`, `false`, and `null`.
Identifiers	Not be a reserved word; begin with a letter; character set is { `0-9, A-Z, a-z, _` }
Literals	Recognized by the presence of a number, ‘character’, “characters”, or `true`, `false`, or `null`.
Operators	The character set of operators: `= > < ! ~ ? : & | + - ^ * / %`
Separators	The separators: `. , ; ... ( ) [ ] { }`

A Note on Terminology

( )	Parentheses
[ ]	Brackets
{ }	Braces

Java Keywords

Reserved words:

`abstract`	`assert`				
`boolean`	`break`	`byte`			
`case`	`catch`	`char`	`class`	`const`	`continue`
`default`	`do`	`double`			
`else`	`enum`	`extends`			
`final`	`finally`	`float`	`for`		
`goto`					
`if`	`implements`	`import`	`instanceof`	`int`	`interface`
`long`					
`native`	`new`				
`package`	`private`	`protected`	`public`		
`return`					
`short`	`static`	`strictfp`	`super`	`switch`	`synchronized`
`this`	`throw`	`throws`	`transient`	`try`	
`void`	`volatile`				
`while`					

Literals: `true, false, null`

Language Elements in Area.java

```
import java.lang.System;

public class Area
{
   public static void main(String[] args)
   {
      int width;
      width = 8;
      int height = 3;
      int area = width * height;
      System.out.println(area);
   }
}
```

Keywords Identifiers Literals Operators Separators

Language Elements in Area.java

```java
import java.lang.System;

public class Area
{
    public static void main(String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

Keywords Identifiers Literals Operators Separators

Language Elements in Area.java

```
import java.lang.System;

public class Area
{
   public static void main(String[] args)
   {
      int width;
      width = 8;
      int height = 3;
      int area = width * height;
      System.out.println(area);
   }
}
```

Keywords **Identifiers** Literals Operators Separators

Language Elements in Area.java

```
import java.lang.System;

public class Area
{
   public static void main(String[] args)
   {
      int width;
      width = 8;
      int height = 3;
      int area = width * height;
      System.out.println(area);
   }
}
```

Keywords Identifiers **Literals** Operators Separators

Language Elements in Area.java

```
import java.lang.System;

public class Area
{
    public static void main(String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

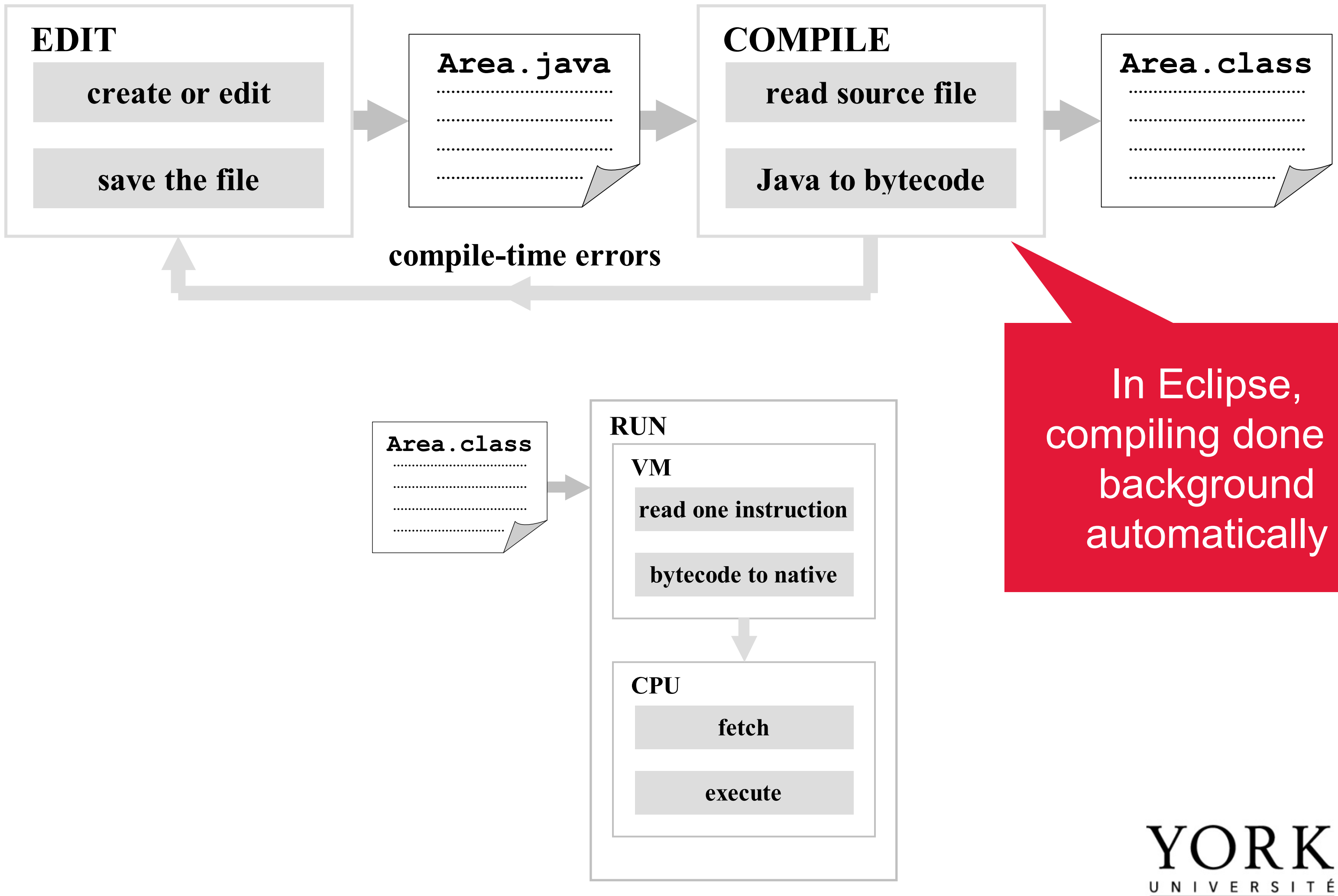
Keywords Identifiers Literals **Operators** Separators

Language Elements in Area.java

```
import java.lang.System;

public class Area
{
   public static void main(String[] args)
   {
      int width;
      width = 8;
      int height = 3;
      int area = width * height;
      System.out.println(area);
   }
}
```

Keywords Identifiers Literals Operators **Separators**

Program Execution



In Eclipse, compiling done in background automatically

Demo

- Quick discussion on OS (linux)
- Terminal vs. IDE (Integrated Development Environment)

- Example 1: terminal to compile & run
- Example 2: IDE to compile & run

Terminal

- Edit: (use a text editor.. e.g. gedit)

```
%  gedit  Area.java
```

- Compile:

```
% javac Area.java
```

- Run:

```
% java Area
```

Topics

- Anatomy of a program
- The declaration statement
- The assignment statement

Declaration Statement

- The statement (from Area.java)

```
int width;
```

is of the general form

```
type name;
```

The name of a primitive or non-primitive type, e.g., `int`, `double`

Name of an identifier (variable) to be associated with a memory block

Variable Scope

- Variables have *scope*
- A variable's scope is the variable's *enclosing block*
- The variable is not known outside of its scope

Variable Names

- Rules and guidelines for names of variables
  - Must be an identifier
  - Must not be in the scope of another variable with the same name
  - A good name reflects the content stored in the variable
  - Style
    - Use lowercase letters, but for multi-word names, capitalize the first letter of each subsequent word

Integer Types

- A type is a range of values and a set of operations on these values
- Operators: `+` (add), `–` (subtract), `*` (multiply), `/` (divide), `%` (remainder)
- Variations

Type	Range	Memory size
`byte`	≈ ±100	1 byte (= 8 bits)
`short`	≈ ±30,000	2 bytes (= 16 bits)
`int`	≈ $\pm 2 \times 10^{9}$	4 bytes (= 32 bits)
`long`	≈ $\pm 9 \times 10^{18}$	8 bytes (= 64 bits)

Default literal

As a literal, L or l suffix (e.g., `long x = 5L;`)

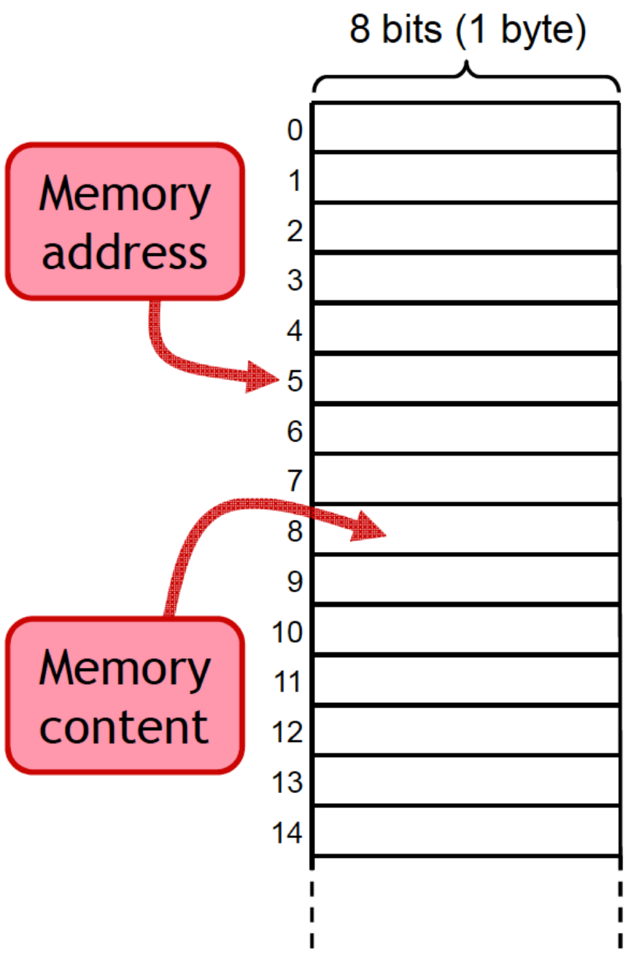
< aside >

- Quick primer on number systems!
- What is a bit? What is a byte??

Exact Range

Type	Bits	Low	High
`byte`	8	-2^{7}	$2^{7} - 1$
`short`	16	-2^{15}	$2^{15} - 1$
`int`	32	-2^{31}	$2^{31} - 1$
`long`	64	-2^{63}	$2^{63} - 1$

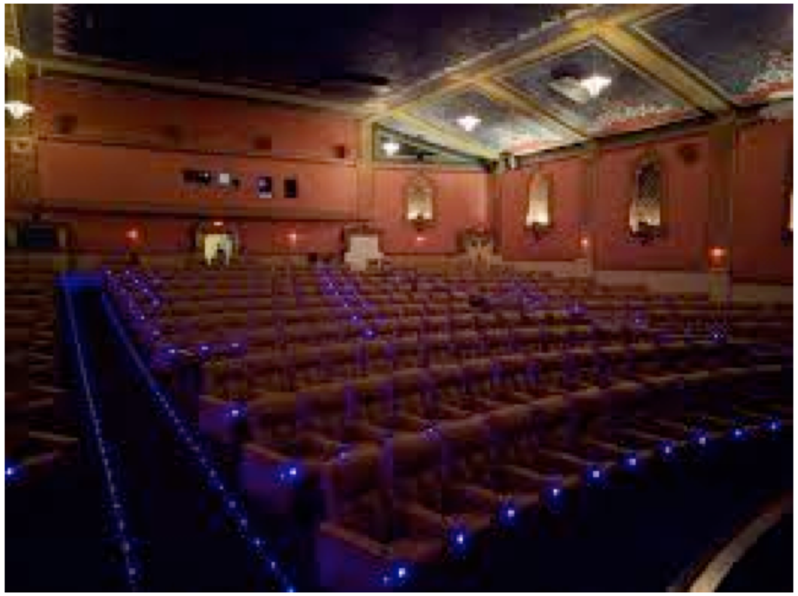
Computer Memory



- Memory is viewed as a one-dimensional arrangement of cells
- Each cell is 8 bits (*Note*: 1 byte = 8 bits)
- The total number of cells is the size of the memory
- Size is articulated in multiples of…
  - Kilobyte (1 KB = 1024 bytes)
  - Megabyte (1 MB = 1024 KB)
  - Gigabyte (1 GB = 1024 MB)
  - *Note*: $2^{10} = 1024$
- Memory addresses start at 0 and extend upward (see figure at left)

Analogy of a theatre

- Theatre: memory block (storage – X number of seats)
- Seats: memory element (individual location in theatre)
- People: values (temporarily resides in a seat)
- Tickets: variables (an identifier connecting name to seat)



Different types ?

Declaration and Memory

- With the declaration

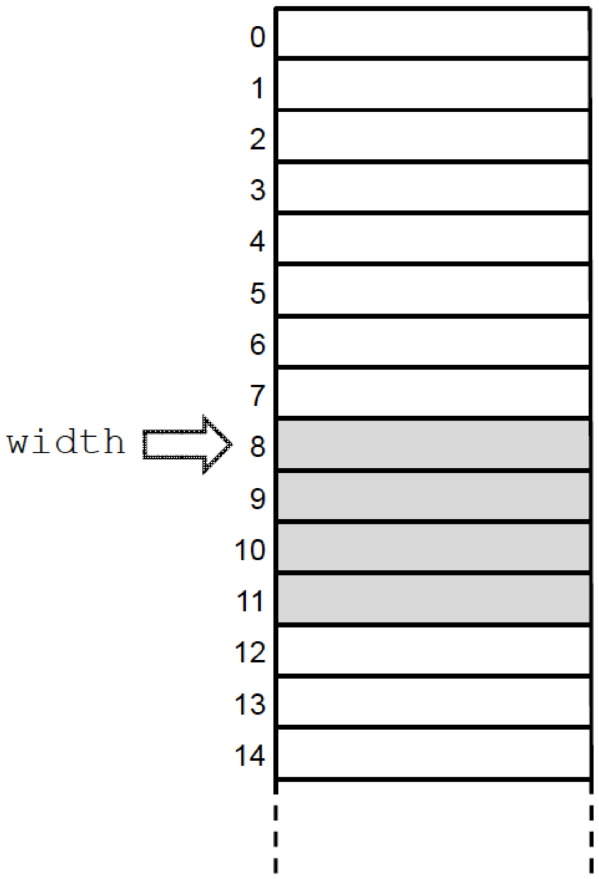
```
    int width;
```

  the compiler will set aside a 4-byte (32-bit) block of memory (see right)

- The compiler has a symbol table, which will have an entry such as

Identifier	Type	Block Address
`width`	`int`	8

- *Note*: No initialization is involved; there is only an association of a name with an address.



Reals (format, storage, range)

- Format
  - Formatted according to the IEEE-754 standard for floating point arithmetic
  - Includes a fractional part and a power (the details needn't concern us here)
- Storage
  - `float` → 4 bytes
  - `double` → 8 bytes
- Range
  - `float` → $\pm 10^{38}$ with 7 significant digits
  - `double` → $\pm 10^{308}$ with 15 significant digits

Real Examples

```
double x;

double interestRate = 1.5;

float z = -1.1f;

double abc = 3.4E-5;
```

Float literal (default is double)

Same as

`0.000034`

Special Cases

- What happens if…
  - Division by zero
    - Integers: throws an arithmetic exception
    - Reals: assigns a fictitious value, `NaN` (“not a number”)
  - Out of range result
    - Integers: range is treated as circular
    - Reals: assigns a fictitious value, `Infinity`