Primitive Types

Characters
(char)

Lecture Contents

- Review of ASCII
- Unicode and Unicode Transformation Format (UTF)
- Java `char` type

List of Java Primitive Types

- *Integers*
  - `byte` (8 bits)
  - `short` (16 bits)
  - **`int`** (32 bits)
  - `long` (64 bits)
- *Real Numbers*
  - `float` (32 bits)
  - **`double`** (64 bits)
- *True/False*
  - **`boolean`** (1 bit?)
- *Letters*
  - `char` (16 bits)

Note: Primitive types tested in *AP Computer Science A* are given in bold font.

Review of ASCII

- The table shows the ASCII ***character code*** for printable characters.
- ASCII = American Standard Code for Information Interchange

ASCII Character Set (0x20-0x7F)

hex	char	hex	char	hex	char	hex	char	hex	char	hex	char
20	*space*	30	0	40	@	50	P	60	`	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	“	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	'	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	*delete*

Unicode

- Unicode assigns numbers to characters (also known as "*code points*")
  - Code points up to 127 (0x7F) are exactly the same as the original ASCII characters
  - Other code points contain other language characters, emoji's, etc.
  - Unicode is able to assign 1,114,112 code points
  - fewer than 160,000 code points have been assigned (15%)

```
0x41:     A
0x597D:   好
0x1F60E:  😎
```

Unicode

- Unicode Transformation Format
  - Encoding of Unicode values for storage and transmission
- UTF-8
  - 8 bits (one byte) for ASCII characters (good for English / Latin text)
  - Most common encoding for the World-Wide Web
  - Characters beyond ASCII require more than one byte
    - Chinese characters, for example, require 3 bytes
- UTF-16
  - Most language characters (including Chinese) require two bytes
  - Each emoji and other symbols require four bytes

Java `char` type

- Java stores characters as UTF-16.
  - Unicode code points from `0x0000` to `0xFFFF` are stored in two bytes
    - Includes the ACII characters in the range `0x0000 – 0x007F`
    - Unicode Basic Multilingual Plane (BMP) code points `0x0000 – 0xFFFF`
    - Most Chinese Characters `0x4E00 – 0x9FFF`
  - Special characters, such as emojis and rare Chinese characters require four bytes.

`0x41:` A

`0x597D:` 好

`0x1F60E:` 😎

Java `char` type

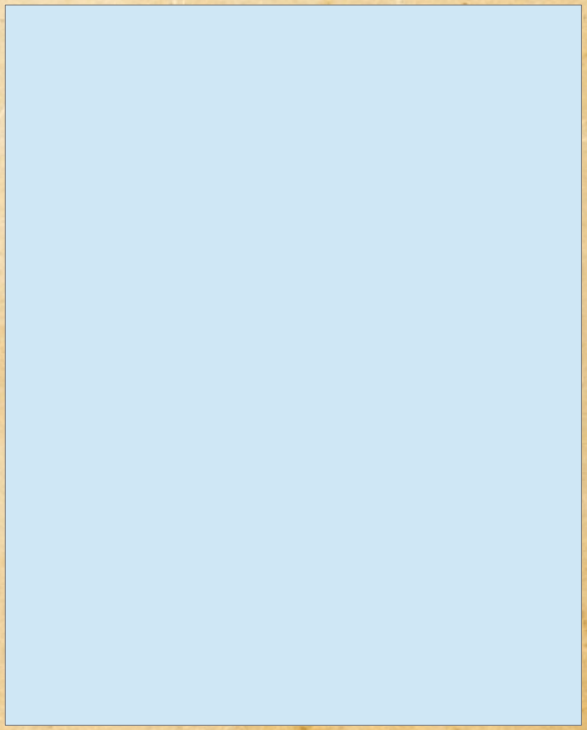
- Java characters are stored using the type `char`.

- Character literals are enclosed in single quotation marks:
  - `System.out.println('A');` ✔

- Only one character can be placed in the single quotation marks:
  - `System.out.println('AB');` ✗ ← **Invalid character constant**

Using the Java `char` type

```
System.out.println('A');

System.out.println('A' + 0);

System.out.println('a');

System.out.println('a' + 0);

System.out.println('0');

System.out.println('0' + 0);
```



Using the Java `char` type

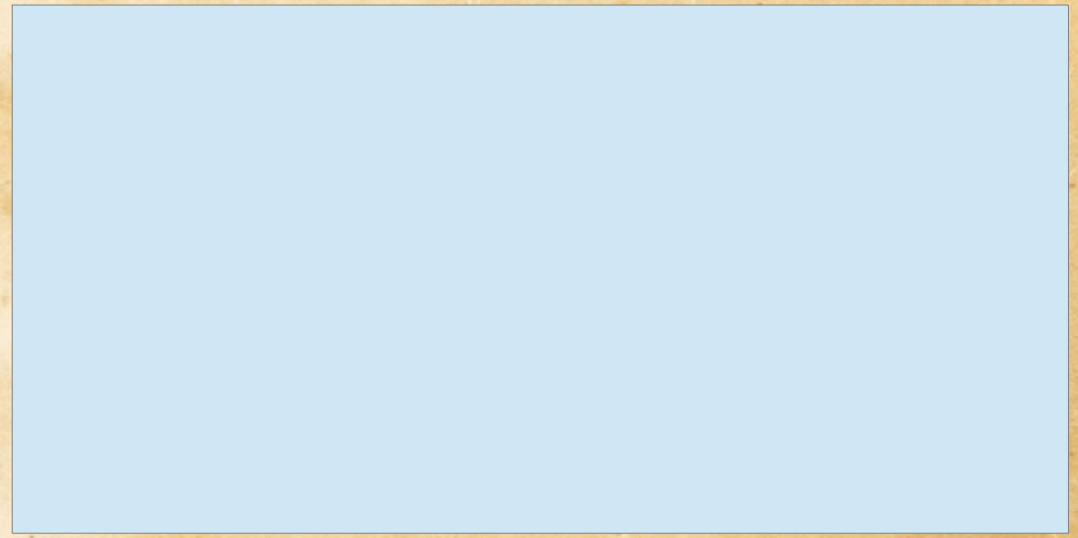
```
System.out.println('A');
System.out.println('A' + 0);
System.out.println('a');
System.out.println('a' + 0);
System.out.println('0');
System.out.println('0' + 0);
```

```
A
65   ← 0x41
a
97   ← 0x61
0
48   ← 0x30
```

Arithmetic operations with `char` and `int` operands result in an `int`

Using the Java `char` type

```
System.out.println('A');

System.out.println('A' + 0);

System.out.println('A' / 2);

System.out.println('A' + 'A');
```



Using the Java `char` type

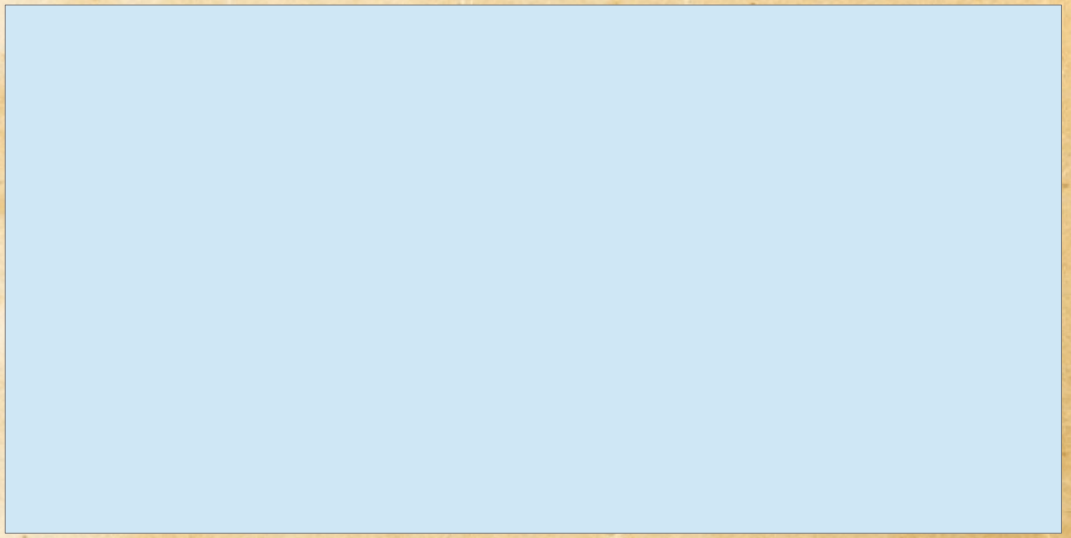
```
System.out.println('A');
System.out.println('A' + 0);
System.out.println('A' / 2);
System.out.println('A' + 'A');
```

```
A
65    ← 0x41
32    ← 0x20 (integer division)
130   ← char + char results in an int
```

Arithmetic operations with `char` and `char` operands result in an `int`

Using the Java `char` type

```
System.out.println('A');

System.out.println('A' + 0);

System.out.println('爱');

System.out.println('爱' + 0);
```



For ASCII characters, the upper byte is all zeros. For other language characters, the upper byte is non-zero.

Using the Java `char` type

```
System.out.println('A');

System.out.println('A' + 0);

System.out.println('爱');

System.out.println('爱' + 0);
```

```
A

65      ← 0x0041 (upper byte all zeros)

爱

29233   ← 0x7231 (uses upper byte)
```

For ASCII characters, the upper byte is all zeros. For other language characters, the upper byte is non-zero.

Using the Java `char` type

- Each emoji character (😀, 😝, 🤓, etc.) requires more than two bytes, so cannot be stored in the Java `char` type.

  ✗ `System.out.println('😭');` ← **Invalid character constant**

- You can still use them in Strings:

  ✔ `System.out.println("😭");` ← **Works fine in a Java `String`!**

- Perhaps we may examine this with more detail when we learn about the Java `String` type.

Primitive Types

Characters
(char)