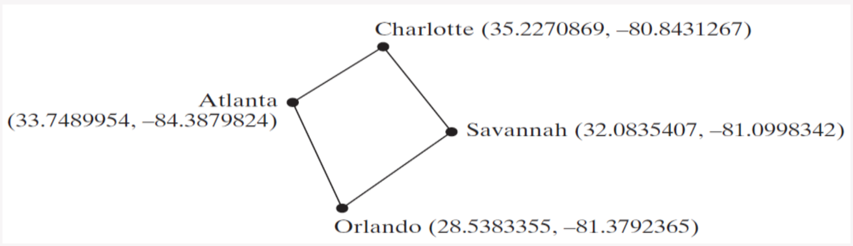
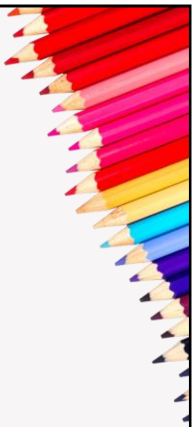
Chapter 4
Mathematical Functions, Characters, and Strings

Motivations

- *The focus of this chapter is to introduce mathematical functions, characters, string objects, and use them to develop programs.*
- Suppose you need to estimate the area enclosed by four cities, given the GPS locations (latitude and longitude) of these cities, as shown in the following diagram. How would you write a program to solve this problem? You will be able to write such a program after completing this chapter.

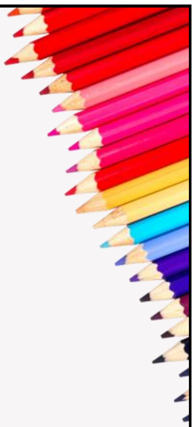

Charlotte (35.2270869, –80.8431267)
Atlanta (33.7489954, –84.3879824)
Savannah (32.0835407, –81.0998342)
Orlando (28.5383355, –81.3792365)

2



Objectives (1)

- To solve mathematics problems by using the methods in the ***Math class*** (§4.2).
- To represent characters using the ***char*** type (§4.3).
- To encode characters using ASCII and *Unicode* (§4.3.1).
- To represent special characters using the escape sequences (§4.3.2).
- To cast a numeric value to a character and cast a character to an integer (§4.3.3).
- To compare and test characters using the static methods in the ***Character class*** (§4.3.4).



Objectives (2)

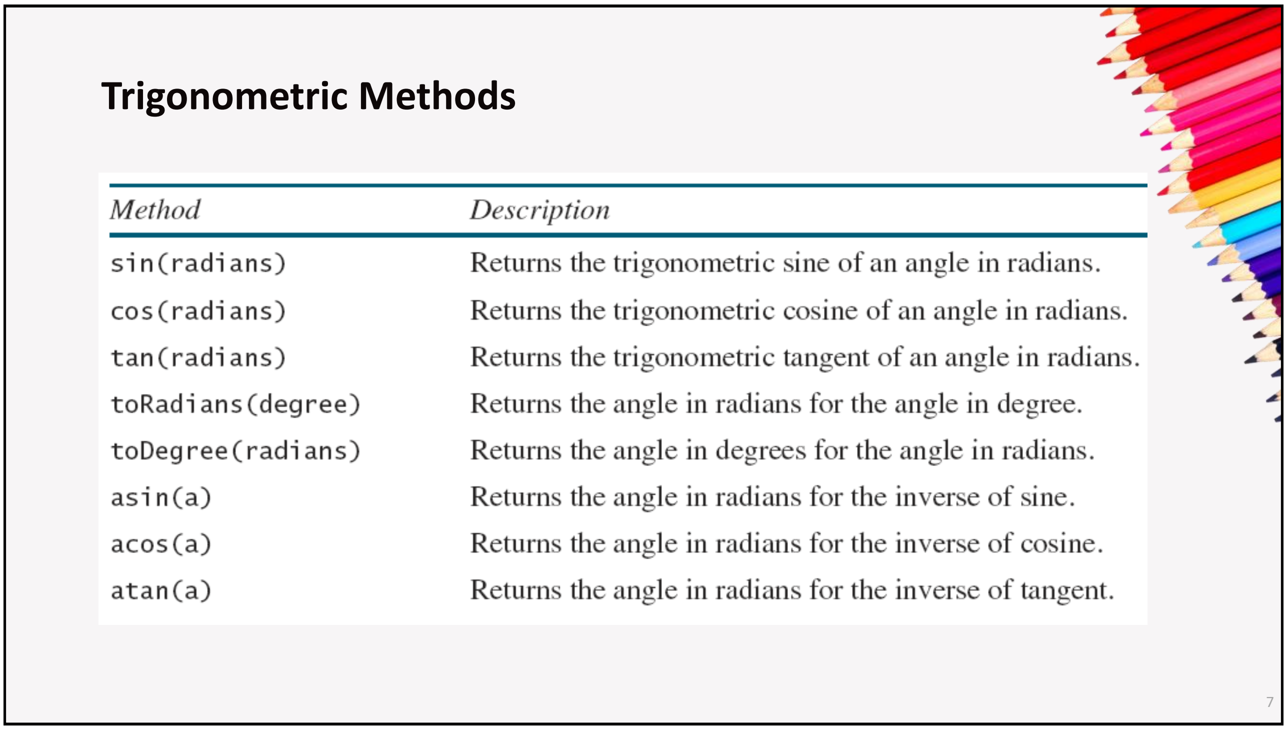
- To introduce objects and instance methods (§4.4).
- To represent strings using the **String** objects (§4.4).
- To return the string length using the **length()** method (§4.4.1).
- To return a character in the string using the **charAt(i)** method (§4.4.2).
- To use the **+** operator to concatenate strings (§4.4.3).
- To **read strings** from the console (§4.4.4).
- To **read a character** from the console (§4.4.5).
- To compare strings using the **equals** method and the **compareTo** methods (§4.4.6).

Objectives (3)

- To obtain **substring**s (§4.4.7).
- To find a character or a substring in a string using the **indexOf** method (§4.4.8).
- To program using characters and strings (GuessBirthday) (§4.5.1).
- To convert a hexadecimal character to a decimal value (HexDigit2Dec) (§4.5.2).
- To revise the lottery program using strings (LotteryUsingStrings) (§4.5.3).
- To format output using the **System.out.printf** method (§4.6).

4.2 Common Mathematical Functions

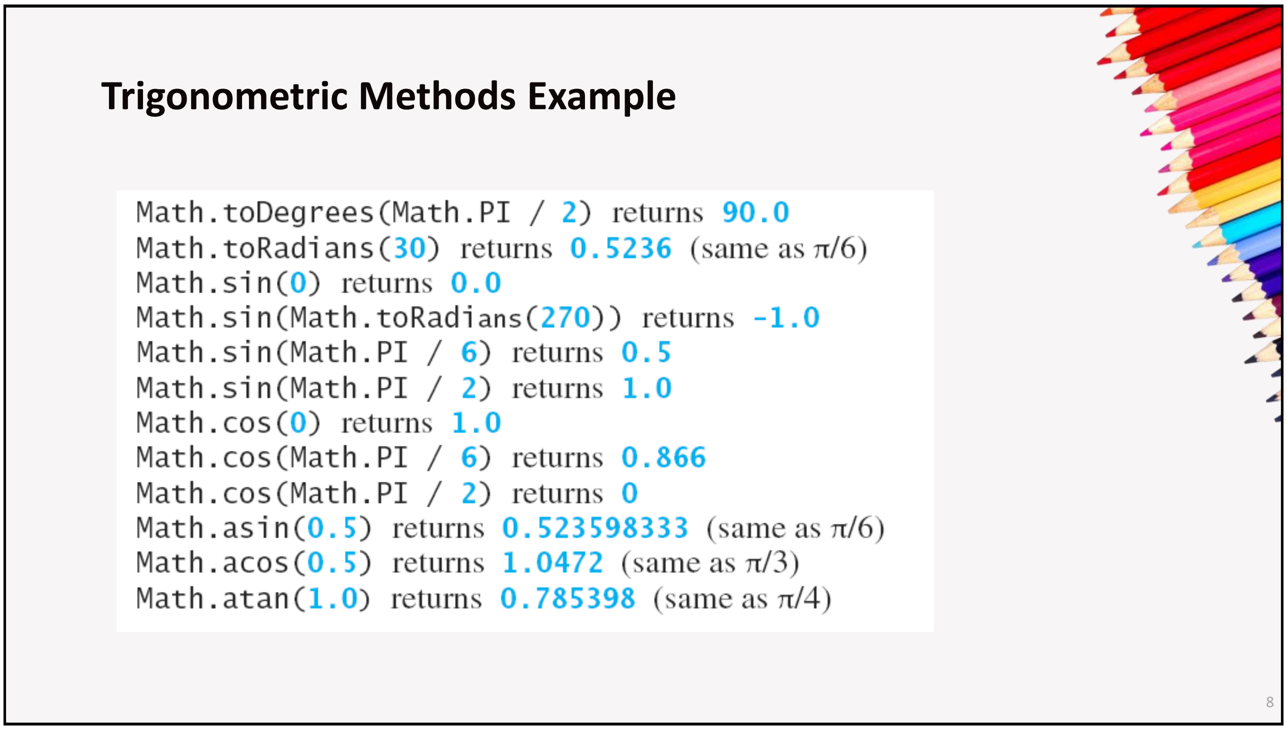
- *Java provides many useful methods in the **Math class** for performing common mathematical functions.*
- Class constants:
  - **PI**
  - **E**
- Class methods:
  - Trigonometric Methods
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods



Trigonometric Methods

Method	*Description*
`sin(radians)`	Returns the trigonometric sine of an angle in radians.
`cos(radians)`	Returns the trigonometric cosine of an angle in radians.
`tan(radians)`	Returns the trigonometric tangent of an angle in radians.
`toRadians(degree)`	Returns the angle in radians for the angle in degree.
`toDegree(radians)`	Returns the angle in degrees for the angle in radians.
`asin(a)`	Returns the angle in radians for the inverse of sine.
`acos(a)`	Returns the angle in radians for the inverse of cosine.
`atan(a)`	Returns the angle in radians for the inverse of tangent.

7



Trigonometric Methods Example

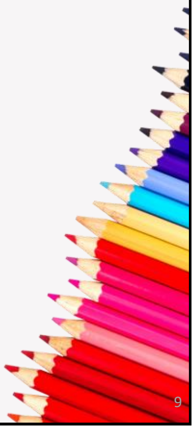
```
Math.toDegrees(Math.PI / 2) returns 90.0
Math.toRadians(30) returns 0.5236 (same as π/6)
Math.sin(0) returns 0.0
Math.sin(Math.toRadians(270)) returns -1.0
Math.sin(Math.PI / 6) returns 0.5
Math.sin(Math.PI / 2) returns 1.0
Math.cos(0) returns 1.0
Math.cos(Math.PI / 6) returns 0.866
Math.cos(Math.PI / 2) returns 0
Math.asin(0.5) returns 0.523598333 (same as π/6)
Math.acos(0.5) returns 1.0472 (same as π/3)
Math.atan(1.0) returns 0.785398 (same as π/4)
```

8

Exponent Methods

Method	*Description*
`exp(x)`	Returns e raised to power of x (e^x).
`log(x)`	Returns the natural logarithm of x ($\ln(x) = \log_e(x)$).
`log10(x)`	Returns the base 10 logarithm of x ($\log_{10}(x)$).
`pow(a, b)`	Returns a raised to the power of b (a^b).
`sqrt(x)`	Returns the square root of x ($\sqrt{x}$) for x >= 0.

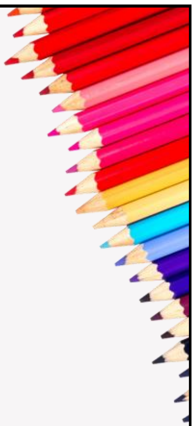
```
Math.exp(1) returns 2.71828
Math.log(Math.E) returns 1.0
Math.log10(10) returns 1.0
Math.pow(2, 3) returns 8.0
Math.pow(3, 2) returns 9.0
Math.pow(4.5, 2.5) returns 22.91765
Math.sqrt(4) returns 2.0
Math.sqrt(10.5) returns 4.24
```



9

Rounding Methods

Method	*Description*
`ceil(x)`	x is rounded up to its nearest integer. This integer is returned as a double value.
`floor(x)`	x is rounded down to its nearest integer. This integer is returned as a double value.
`rint(x)`	x is rounded up to its nearest integer. If x is equally close to two integers, the even one is returned as a double value.
`round(x)`	Returns (int)Math.floor(x + 0.5) if x is a float and returns (long)Math.floor(x + 0.5) if x is a double.



10

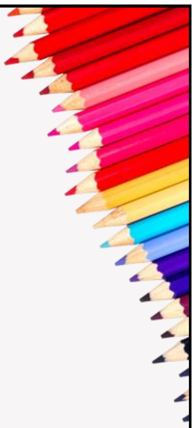
Rounding Methods Examples

```
Math.ceil(2.1) returns 3.0
Math.ceil(2.0) returns 2.0
Math.ceil(-2.0) returns -2.0
Math.ceil(-2.1) returns -2.0
Math.floor(2.1) returns 2.0
Math.floor(2.0) returns 2.0
Math.floor(-2.0) returns -2.0
Math.floor(-2.1) returns -3.0
```

```
Math.rint(2.1) returns 2.0
Math.rint(2.0) returns 2.0
Math.rint(-2.0) returns -2.0
Math.rint(-2.1) returns -2.0
Math.rint(2.5) returns 2.0
Math.rint(-2.5) returns -2.0
Math.round(2.6f) returns 3
Math.round(2.0) returns 2
Math.round(-2.0f) returns -2
Math.round(-2.6) returns -3
```

11

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.
- **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
```

12

The random Method

- **`random()`** generates a random double value greater than or equal to 0.0 and less than 1.0 (0.0 <= Math.random() < 1.0).

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

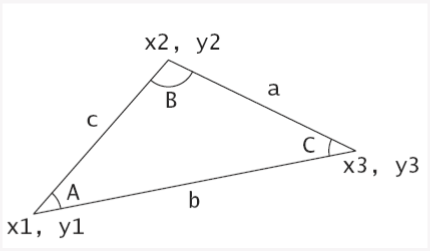
`50 + (int)(Math.random() * 50)` $\longrightarrow$ Returns a random integer between `50` and `99`.

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

13

Case Study: Computing Angles of a Triangle

- 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.



```
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))
```

ComputeAngles

Run

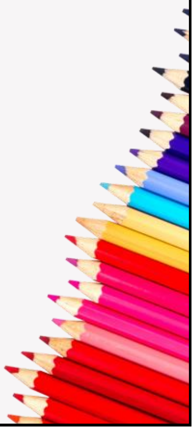
14

Characters and Strings

4.3 Character Data Type and Operations

- *A character data type represents a single character.*
- A character literal is enclosed in single quotation marks.

```
char letter = 'A';          (ASCII)
char numChar = '4';         (ASCII)


char letter = '\u0041';   (Unicode)
char numChar = '\u0034'; (Unicode)
```



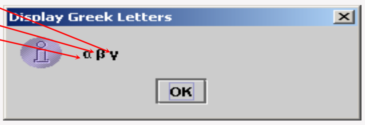
Unicode and ASCII code

- Mapping a character to its binary representation is called *encoding.*
- How characters are encoded is defined by an *encoding scheme.*
- 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 was originally designed as a **16-bit** character encoding.
  - Those characters that go beyond the original 16-bit limit are called *supplementary characters.*

Unicode Format

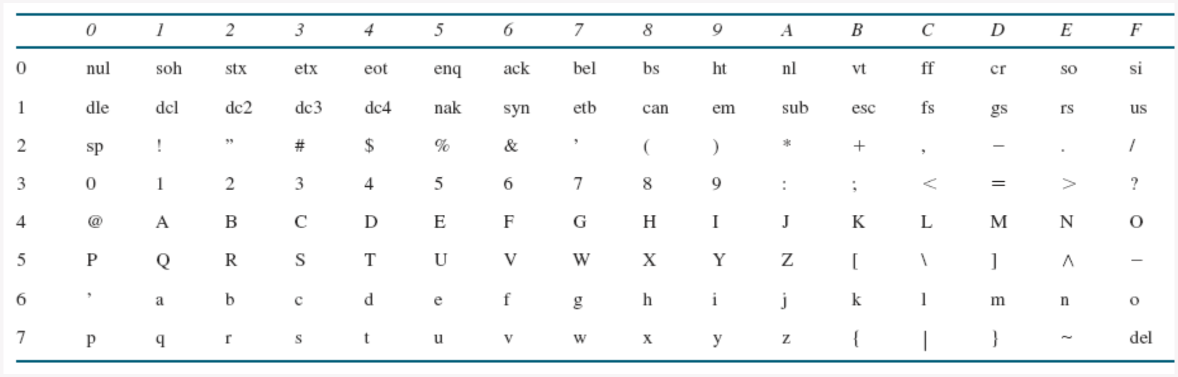
- A 16-bit Unicode takes two bytes, preceded by **\u,** expressed in four hexadecimal digits tha trun from **\u0000 to \uFFFF.**
- So, Unicode can represent `65535 + 1` characters.
- Unicode includes ASCII code, with **\u0000 to \u007F** corresponding to the 128 ASCII characters.

Unicode **\u03b1 \u03b2 \u03b3** for three Greek letters



Appendix B: ASCII Character Set

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

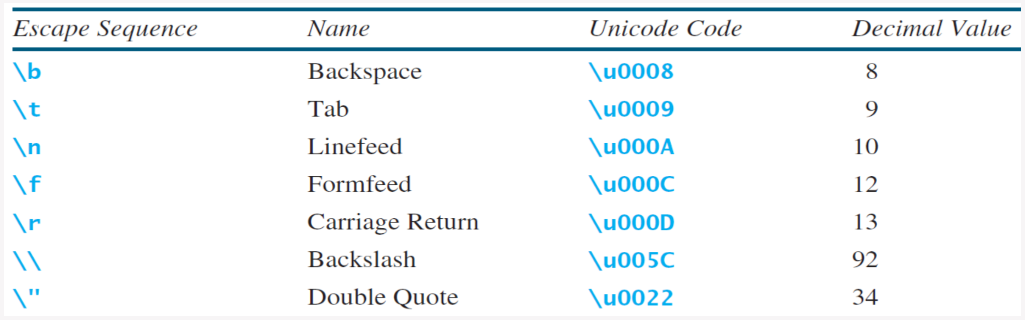


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	ff	cr	so	si
1	dle	dcl	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
2	sp	!	”	#	$	%	&	’	(	)	*	+	,	–	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	∧	—
6	‘	a	b	c	d	e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

19

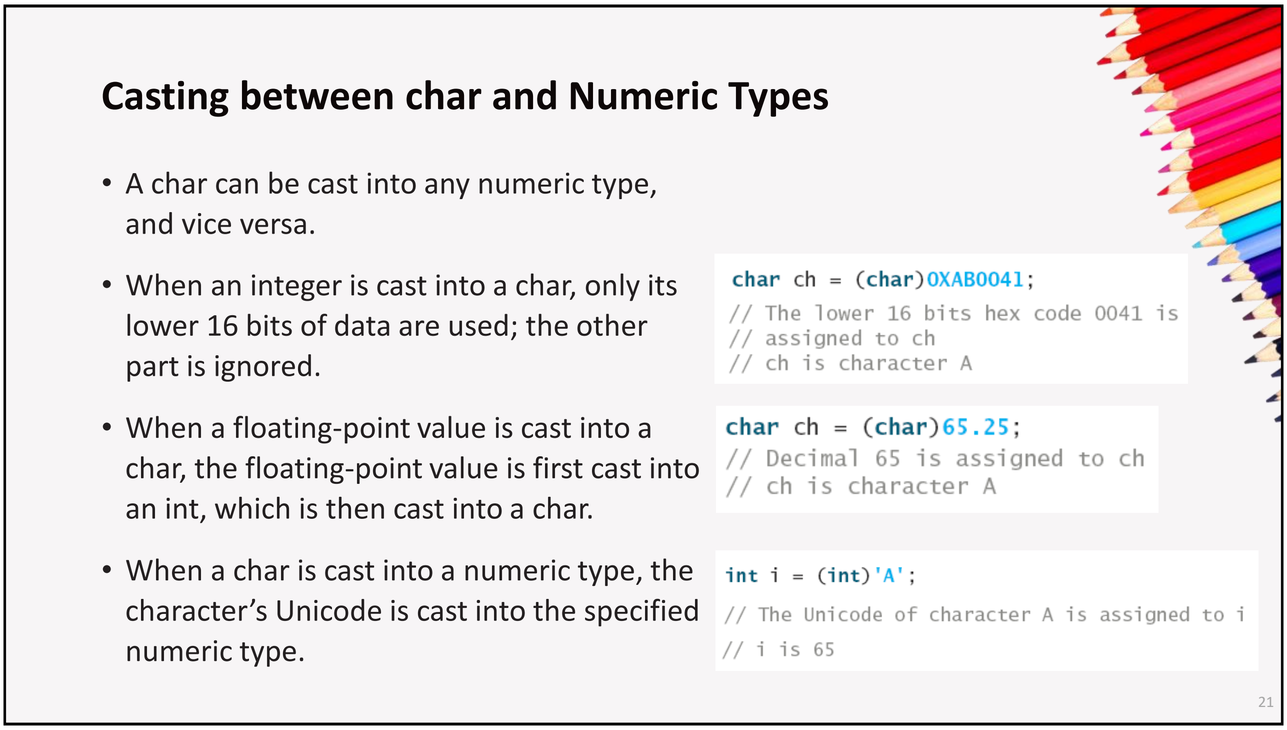
Escape Sequences for Special Characters

- Java uses an escape sequence to represent special character.
- The escape sequence consists of a backslash(\) followed by a character or a combination of digits.
  - The backslash \ is called an ***escape character.***



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

20



Casting between char and Numeric Types

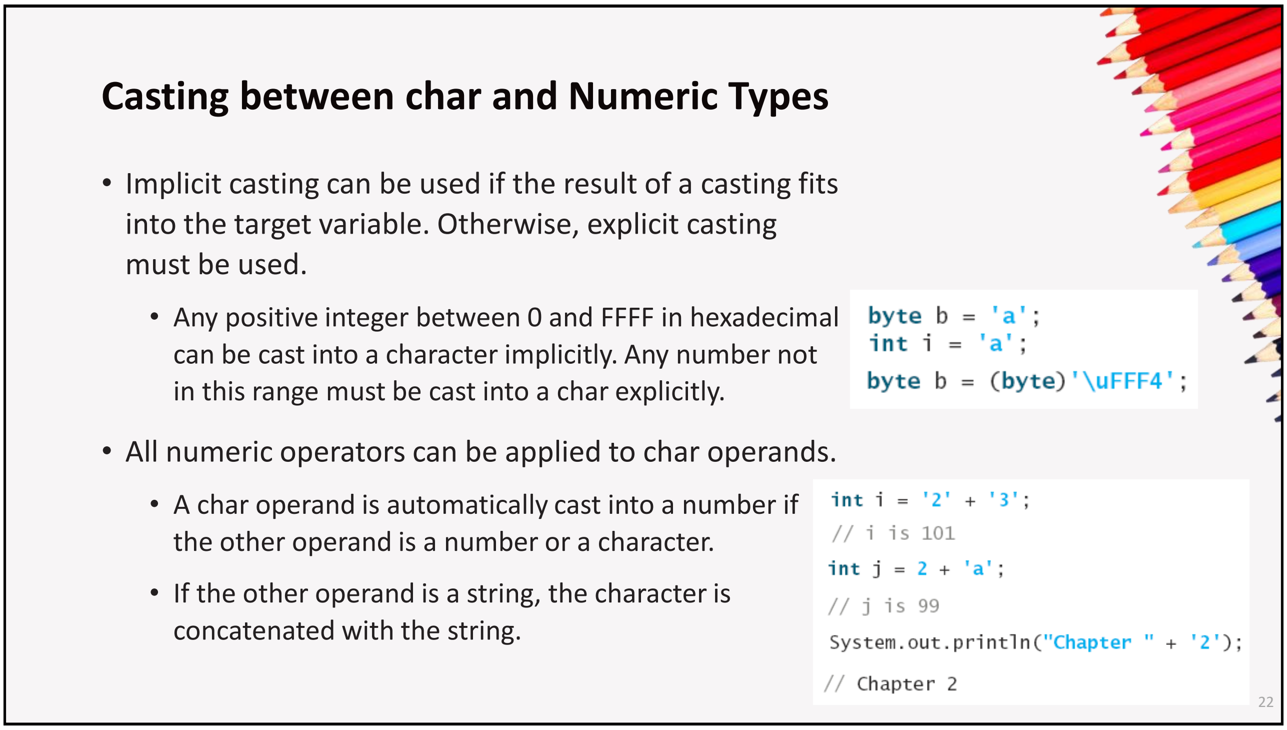
- A char can be cast into any numeric type, and vice versa.
- When an integer is cast into a char, only its lower 16 bits of data are used; the other part is ignored.
- When a floating-point value is cast into a char, the floating-point value is first cast into an int, which is then cast into a char.
- When a char is cast into a numeric type, the character's Unicode is cast into the specified numeric type.

```
char ch = (char)0XAB0041;
// The lower 16 bits hex code 0041 is
// assigned to ch
// ch is character A
```

```
char ch = (char)65.25;
// Decimal 65 is assigned to ch
// ch is character A
```

```
int i = (int)'A';
// The Unicode of character A is assigned to i
// i is 65
```

21



Casting between char and Numeric Types

- Implicit casting can be used if the result of a casting fits into the target variable. Otherwise, explicit casting must be used.
  - Any positive integer between 0 and FFFF in hexadecimal can be cast into a character implicitly. Any number not in this range must be cast into a char explicitly.
- All numeric operators can be applied to char operands.
  - A char operand is automatically cast into a number if the other operand is a number or a character.
  - If the other operand is a string, the character is concatenated with the string.

```
byte b = 'a';
int i = 'a';
byte b = (byte)'\uFFF4';
```

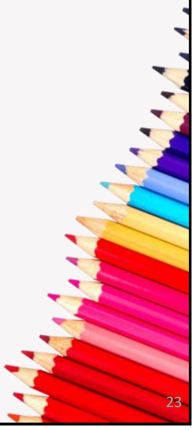
```
int i = '2' + '3';
// i is 101
int j = 2 + 'a';
// j is 99
System.out.println("Chapter " + '2');
// Chapter 2
```

22

Comparing and Testing Characters

- Two characters can be compared using the relational operators just like comparing two numbers.
- This is done by comparing the Unicodes of the two 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");
```

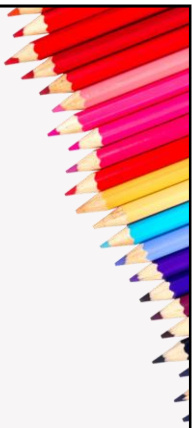


Methods in the Character Class

- For convenience, Java provides the following methods in the **Character** class for testing characters.

Usage: `Character.methodname(ch)`

Method	*Description*
`isDigit(ch)`	Returns true if the specified character is a digit.
`isLetter(ch)`	Returns true if the specified character is a letter.
`isLetterOfDigit(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.



Methods in the Character Class

```
System.out.println("isDigit('a') is " + Character.isDigit('a'));
System.out.println("isLetter('a') is " + Character.isLetter('a'));
System.out.println("isLowerCase('a') is "
  + Character.isLowerCase('a'));
System.out.println("isUpperCase('a') is "
  + Character.isUpperCase('a'));
System.out.println("toLowerCase('T') is "
  + Character.toLowerCase('T'));
System.out.println("toUpperCase('q') is "
  + Character.toUpperCase('q'));
```

```
isDigit('a') is false
isLetter('a') is true
isLowerCase('a') is true
isUpperCase('a') is false
toLowerCase('T') is t
toUpperCase('q') is Q
```

4.4 The String Type

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

```
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.
- 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.

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.

27

Instance Method and Static Method

- 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*
- The syntax to invoke an instance method is

  `referenceVariable.methodName(arguments)`

- 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.
- ***`ClassName.methodName(arguments)`***

28

Getting String Length

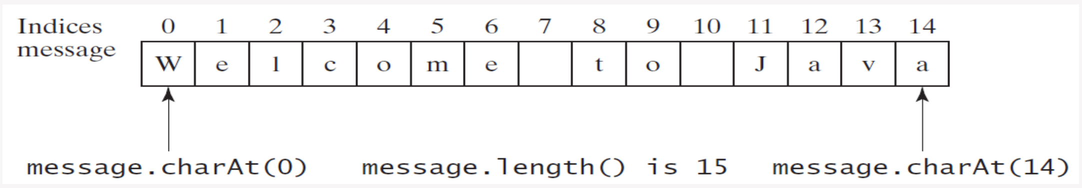
- You can use the length() method to return the number of characters in a string.

```
String message = "Welcome to Java";
System.out.println("The length of " + message + " is "
  + message.length());
```

```
The length of Welcome to Java is 15
```

Getting Characters from a String

- The s.charAt(index) method can be used to retrieve a specific character in a string s, where the index is between 0 and s.length()–1.



```
String message = "Welcome to Java";
System.out.println("The first character in message is "
   + message.charAt(0));
```

String Concatenation

- You can use the **concat** method to concatenate two strings. `String s3 = s1.concat(s2);`
- You can use the plus (**+**) operator or (**+=**) operator to concatenate two strings for convenience. `String s3 = s1 + s2;` `message += " and Java is fun";`

```
// 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
```

- If **i = 1 and j = 2,** what is the output of the following statements?

```
System.out.println("i + j is " + (i + j));
```

31

Converting Strings

- The **toLowerCase()** method returns a new string with all lowercase letters and the **toUpperCase()** method returns a new string with all uppercase letters.
- The **trim()** method returns a new string by eliminating whitespace characters from both ends of the string.
  - The characters **' ', \t, \f, \r, or \n** are known as *whitespace characters.*

`"Welcome".toLowerCase()` returns a new string `welcome`.
`"Welcome".toUpperCase()` returns a new string `WELCOME`.
`"\t Good Night \n".trim()` returns a new string `Good Night`.

32

Reading a String from the Console

- To read a string from the console, invoke the **next()** method on a Scanner object.
  - The **next()** method reads a string that ends with a whitespace character.
- You can use the **nextLine()** method to read an entire line of text.
  - The **nextLine()** method reads a string that ends with the *Enter key pressed.*

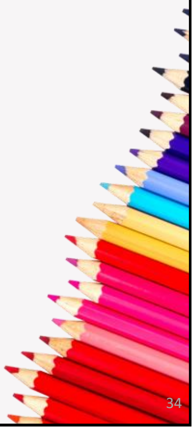
```
Scanner input = new Scanner(System.in);
System.out.print("Enter three words separated by spaces: ");
String s1 = input.next();
String s2 = input.next();
String s3 = input.next();
System.out.println("s1 is " + s1);
System.out.println("s2 is " + s2);
System.out.println("s3 is " + s3);
```

```
Scanner input = new Scanner(System.in);
System.out.println("Enter a line: ");
String s = input.nextLine();
System.out.println("The line entered is " + s);
```

Reading a Character from the Console

- To read a character from the console, use the **nextLine()** method to read a string and then invoke the **charAt(0)** method on the string to return a character.

```
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);
```



Comparing Strings

- Comparison Methods for **String Objects:**

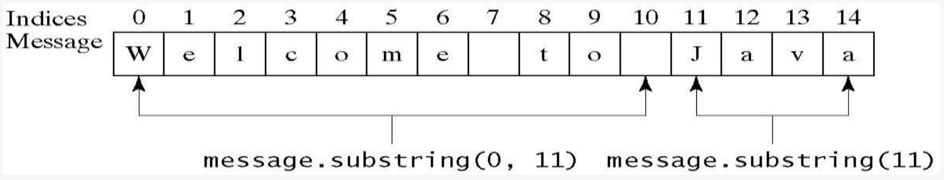
Method	*Description*
`equals(s1)`	Returns true if this string is equal to string `s1`.
`equalsIgnoreCase(s1)`	Returns true if this string is equal to string `s1`; it is case insensitive.
`compareTo(s1)`	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater 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.
`contains(s1)`	Returns true if s1 is a substring in this string.

OrderTwoCities Run

Obtaining Substrings

- You can obtain a substring from a string using the **substring** method in the String class.

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 4.2. Note that the character at `endIndex` is not part of the substring.



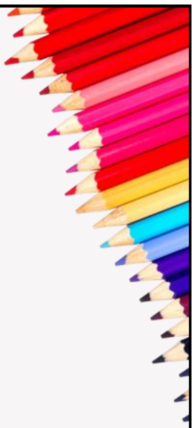
Finding a Character or a Substring in a String

Method	*Description*
`index(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.

37

Finding a Character or a Substring in a String

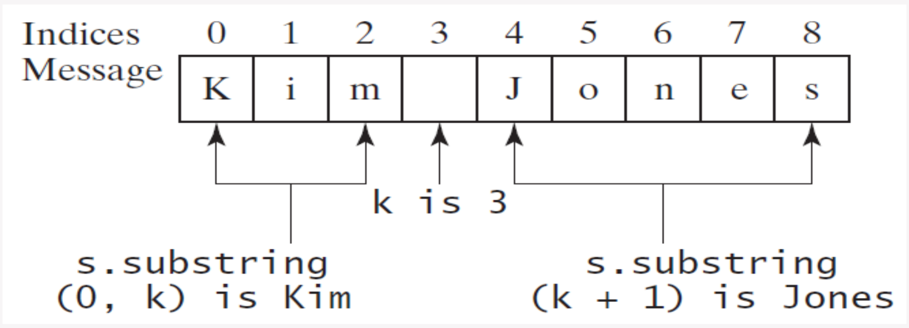
```
"Welcome to Java".indexOf('W') returns 0.
"Welcome to Java".indexOf('o') returns 4.
"Welcome to Java".indexOf('o', 5) returns 9.
"Welcome to Java".indexOf("come") returns 3.
"Welcome to Java".indexOf("Java", 5) returns 11.
"Welcome to Java".indexOf("java", 5) returns -1.

"Welcome to Java".lastIndexOf('W') returns 0.
"Welcome to Java".lastIndexOf('o') returns 9.
"Welcome to Java".lastIndexOf('o', 5) returns 4.
"Welcome to Java".lastIndexOf("come") returns 3.
"Welcome to Java".lastIndexOf("Java", 5) returns -1.
"Welcome to Java".lastIndexOf("Java") returns 11.
```



38

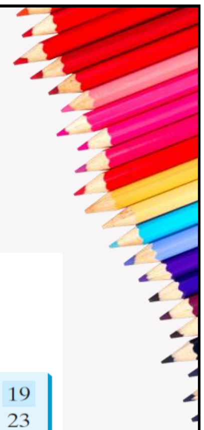
Finding a Character or a Substring in a String

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



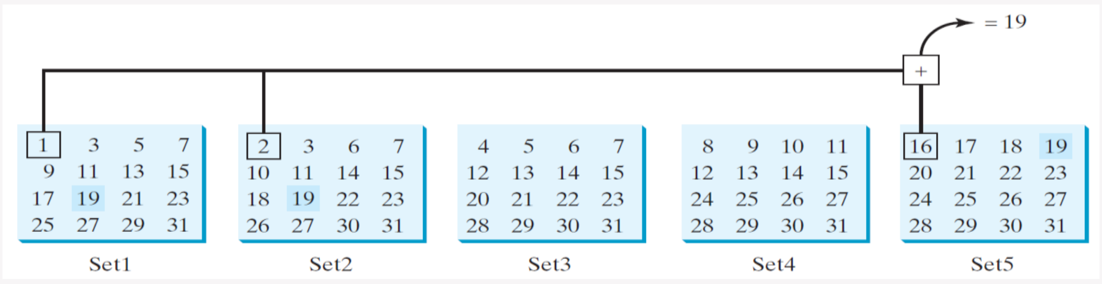
Conversion between Strings and Numbers

- To convert a string into an int value, use the Integer.parseInt method
  - `int intValue = Integer.parseInt(intString);`
- To convert a string into a double value, use the Double.parseDouble method
  - `double doubleValue = Double.parseDouble(doubleString);`
- You can convert a number into a string, simply use the string concatenating operator as follows:
  - `String s = number + "";`



Case Study: Guessing Birthday

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



Set1			
1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

Set2			
2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

Set3			
4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

Set4			
8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

Set5			
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

GuessBirthday   Run

Mathematics Basis for the Game

- A number between **1** and **31** can be represented using a five-digit binary number.
- A five-digit binary number can be obtained by adding binary numbers **1, 10, 100, 1000, or 10000.**

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

$$\begin{array}{rr} & b_5\,0\,0\,0\,0 \\ & b_4\,0\,0\,0 \\ & b_3\,0\,0 \\ & b_2\,0 \\ + & b_1 \\ \hline & b_5 b_4 b_3 b_2 b_1 \end{array} \qquad \begin{array}{rr} & 10000 \\ & 10 \\ + & 1 \\ \hline & 10011 \\ & 19 \end{array} \qquad \begin{array}{rr} & 10000 \\ & 1000 \\ & 100 \\ & 10 \\ + & 1 \\ \hline & 11111 \\ & 31 \end{array}$$

Case Studies

- Converting a Hexadecimal Digit to a Decimal Value
  - Write a program that converts a hexadecimal digit into a decimal value.

HexDigit2Dec

Run

- 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.

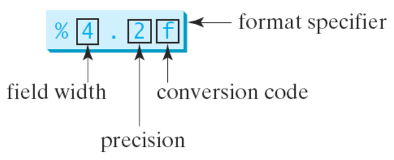
LotteryUsingStrings

Run

4.6 Formatting Console Output

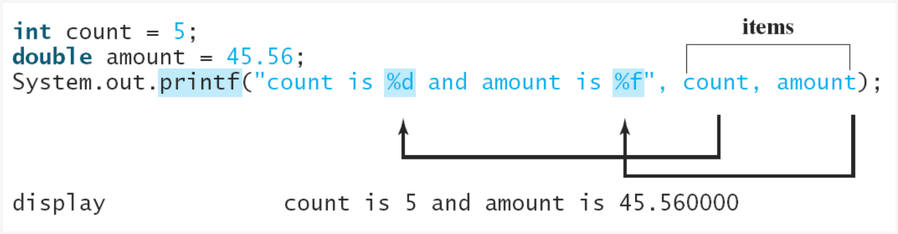
- *You can use the* ***System.out.printf*** *method to display formatted output on the console.*
- `System.out.printf(format, items);`
  - Where format is a string that may consist of substrings and format specifiers.
  - A **format** specifier specifies how an item should be displayed.
  - An **item** may be a numeric value, character, boolean value, or a string.
  - Each specifier begins with a percent sign.

```java
double amount = 12618.98;
double interestRate = 0.0013;
double interest = amount * interestRate;
System.out.printf("Interest is $%4.2f",
  interest);
```



Frequently-Used Specifiers

Format Specifier	*Output*
%b	a Boolean value
%c	a character
%d	a decimal integer
%f	a floating-point number
%e	a number in standard scientific notation
%s	a string



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

display            count is 5 and amount is 45.560000
```

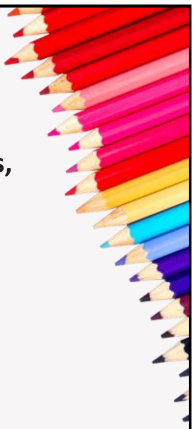
FormatDemo

```
public class FormatDemo {
  public static void main(String[] args) {
    // Display the header of the table
    System.out.printf("%-10s%-10s%-10s%-10s%-10s\n", "Degrees",
      "Radians", "Sine", "Cosine", "Tangent");

    // Display values for 30 degrees
    int degrees = 30;
    double radians = Math.toRadians(degrees);
    System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees,
      radians, Math.sin(radians), Math.cos(radians),
      Math.tan(radians));

    // Display values for 60 degrees
    degrees = 60;
    radians = Math.toRadians(degrees);
    System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees,
      radians, Math.sin(radians), Math.cos(radians),
      Math.tan(radians));
  }
}
```

FormatDemo

Run

Chapter Summary



Chapter Summary

- Java provides the mathematical methods **sin, cos, tan, asin, acos, atan, toRadians, toDegree, exp, log, log10, pow, sqrt, cell, floor, rint, round, min, max, abs,** and **random** in the Math class for performing mathematical functions.
- The character type char represents a single character.
- The Character class contains the methods **isDigit, isLetter, isLetterOrDigit, isLowerCase, isUpperCase** for testing whether a character is a digit, letter, lowercase, and uppercase. It also contains the **toLowerCase** and **toUpperCase** methods for returning a lowercase or uppercase letter.
- A string is a sequence of characters. A string value is enclosed in matching double quotes ("). A character value is enclosed in matching single quotes (').
- Strings are objects in Java.

Chapter Summary

- A method that can only be invoked from a specific object iscalled an *instance method.* A non-instance method is called a *static method,* which can be invoked without using an object.
- You can get the length of a string by invoking its **length()** method, retrieve a characterat the specified index in the string using the **charAt(index)** method, and use the **indexOf** and **lastIndexOf** methods to find a character or a substring in a string.
- You can use the **concat** method to concatenate two strings, or the plus (**+**) operator to concatenate two or more strings.
- You can use the **substring** method to obtain a substring from the string.
- You can use the **equals** and **compareTo** methods to compare strings**.**
- The **printf** method can be used to display a formatted output using format specifiers.

Programming Exercises

1, 5, 6, 9, 11, 13,
17, 20, 21, 22, 25