CSE 101 - COMPUTER PROGRAMMING I MATHEMATICAL FUNCTIONS, CHARACTERS, AND STRINGS

Joseph LEDET
Department of Computer Engineering
Akdeniz University
josephledet@akdeniz.edu.tr

MATHEMATICAL FUNCTIONS

Java provides many useful methods in the **Math** class for performing common mathematical functions.

2

THE MATH CLASS

- Class constants:
  - PI
  - E

- Class methods:
  - Trigonometric Methods
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods

3

TRIGONOMETRIC METHODS

- **sin(double a)**

- **cos(double a)**

- **tan(double a)**

- **acos(double a)**

- **asin(double a)**

- **atan(double a)**

  Radians

  toRadians(90)

```
Examples:

Math.sin(0) returns 0.0
Math.sin(Math.PI / 6)
  returns 0.5
Math.sin(Math.PI / 2)
  returns 1.0
Math.cos(0) returns 1.0
Math.cos(Math.PI / 6)
  returns 0.866
Math.cos(Math.PI / 2)
  returns 0
```

4

EXPONENT METHODS

- **`exp(double a)`**
  Returns e raised to the power of a.

- **`log(double a)`**
  Returns the natural logarithm of a.

- **`log10(double a)`**
  Returns the 10-based logarithm of a.

- **`pow(double a, double b)`**
  Returns a raised to the power of b.

- **`sqrt(double a)`**
  Returns the square root of a.

```
Examples:

Math.exp(1) returns 2.71
Math.log(2.71) returns 1.0
Math.pow(2, 3) returns 8.0
Math.pow(3, 2) returns 9.0
Math.pow(3.5, 2.5) returns
    22.91765
Math.sqrt(4) returns 2.0
Math.sqrt(10.5) returns 3.24
```

ROUNDING METHODS

- **`double ceil(double x)`**
  x rounded up to its nearest integer. This integer is returned as a double value.

- **`double floor(double x)`**
  x is rounded down to its nearest integer. This integer is returned as a double value.

- **`double rint(double x)`**
  x is rounded to its nearest integer. If x is equally close to two integers, the even one is returned as a double.

- **`int round(float x)`**
  Return (int)Math.floor(x+0.5).

- **`long round(double x)`**
  Return (long)Math.floor(x+0.5).

ROUNDING METHODS EXAMPLES

```
Math.ceil(2.1) returns 3.0          Math.rint(2.0) returns 2.0

Math.ceil(2.0) returns 2.0          Math.rint(-2.0) returns -2.0

Math.ceil(-2.0) returns -2.0        Math.rint(-2.1) returns -2.0

Math.ceil(-2.1) returns -2.0        Math.rint(2.5) returns 2.0

Math.floor(2.1) returns 2.0         Math.rint(-2.5) returns -2.0

Math.floor(2.0) returns 2.0         Math.round(2.6f) returns 3

Math.floor(-2.0) returns -2.0       Math.round(2.0) returns 2

Math.floor(-2.1) returns -3.0       Math.round(-2.0f) returns -2

Math.rint(2.1) returns 2.0          Math.round(-2.6) returns -3
```

7

MIN, MAX, AND ABS

- `max(a, b)` and `min(a, b)`
  Returns the maximum or minimum of two parameters.

- `abs(a)`
  Returns the absolute value of the parameter.

- `random()`
  Returns a random `double` value in the range [0.0, 1.0).

```
Examples:

Math.max(2, 3) returns 3
Math.max(2.5, 3) returns
  3.0
Math.min(2.5, 3.6)
  returns 2.5
Math.abs(-2) returns 2
Math.abs(-2.1) returns
  2.1
```

8

THE <u>RANDOM</u> METHOD

Generates a random <u>double</u> value greater than or equal to 0.0 and less than 1.0 (<u>0 <= Math.random() < 1.0</u>).

Examples:

```
(int)(Math.random() * 10)
```
→ Returns a random integer between 0 and 9.

```
50 + (int)(Math.random() * 50)
```
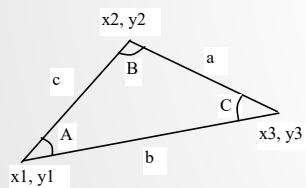→ Returns a random integer between 50 and 99.

In general,

```
a + Math.random() * b
```
→ Returns a random number between a and a + b, excluding a + b.

9

CASE STUDY: COMPUTING ANGLES OF A TRIANGLE


```
A = acos((a * a - b * b - c * c) / (-2 * b * c))
B = acos((b * b - a * a - c * c) / (-2 * a * c))
C = acos((c * c - b * b - a * a) / (-2 * a * b))
```

Write a program that prompts the user to enter the x- and y-coordinates of the three corner points in a triangle and then displays the triangle's angles.

IMPORTANT NOTE: If you cannot run the buttons, see www.cs.armstrong.edu/liang/javaslidenote.doc

ComputeAngles   Run

10

CHARACTER DATA TYPE

char letter = 'A'; (ASCII)

char numChar = '4'; (ASCII)

Four hexadecimal digits.

char letter = '\u0041'; (Unicode)

char numChar = '\u0034'; (Unicode)

NOTE: The increment and decrement operators can also be used on <u>char</u> variables to get the next or preceding Unicode character. For example, the following statements display character <u>b</u>.

```
char ch = 'a';
System.out.println(++ch);
```

11

UNICODE FORMAT

Java characters use *Unicode*, a 16-bit encoding scheme established by the Unicode Consortium to support the interchange, processing, and display of written texts in the world's diverse languages. Unicode takes two bytes, preceded by \u, expressed in four hexadecimal numbers that run from '\u0000' to '\uFFFF'. So, Unicode can represent 65535 + 1 characters.

Unicode \u03b1 \u03b2 \u03b3 for three Greek letters

Display Greek Letters

α β γ

OK
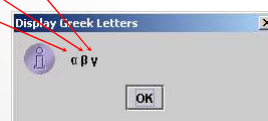
12

ASCII CODE FOR COMMONLY USED CHARACTERS

Characters	Code Value in Decimal	Unicode Value
'0' to '9'	48 to 57	\u0030 to \u0039
'A' to 'Z'	65 to 90	\u0041 to \u005A
'a' to 'z'	97 to 122	\u0061 to \u007A

13

ESCAPE SEQUENCES FOR SPECIAL CHARACTERS

Escape Sequence	Name	Unicode Code	Decimal Value
\b	Backspace	\u0008	8
\t	Tab	\u0009	9
\n	Linefeed	\u000A	10
\f	Formfeed	\u000C	12
\r	Carriage Return	\u000D	13
\\	Backslash	\u005C	92
\"	Double Quote	\u0022	34

14

APPENDIX B: ASCII CHARACTER SET

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.1 ASCII Character Set in the Decimal Index

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	}	|	}	~	del		

15

ASCII CHARACTER SET, CONT.

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.2 ASCII Character Set in the Hexadecimal Index

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	ff	cr	so	si
1	dle	dcl	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
2	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	del

16

CASTING BETWEEN CHAR AND NUMERIC TYPES

```
int i = 'a'; // Same as int i = (int)'a';


char c = 97; // Same as char c = (char)97;
```

17

COMPARING AND TESTING CHARACTERS

if (ch >= **'A'** && ch <= **'Z'**)
  System.out.println(ch + **" is an uppercase letter"**);
else if (ch >= **'a'** && ch <= **'z'**)
  System.out.println(ch + **" is a lowercase letter"**);
else if (ch >= **'0'** && ch <= **'9'**)
  System.out.println(ch + **" is a numeric character"**);

18

METHODS IN THE CHARACTER CLASS

Method	Description
isDigit(ch)	Returns true if the specified character is a digit.
isLetter(ch)	Returns true if the specified character is a letter.
isLetterOrDigit(ch)	Returns true if the specified character is a letter or digit.
isLowerCase(ch)	Returns true if the specified character is a lowercase letter.
isUpperCase(ch)	Returns true if the specified character is an uppercase letter.
toLowerCase(ch)	Returns the lowercase of the specified character.
toUpperCase(ch)	Returns the uppercase of the specified character.

19

THE STRING TYPE

The char type only represents one character. To represent a string of characters, use the data type called String. For example,

String message = "Welcome to Java";

String is actually a predefined class in the Java library just like the System class and Scanner class. The String type is not a primitive type. It is known as a *reference type*. Any Java class can be used as a reference type for a variable. Reference data types will be thoroughly discussed in Chapter 9, "Objects and Classes." For the time being, you just need to know how to declare a String variable, how to assign a string to the variable, how to concatenate strings, and to perform simple operations for strings.

20

SIMPLE METHODS FOR **STRING** OBJECTS

Method	Description
length()	Returns the number of characters in this string.
charAt(index)	Returns the character at the specified index from this string.
concat(s1)	Returns a new string that concatenates this string with string s1.
toUpperCase()	Returns a new string with all letters in uppercase.
toLowerCase()	Returns a new string with all letters in lowercase.
trim()	Returns a new string with whitespace characters trimmed on both sides.

21

SIMPLE METHODS FOR **STRING** OBJECTS

Strings are objects in Java. The methods in the preceding table can only be invoked from a specific string instance. For this reason, these methods are called *instance methods*. A non-instance method is called a *static method*. A static method can be invoked without using an object. All the methods defined in the **Math** class are static methods. They are not tied to a specific object instance. The syntax to invoke an instance method is
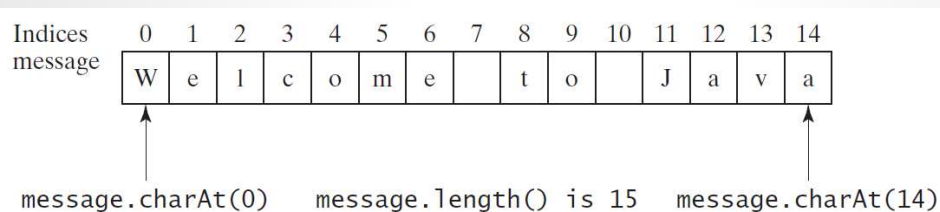
referenceVariable.methodName(arguments).

22

GETTING STRING LENGTH

String message = **"Welcome to Java"**;

System.out.println(**"The length of "** + message + **" is "**
  + message.length());

23

GETTING CHARACTERS FROM A STRING

Indices	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
message	W	e	l	c	o	m	e		t	o		J	a	v	a

message.charAt(0)   message.length() is 15   message.charAt(14)

String message = **"Welcome to Java"**;

System.out.println(**"The first character in message is "**
    + message.charAt(0));

24

CONVERTING STRINGS

"Welcome".toLowerCase() returns a new string, welcome.

"Welcome".toUpperCase() returns a new string, WELCOME.

" Welcome ".trim() returns a new string, Welcome.

25

STRING CONCATENATION

String s3 = s1.concat(s2); or String s3 = s1 + s2;

// Three strings are concatenated
String message = "Welcome " + "to " + "Java";

// String Chapter is concatenated with number 2
String s = "Chapter" + 2; // s becomes Chapter2

// String Supplement is concatenated with character B
String s1 = "Supplement" + 'B'; // s1 becomes SupplementB

26

READING A STRING FROM THE CONSOLE

```
Scanner input = new Scanner(System.in);

System.out.print("Enter three words separated by
spaces: ");

String s1 = input.next();

String s2 = input.next();

String s3 = input.next();

System.out.println("s1 is " + s1);

System.out.println("s2 is " + s2);

System.out.println("s3 is " + s3);
```

27

READING A CHARACTER FROM THE CONSOLE

```
Scanner input = new Scanner(System.in);

System.out.print("Enter a character: ");

String s = input.nextLine();

char ch = s.charAt(0);

System.out.println("The character entered is " +
ch);
```

28

COMPARING STRINGS

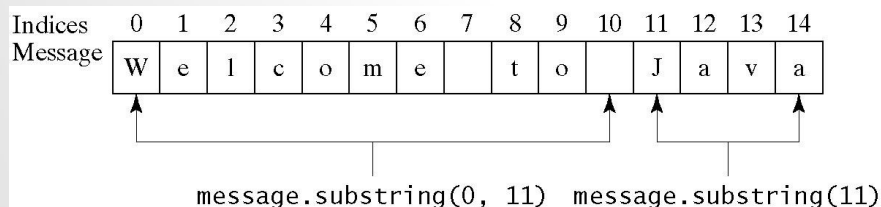| Method	Description
equals(s1)	Returns true if this string is equal to string s1.
equalsIgnoreCase(s1)	Returns true if this string is equal to string s1; it is case insensitive.
compareTo(s1)	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater than, equal to, or less than s1.
compareToIgnoreCase(s1)	Same as compareTo except that the comparison is case insensitive.
startsWith(prefix)	Returns true if this string starts with the specified prefix.
endsWith(suffix)	Returns true if this string ends with the specified suffix.

OrderTwoCities   Run

29

OBTAINING SUBSTRINGS

Method	Description
substring(beginIndex)	Returns this string's substring that begins with the character at the specified beginIndex and extends to the end of the string, as shown in Figure 4.2.
substring(beginIndex, endIndex)	Returns this string's substring that begins at the specified beginIndex and extends to the character at index endIndex - 1, as shown in Figure 9.6. Note that the character at endIndex is not part of the substring.

Indices	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Message	W	e	l	c	o	m	e		t	o		J	a	v	a

message.substring(0, 11)   message.substring(11)

30

# FINDING A CHARACTER OR A SUBSTRING IN A STRING

Method	Description
indexOf(ch)	Returns the index of the first occurrence of ch in the string. Returns −1 if not matched.
indexOf(ch, fromIndex)	Returns the index of the first occurrence of ch after fromIndex in the string. Returns −1 if not matched.
indexOf(s)	Returns the index of the first occurrence of string s in this string. Returns −1 if not matched.
indexOf(s, fromIndex)	Returns the index of the first occurrence of string s in this string after fromIndex. Returns −1 if not matched.
lastIndexOf(ch)	Returns the index of the last occurrence of ch in the string. Returns −1 if not matched.
lastIndexOf(ch, fromIndex)	Returns the index of the last occurrence of ch before fromIndex in this string. Returns −1 if not matched.
lastIndexOf(s)	Returns the index of the last occurrence of string s. Returns −1 if not matched.
lastIndexOf(s, fromIndex)	Returns the index of the last occurrence of string s before fromIndex. Returns −1 if not matched.

31

FINDING A CHARACTER OR A SUBSTRING IN A STRING

int k = s.indexOf(' ');
String firstName = s.substring(0, k);
String lastName = s.substring(k + 1);
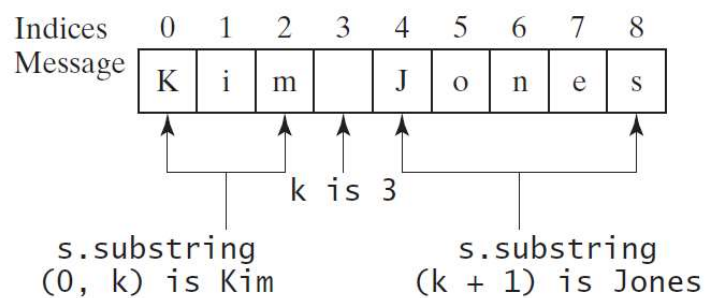
32

CONVERSION BETWEEN STRINGS AND NUMBERS

int intValue = Integer.parseInt(intString);
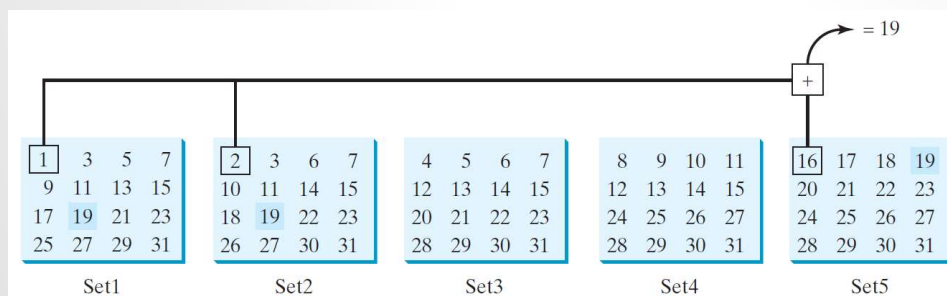**double** doubleValue = Double.parseDouble(doubleString);

String s = number + **""**;

33

PROBLEM: GUESSING BIRTHDAY

The program can guess your birth date. Run to see how it works.

| Set1 | Set2 | Set3 | Set4 | Set5 |

GuessBirthday   Run

34

MATHEMATICS BASIS FOR THE GAME

19 is 10011 in binary. 7 is 111 in binary. 23 is 11101 in binary

```
  10000
     10
+     1
  10011
```

```
  00110
     10
+     1
  00111
```

```
  10000
   1000
    100
+     1
  11101
```

19   7   23

Decimal	Binary
1	00001
2	00010
3	00011
...	
19	10011
...	
31	11111

b_5 0 0 0 0   10000
  b_4 0 0 0   1000
    b_3 0 0   100
      b_2 0   10
+ b_1 1
  $b_5\, b_4\, b_3\, b_2\, b_1$

```
  10000
     10
+     1
  10011
    19
```

```
  10000
   1000
    100
     10
+     1
  11111
    31
```

35

CASE STUDY: CONVERTING A HEXADECIMAL DIGIT TO A DECIMAL VALUE

Write a program that converts a hexadecimal digit into a decimal value.

HexDigit2Dec   Run

36

CASE STUDY: REVISING THE LOTTERY PROGRAM USING STRINGS

A problem can be solved using many different approaches. This section rewrites the lottery program in Listing 3.7 using strings. Using strings simplifies this program.

LotteryUsingStrings   Run

37

FORMATTING OUTPUT

Use the printf statement.

System.out.printf(format, items);

Where format is a string that may consist of substrings and format specifiers. A format specifier specifies how an item should be displayed. An item may be a numeric value, character, boolean value, or a string. Each specifier begins with a percent sign.

38

FREQUENTLY-USED SPECIFIERS

Specifier	Output	Example
%b	a boolean value	true or false
%c	a character	'a'
%d	a decimal integer	200
%f	a floating-point number	45.460000
%e	a number in standard scientific notation	4.556000e+01
%s	a string	"Java is cool"

```
int count = 5;
double amount = 45.56;                                          items
System.out.printf("count is %d and amount is %f", count, amount);


display         count is 5 and amount is 45.560000
```

39

FORMATDEMO

The example gives a program that uses **printf** to display a table.

FormatDemo   Run

40