

# CSE1710

Week 12, Lecture 22

Click to edit this text

Second level

Third level

Fifth level

Fall 2013 ♦ Tuesday, Nov 26, 2013



## Big Picture

The next **three** class meetings will focus on Chapter 6 concepts.

There will be a labtest on Chapter 5 concepts on **Thurs Nov 28/Fri Nov 29**.

### NOTE1:

- Tuesday Dec 03 is designated as a study day. No classes.

### NOTE2:

- The course syllabus indicates that there is 5% for "In-class quizzes & eChecks"
- After the labtest this week, there will be a set of exercises distributed on Friday Nov 29<sup>th</sup> (written answers and eCheck).
- The exercises will be due Friday Dec 6<sup>th</sup> (on-line submission).



## Recap: the char type

- `char` is one of the 8 primitive types in Java
- a `char` value represents a single 16-bit **Unicode** character.
  - $2^{16} = 65,536$  unique representations
  - Minimum value is 0 (or `'\u0000'`)
  - Maximum value is 65,535 (or `'\uffff'`)
- `unicode` makes it possible to talk about the *distance* between two characters

3



## Recap: About Unicode

Unicode is a computing industry standard

- provides consistent encoding, representation and handling of text
- covers most of the world's writing systems
- used by Java and many other programming languages
- defines a series of 17 planes
  - we will use the basic plane; it contains the most frequently used characters
- The intent behind unicode is to abstract away the **graphemes** (underlying characters) from their **glyphs**

grapheme: the letter 'e'

glyphs: e  e e e

4



## Unicode

- The Unicode Standard consists of a repertoire of more than 109,000 characters covering 93 scripts
  - Cyrillic, Latin, Bengali, Thai, Greek, ...
  - the basic set is “Controls and Basic Latin”
  - U000.pdf, also see Appendix A of JBA
- Unicode value denoted by \uXXXX, where XXXX is a hexadecimal value
  - the decimal value 15 is represented as \u000F

5

The character **J** is found in:

- column ‘004’
- row ‘A’,

together: ‘004A’

This hexadecimal number is denoted \u004A

6

<http://unicode.org/charts/PDF/U0000.pdf>

	000	001	002	003	004	005	006	007
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	JACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	<b>J</b>	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

6

## To convert a Unicode hexadecimal number to decimal:

1. take the hex number and identify the four digits:

`\u004A` →  $d_3d_2d_1d_0$  → 0 0 4 A

2. Convert each hex digit to decimal:

- the hex  $d_i$  span the digits: [0, ..., 9, A, B, C, D, E, F]
- this maps to the decimal digits: [0, 15]
- hex 'A' maps to decimal '10', ..., 'F' maps to '15'

3. Plug the digits into the following formula:

$$d_3d_2d_1d_0 = d_3 \times 16^3 + d_2 \times 16^2 + d_1 \times 16^1 + d_0 \times 16^0$$

Example: so to convert `\u004A` to decimal:

$$= 0 \times 16^3 + 0 \times 16^2 + 4 \times 16^1 + 10 \times 16^0$$

$$= 4 \times 16 + 10 \times 1$$

$$= 64 + 10 = 74$$

7

7



## A Caveat:

some Unicode characters cannot be *printed* to the console

some graphemes don't have corresponding glyphs

For example,

consider `\u10EC` is taken from the "Georgian" table

the grapheme represents the character:



*We can represent the character, but PrintStream cannot print it to the console...*

```
char c1 = '\u10EC';
```

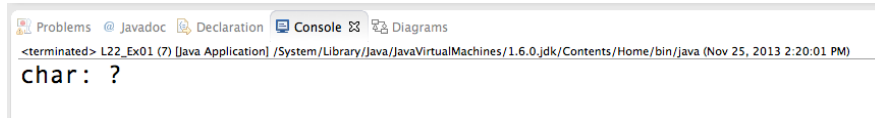
8



# Example

```
1 package lecture22;  
2  
3 import java.io.PrintStream;  
4  
5 public class L22_Ex01 {  
6  
7     public static void main(String[] args) {  
8         PrintStream output = System.out;  
9         char a = '\u10EC';  
10        output.printf("char: %s\n", a);  
11    }  
12 }
```

## OUTPUT



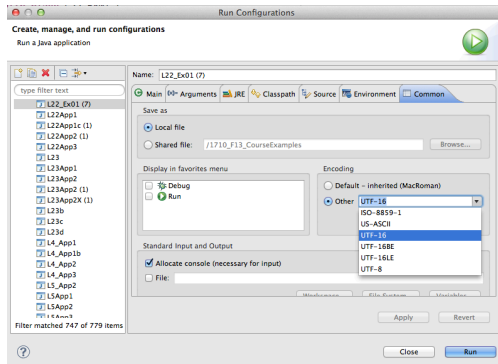
9



# A possible solution

Under Run->Run Configurations

Modify the output character encoding that is used in Eclipse to "UTF-16"



10



## Example

```
1 package lecture22;  
2  
3 import java.io.PrintStream;  
4  
5 public class L22_Ex01 {  
6  
7     public static void main(String[] args) {  
8         PrintStream output = System.out;  
9         char a = '\u10EC';  
10        output.printf("char: %s\n", a);  
11    }  
12 }
```

### OUTPUT



The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

11



## Moving from char to String objects

- a char value represents a **single character**
- to represent **a sequence of characters**, we use the services of the String class

12



## What is the String class?

Provides services:

- **to represent** strings objects
- to obtain information about the string, **to query** the string
- **to create new, modified** strings

*This is a very, very important characteristic of the String class*

**DOES NOT** provide services:

- to mutate or modify the object's state

13



## What is the state of a String object?

String objects has **two** attributes:

- a sequence of chars
- a comparator

*RECALL: The state of an object is given by the values of all of its attributes*

- We are not going to spend **any** further time on this.
- A string comparator is a service that can take any two strings and determine their relative order.
- We will stick with **the default comparator**, which uses lexicographic ordering
- **lexicographic** is a fancy way of saying "dictionary" or "alphabetic" ordering

14



## How do we create a String object?

Either the **typical way** or the **short-cut way**:

### Typical:

```
String s1 = new String("hello");
```

### Short-cut:

```
String s2 = "hello";
```

15



## Predict the outcome

```
String s1 = "hello1";  
String s2 = "hello2";  
char c1 = "h";  
char c2 = 'h';  
String s3 = 'h';  
char c3 = '\u0045';  
char c4 = "\u0045";  
String s4 = "\u0048\u0065\u006C\u006C\u006F";  
  
char c5 = 'p';  
c5 = "h";
```

16





## Take Home Message

You cannot **interchange** char and String values

They are different!

17



### Details about the String object's attribute: the character sequence...

- the sequence is indexed
  - the **first** position is index "0"
  - the **final** position is index "the length of the sequence minus 1"
- The String object has services so we can ask about its **character sequence**
- Methods include:
  - int : length()
  - char : charAt(int)

18

- what if index is out of bounds?
- can the length ever be smaller than 0?

18



## Predict the outcome

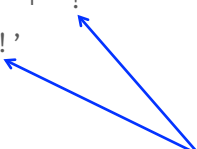
```
String s1 = "hello1";  
String s2 = "\u0048\u0065\u006C\u006C\u006F";  
String s3 = new String("");  
String s4 = "";  
String s5 = " ";
```

```
s1.length();      s1.charAt(0);  
s2.length();      s2.charAt(2);  
s3.length();      s3.charAt(0);  
s4.length();      s4.charAt(0);  
s5.length();      s5.charAt(1);
```

19



## String masquerades as a primitive...

- We've seen this in the "short-cut" instantiation
- Here's another way: the + operator
  - it is possible to use the + operator between two String operands
  - "hello " + "there"
  - "\u0048\u0065\u006C" + "\u006C\u006F"
  - "hello" + '!'
  - "" + '!' 

*The char operand gets promoted to a string!*

20



## The + Operator : Predict the Output

```
String x = "hi\n";  
String y = "there";  
String z = x + y;  
output.println(z);  
  
char a = 'H';  
char b = 'I';  
output.println(a+b);
```

21



## What is the “empty” string?

- If the state of a String object is such that its sequence has no characters at all, how do we understand this?
  - this is the empty string
  - the string has length zero
- **THIS IS NOT A NULL STRING**

22



## What is the “null string”?

- technically speaking, “null string” is not a correctly-formed term, there is **no such thing**
- HOWEVER, it is often used to mean a string reference that is set to null.
- This means that a String reference has been declared, but that there is NO String object
- the object reference’s value is null

23



### REMEMBER!

- Any string is represented by **an object**
- A variable of type String is an **object reference**:
  - it is used to store **the address** of a String object.
- The String object has a **state**
  - the **state** of an object is defined as **the value of all its attributes**
  - the **only\*** attribute of a String object is the attribute that represents the sequence of characters
  - the state of a String object basically boils down to **what is its sequence of characters?**

24

24



## String object vs char value

	<u>String</u>	<u>Character</u>	
type:	String <i>non-primitive</i>	char <i>primitive</i>	
operators:	+	+ - / * %	(arithmetic operators) <i>cast to int</i>

25



### Can we modify the state of a String object?

- NO
- Once a string object is created, it cannot be changed.
  - This is called *immutability*
  - Strings are *immutable*
- This is an unusual property – MOST other objects are mutable
- Given this, is it correct to say that String has mutators?
  - not technically; they are actually *generators of new modified objects*

26

26



## But what if we need to modify the state of a String object?

Instead of modifying the sequence, we can

1. create new strings that are modified versions of the originals
  - It is fast and easy, thanks to the + operator
  - there are other methods (substring, etc) – we'll cover next lecture
2. use the services of StringBuffer – we'll cover next lecture

*Next lecture:* regular expressions