CS1150
Principles of Computer Science
Math Functions, Characters and Strings (Part II)

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

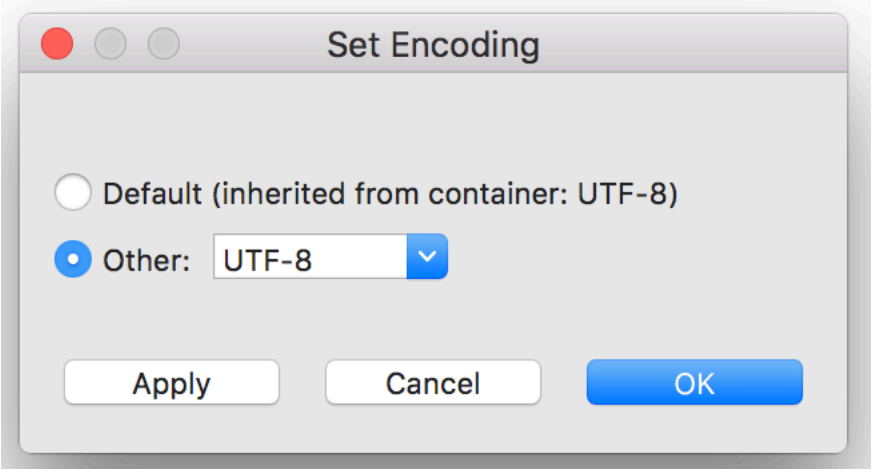
How to generate a random character?

- Every character has a unique Unicode between 0 and FFFF in hexadecimal (65535 in decimal)
  - To generate a random character: generate a random integer between 0 and 65535 (inclusive)
  - char c = (**char**)(Math.random() * (**65535** + **1**));

```
// char c = 97 is equivalent to char c = (char) 97
int d = (int)(Math.random() * (65535 + 1));
System.out.println(d);     // 44188
System.out.println((char)d);     // 겜
```

Seeing ?? In Eclipse

- With the current window selected --> Edit --> Set Encoding --> Choose UTF-8 in Other



How to generate a random lower case character?

- The Unicodes for lowercase letters are consecutive integers starting from the Unicode for a, then that for b, c, . . . , and z
  - The Unicode for ‘a’ is (int)'a’
  - A random integer between (int)'a' and (int)'z'
    - (int)((int)'a' + Math.random() * ((int)'z' - (int)'a' + 1))

all numeric operators can be applied to the char operands

- (int)('a' + Math.random() * ('z' - 'a' + 1))
- char c = (char)(’a' + Math.random() * (’z' - ’a' + 1));

How to generate a random character?

- A random character between any two characters ch1 and ch2 with ch1 < ch2 can be generated as:
  - (char)(ch1 + Math.random() * (ch2 – ch1 + 1))
  - Example: random upper case letter
    - (char)('A' + Math.random() * ('Z'– 'A' + 1))
  - Example: random numeric character
    - (char)('0' + Math.random() * ('9'– '0' + 1))

How to convert a numeric int character to its int value?

- Converting ‘0’ to 0, etc.
- Example: how to convert ‘0’ to 0?
  - ‘0’ - ‘0’ is 0
  - ‘1’ - ‘0’ is 1
  - ‘2’ - ‘0’ is 2
  - ……

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 a predefined class in Java just like the Math class and Character class (note all classes have names first letter capitalized).

The String Type

- A **char** is in single quotes and a **String** is in double quotes
  - char middleInitial = "M"; // Error - can't convert String to char
  - char middleInitial = 'M'; // Correct

  - string studentName = "Max"; // Error - uppercase "String"
  - String studentName = 'Max'; // Error - double quotes
  - String studentName = "Max"; // Correct

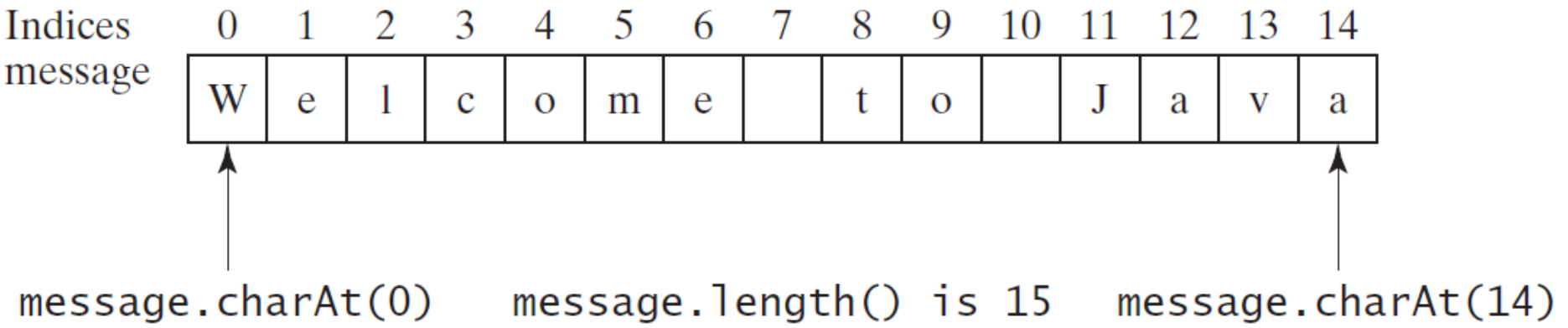
Simple Methods for String Objects

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.

Getting String Length

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

Getting Characters from a String



Index runs from 0 to **length() minus 1**. Examples:

```
String message = "Welcome to Java";
System.out.println("The first character in message is " +
                   message.charAt(0));
System.out.println("The last character is " +
                   message.charAt(message.length()-1));
```

How to iterate the String through each character?

Converting Strings

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

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

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

trim(): Returns a string with all white space characters removed from **front or end** of string

White space characters include space, tab, line feed, form feed, carriage return

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
```

String Concatenation

- Notes about +
  - When only one of the operands is a String, the other will be converted into a String and concatenation occurs
  - When neither operand is a String, then addition operation occurs
- Example: String1.java

Reading a String from the Console

next(): reads string until a white space character is reached

```
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);
```

Reading a String from the Console

nextLine(): reads string until an enter key is reached

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

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

String s = input.nextLine();

System.out.println("The String entered is " + s);
```

Reading a Character from the Console

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

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

String s = input.nextLine();  // input.next() works too

char ch = s.charAt(0);

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

Shortcut: input.nextLine().charAt(0);

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.

Comparing Strings

- Attempting to compare Strings with relational operators will cause errors
  - Use equals() or equalsIgnoreCase() instead of ==
    - str1.equals(str2) // returns true or false
  - Use compareTo() instead of <, >, <=, >=
    - str1.compareTo(str2) // returns an integer
    - > 0 if str1 is greater than str2
    - = 0 if str1 is equal to str2
    - < 0 if str1 is less than str2
  - Example: Order2Cities.java

Case Study: Lottery Program Using Strings

- Generates a random two-digit number, then prompts the user to enter a two-digit number, and determines whether the user wins according to the following rule:

1. If the user input matches the lottery number in the exact order, the award is $10,000.
2. If all the digits in the user input match all the digits in the lottery number, but not in the right order, the award is $3,000.
3. If one digit in the user input matches a digit in the lottery number, the award is $1,000.

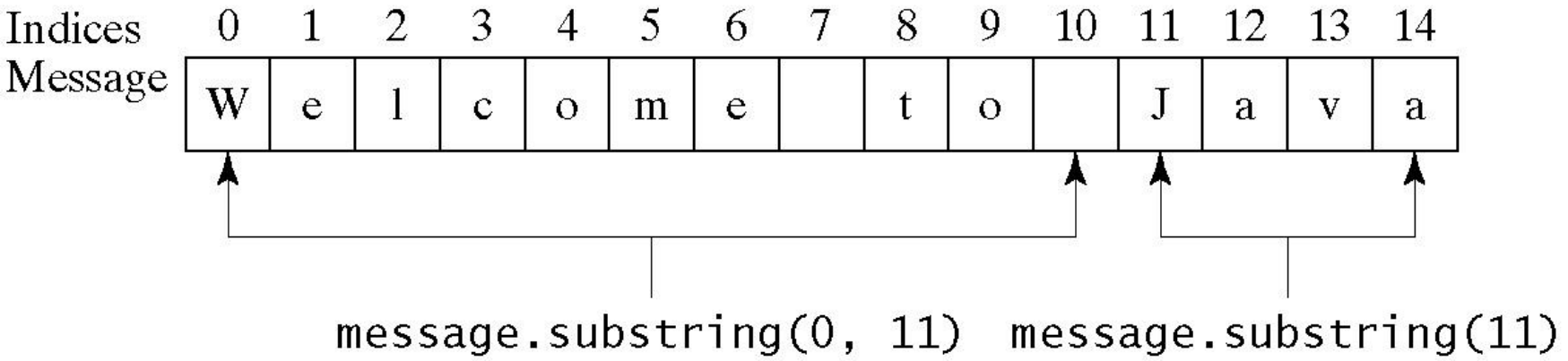
Lottery.java

Conversion between Numbers and Strings

- Using +
  - When one of the operands is a String, the other will be converted into a String and concatenation occurs
  - String s = 6 + “”; // converts integer 6 to a string

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.
`substring(beginIndex, endIndex)`	Returns this string's substring that begins at the specified `beginIndex` and extends to the character at index `endIndex – 1`. Note that the character at `endIndex` is not part of the substring.

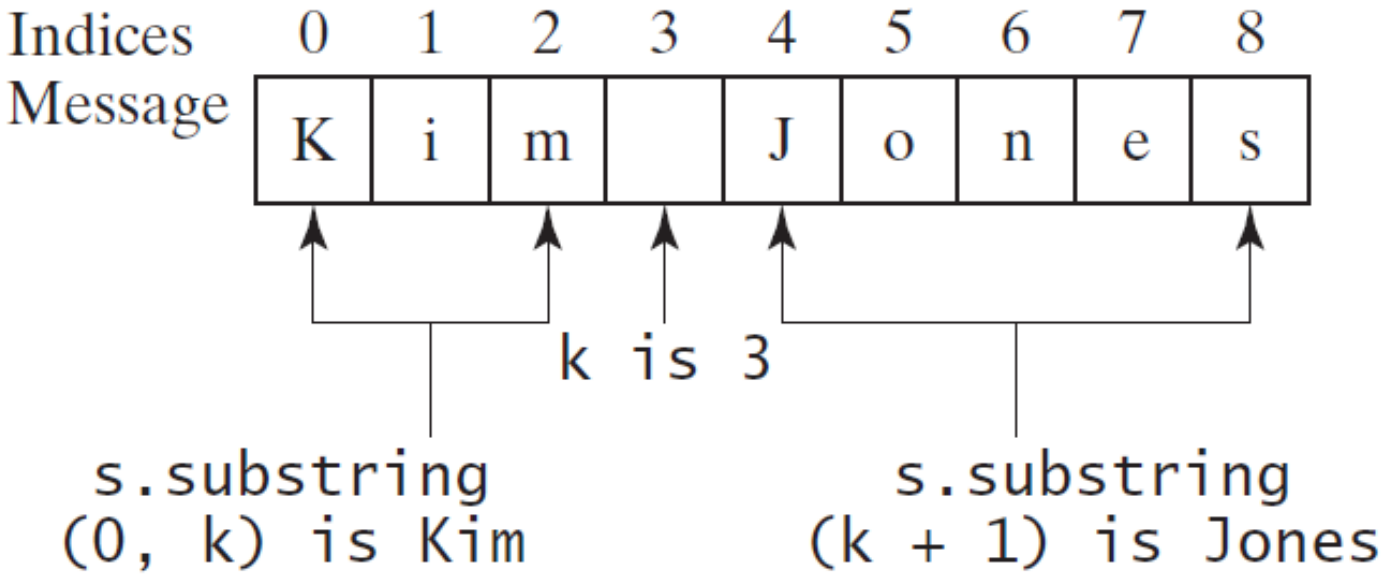


Finding a Character or a Substring in a String

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.

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

- Integer.parseInt()
  - Converts a **String** into an integer. Example:

```
String numericString = "817";
int number = Integer.parseInt(numericString);
System.out.println("Converted string " + numericString + " into integer "
+ number);
```

- Double.parseDouble()
  - Converts a **String** into a double

Formatting Output

Use the printf statement.

System.out.**printf**(format, items);

Where format is a string that may consist of substrings and format specifiers.

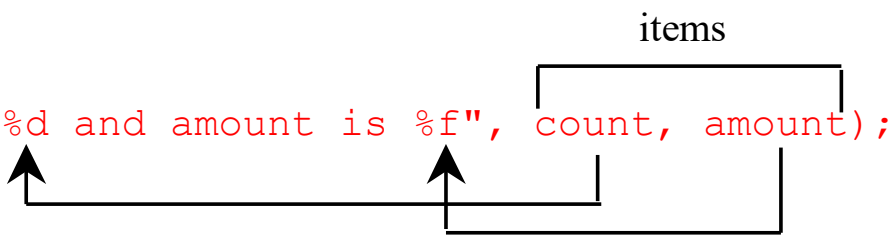
A format specifier specifies how an item should be displayed. An item may be a numeric value, character, boolean value, or a string. Each specifier begins with a percent sign %.

Does not work with println.

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;
System.out.printf("count is %d and amount is %f", count, amount);
```




```
display              count is 5 and amount is 45.560000
```

Formatting Output

- If printing several items in System.out.printf, the format specifiers must be in correct order

```
System.out.printf("%8d%5.2f", 123, 3.45);
// Correct: integer first, then float

System.out.printf("%8d%5.2f", 3.45, 123); // Error
```

Summary

- Math class and methods
- Characters
- Strings