Lecture 2 – Primitive Types

08-671
Java Programming for App Developers

September 3, 2015

Jeffrey L. Eppinger & Terry Lee

How many are in the class?

- Still trying to get in?
  - Stop by and see me after class

Our Teaching Assistants

1. Renzo Bautisa
2. Anny Ni
3. Hao Fu
4. Akhil Prakash
5. Hao Tang
6. Yifu Wang
7. Xinyue Wu
8. Tianyue Xiao

See Blackboard/Piaza for office hours & additional contact info

08-671 Lecture Topics

(subject to change – but only a little bit)

#1 Intro
#2 Primitive Types
#3 Java Classes
#4 Reference Types
#5 Loops & Arrays
#6 Lists & Sorting
#7 Maps
#8 File & Network I/O
#9 Swing Interfaces
#10 Swing Actions
#11 Threads
#12 Exceptions
#13 Functional Programming
#14 In-class Written Exam

* Final Exam – this will be a 3-hour programming problem

Access Issues

- You should have access to:
  - Blackboard
  - Piazza
  - Panopto (video)
  - Autolab
  - If not see me after class (or e-mail) to resolve
- If can't get into the course, but want Blackboard access
  - See me after lecture

Outline

- ✓ Administrative Issues
- ⇢ Questions
- JAPL
- Readings & Homework #2

Question for You?

- What to see HW#2?

Compute the last UPC digit

- The last digit of a UPC is computed from the first 11 digits
- If the 12 digits of the UPC are:

  a b c d e f g h i j k x

- The formula to compute *x* is

  $x = (10 - (3a+b+3c+d+3e+f+3g+h+3i+j+3k) \bmod 10) \bmod 10$

- See HW#2 spec on Blackboard for details
  - It should appear at 1:30pm today

Outline

- ✓ Administrative Issues
- ✓ Questions
- ⇢ JAPL
- Readings & Homework #2

JAPL

- Just Another Programming Language
  - Data Types
  - Literals
  - Variables
  - Expressions
  - Statements
  - Methods
  - Arrays
  - Comments

Source: Apple Computer

Primitive Data Types

- Written in lower case letters
- Declares storage of one bit to eight bytes
- You can store only certain values:
  - Numbers
  - Booleans
  - Characters

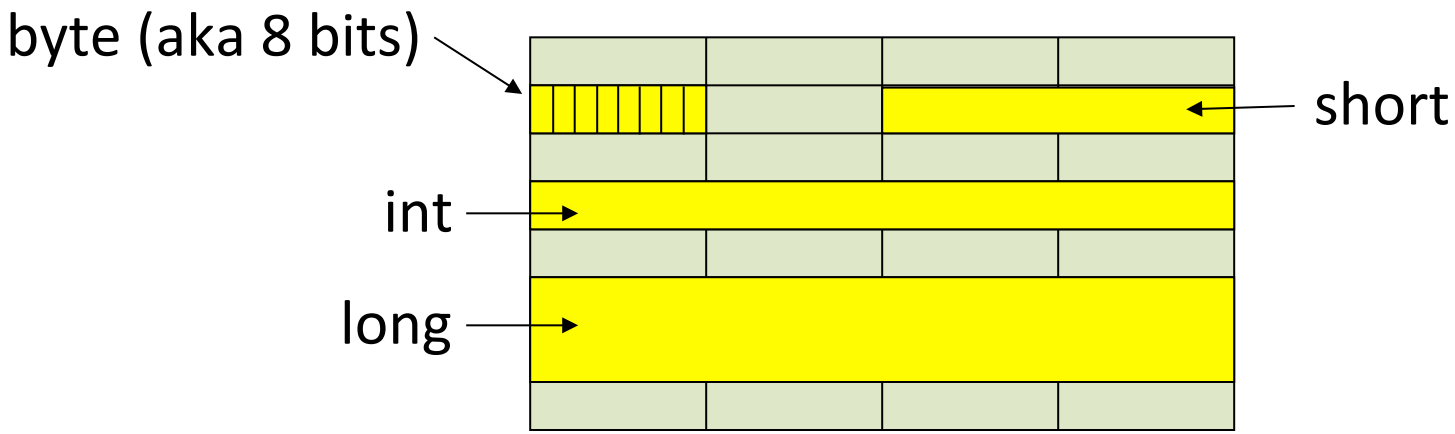
Integer Primitive Data Types

`int` – integer values (range is approx ±2 billion)

`long` – integer values (range is approx ±9 quintillion)

`byte` – integer values (range is -128 to +127)

`short` – integer values (range is approx ±32 thousand)



Binary Numbers

- Internally, modern computers use binary numbers to store their data.

Decimal	Binary	Hex
0	0	0
1	1	1
2	10	2
3	11	3
8	1000	8
10	1010	A
15	1111	F
16	10000	10
127	1111111	7F
255	11111111	FF

The Chart

Type	Min	Max	Size (in bytes)
byte	-128	127	1
short	-32,768	32,767	2
int	-2,147,483,648	2,147,483,647	4
long	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	8

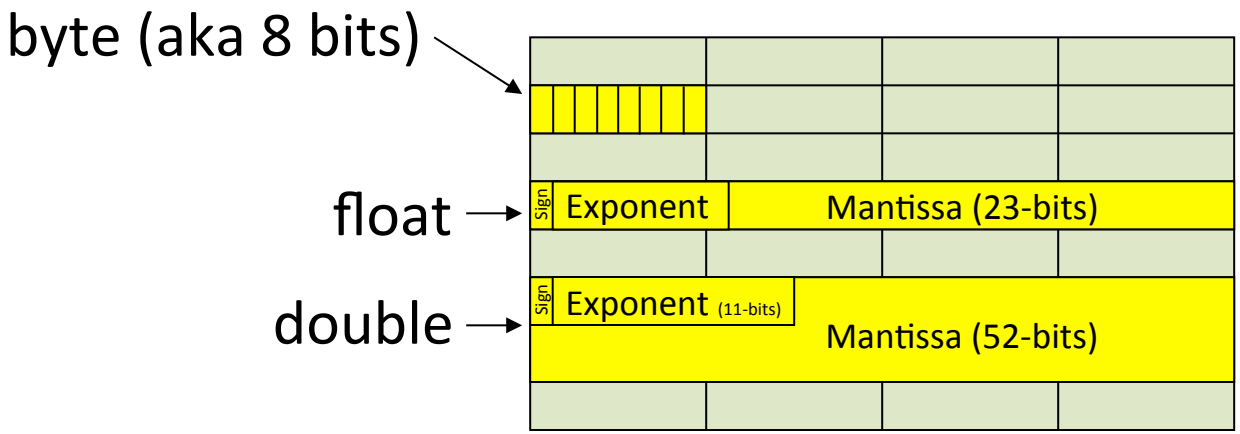
Floating Point Primitive Data Types

`float` – approximates real values

- » Range: approximately ± 3.4×10^{38}
- » Precision: only about 6-7 decimal digits

`double` – approximates real values

- » Range: approximately ± 10^{308}
- » Precision: only about 15 decimal digits



The Chart

Type	Min	Max	Size (in bytes)
byte	-128	127	1
short	-32,768	32,767	2
int	-2,147,483,648	2,147,483,647	4
long	-9,223,372,036, 854,775,808	9,223,372,036,854,775,807	8
float	$\pm 1.4 \times 10^{-45}$	$\pm 3.4028235 \times 10^{38}$	4
double	$\pm 4.9 \times 10^{-324}$	$\pm 1.7976931348623157 \times 10^{308}$	8

Just So You Know

- The Java Class Library has arbitrarily large number classes:
  - The BigInteger Class
  - The BigDecimal Class

Truth Value Primitive Data Types

- Declared using the **`boolean`** keyword
- Stores logic values of **`true`** and **`false`**

The Chart

Type	Min	Max	Size (in bytes)
byte	-128	127	1
short	-32,768	32,767	2
int	-2,147,483,648	2,147,483,647	4
long	-9,223,372,036, 854,775,808	9,223,372,036,854,775,807	8
float	$\pm 1.4 \times 10^{-45}$	$\pm 3.4028235 \times 10^{38}$	4
double	$\pm 4.9 \times 10^{-324}$	$\pm 1.7976931348623157 \times 10^{308}$	8
boolean	false	true	1 ?

Character Primitive Data Types

- **`char`** -- Stores "character" values
- Java characters are presented using 2 bytes
- Numeric range in values is 0 to 65,535
  - Unicode is mapping between the numeric values and "characters"
  - "Low" numbers use the old ASCII mapping
    - Dates back to the 1960's teletype codes
    - A to Z are 65 to 90
    - a to z are 97 to 122
    - 0 to 9 are 30 to 39
    - Characters on US Engish keyboard have codes from 32 to 127
  - Also specified Latin, Greek, Cyrillic
  - Unicode variants emerging: UTF-8 is the current popular

7-bit ASCII Characters

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Source: Alain Courteau

Literal Values

- `5`
- `142L`
- `5l`
- `25000.00`
- `25000F`
- `-458`
- `45e33`
- `45e-33D`
- `true`
- `false`

- `'a'`
- `"Hi Mom!"`
- `"H"`
- `'\n'` (`'\u000A'`)
- `'\12'` (`'\x0C'`)
- `'\u004D'` (`'M'`)
- `'\u00D6'` (`'Ö'`)
- `"\""` (not `"'"`)
- `'\''` (not `"'"`)

Variables

- Declare variables

```
String greeting;
String greeting = "Hi Mom";
int count;
int count = 0;
long soBig = 10987654321;
float salary = oldSalary * 1.03;
double natlDebt = 4000000000000;
boolean keepRunning = true;
```

Note: Strings

- Strings are not primitive values/types
- They are reference types
- Note that String starts with a capital letter
- We'll discuss details of reference types next week

Example 1

```
public class Example1 {
    public static void main(String[] args) {
        int age = 8;
        System.out.println("Hi Mom!");
        System.out.print("Look at me!  I am ");
        System.out.println(age);
    }
}
```

Constants

- Declare constants

```
final float PI = 3.14159;
final double e = 2.718281828459045;

final int LEFT   = 1;
final int RIGHT  = 2;
final int TOP    = 3;
final int BOTTOM = 4;

final String BLOOD_TYPE = "A" + "B";
```

Example 2 (won't compile)

```
public class Example2 {
    public static void main(String[] args) {
        final String BLOOD_TYPE = "A" + "B";
        BLOOD_TYPE = "O";
    }
}
```

Naming Conventions

- Classes start with a capital letter
- variables and methods start with lower case
- In multi-word class, variable, method names, the subsequent words are capitalized:

```
public class MyCoolClass {
    int complexMathFunction(int x) {
        int localVariable;
        …
```

- CONSTANTS are all capital letters
  - Underscore is used to separate words

```
final int MAX_VALUE = 2147483647;
```

Expressions

- Comparison

```
x == 34
x != y
x < y
x > y
x <= y
x >= y
```

- Math

```
x + y
x - y
x * y
x / y
x % y
x * y + z
x * ( y + z )
```

Example 3

```
public class Example3 {
    public static void main(String[] args) {

        int x = 44;
        System.out.println("x = " + x);

        int half = x / 2;
        System.out.println("half = " + half);

    }
}
```

More Expressions

- Logic (boolean)

```
x == 34 && y == 4
x == 34 || y == 4
!(x==34 || y==4)
```

- Assignment

```
x = y
x = y = 4
x += y
x -= y
x *= y
x /= y
x %= y
```

Weird Expressions

- Bit Operations

```
x & 1
x | 1
x ^ y
x << 2
x >> 10
~x
x ^= y
x >>> 1
```

- Increment Operations

```
x++
x--
```

- Arrays (more later)

```
args[0]
```

- Field access

```
jeff.lastName
```

- Class instances (more later)

```
s instanceof String
```

Numeric Gotchas

- Silent type conversion
- Silent integer overflow
  - It just wraps around
- Silent floating-point rounding

Puzzle Break

What does this do?

```
/*
 * Puzzle #3 from "Java Puzzlers" by Josh Bloch & Neal Gafter
 */

public class LongDivision {
    public static void main(String[] args) {
        final long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;
        final long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;
        System.out.println(MICROS_PER_DAY / MILLIS_PER_DAY);
    }
}
```

Input Comes as Strings

- We'll talk more about Strings next week, but …
  - You can use `args` to get command line input:

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

  - You can use the String, Integer, and Long classes
  - Let's out the JavaDoc for these classes

Example 4

```
public class Example4 {
  public static void main(String[] args) {
    System.out.println(args[0]);
  }
}
```

Example 5

```
public class Example5 {
    public static void main(String[] args) {

        int x = Integer.parseInt(args[0]);
        System.out.println("x = " + x);

        int half = x / 2;
        System.out.println("half = " + half);

    }
}
```

Example 6

```
public class Example6 {
    public static void main(String[] args) {

        long x = Long.parseLong(args[0]);
        System.out.println("x = " + x);

        long half = x / 2;
        System.out.println("half = " + half);

    }
}
```

Example 7

```
public class Example7 {
    public static void main(String[] args) {
        System.out.println("args[0]=" + args[0]);
        System.out.println("args[1]=" + args[1]);
        System.out.println("args[2]=" + args[2]);
        System.out.println("args[3]=" + args[3]);
        System.out.println("args[4]=" + args[4]);
        System.out.println("args[5]=" + args[5]);
    }
}
```

Interesting stuff re: Expressions

- Operator precedence

```
x + y < z + 4 && b-- > 0
```

- Remember: String concatenation

```
greeting + "  I've missed you!!"
```

- Difference between = and == (double equal)

```
x = y;
if (x == y) then ...
```

Statements

- This is the way to make a program
- Variable declarations

```
int i=5;
```

- Multiple statements

```
int i=5; String greeting = "Hi";
```

Interesting things about Variables

- Must be declared (unlike FORTRAN)
- Can be declared in the middle of code blocks (unlike other languages)
- Can be declared in `for` construct (unlike others)
- Declarations can set initial value to the result of an expression

Comments

- Comments in Java have three variations:
  - One line comments start with `//`
  - Multi-line comment is delineated by `/*` … `*/`
    - One line comments can be included in multi-line comments
    - Multi-line are handy for commenting out code so one line comments are used in the running code
  - JavaDoc comments delineated by `/**` … `*/`
    - JavaDoc comments can be proceed by tools
    - Eclipse has support for JavaDoc comments

Example JavaDoc Comment

```
/**
 * 08-671 Homework #2
 * @author Jeff Eppinger (je0k@andrew.cmu.edu)
 * September 3, 2015
 */
```

Program Structure

- One class per file:

```
public class MySample { ... }
```

- File must have the same name as the class:

```
MySample.java
```

- Main method must be declared as follows:

```
public static void main(String[] args) {
    ...
}
```

HelloWorld (from Lecture 1)

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Editing Alternatives

- ✓ Notepad
- ✓ Wordpad
- ✓ TextEdit
- ⇢ Sublime Text 2 ($70, but has unlimited trial)
- Eclipse (Maybe next week)
- (You can use anything you like)

Coding Conventions

- Links to Java Coding Conventions are posted on the Blackboard
- For now be sure to:
  - Put a comment at the top of each file with your name, course number, and date
  - Indent (4 spaces -- as shown my examples)
  - Follow brace conventions (as in my examples)
  - Follow the capitalization conventions

Outline

- ✓ Administrative Issues
- ✓ Questions
- ✓ JAPL
- ⇢ Readings & Homework #2

Homework & Readings

- HW#2 will be posted on the Blackboard
  - In mere moments
- Due Mon, 9/7 (!)
- Readings (covering Basic Programming)
  - Head First Java through Chapter 5
  - Java in 21 Days: through Day 5
- You can do the entire homework using only what was covered in this lecture

Recitation Tomorrow

- In DH A302 @ 1:30pm
- Discuss HW#2 Strategies
  - We will ask some of you to show us your progress
    - Volunteers receive prizes potentially worth MILLIONS

Sample Exam Question

- What's the difference between a primitive type and a reference type?