Table 3.2 Useful Static Methods in the `Math` Class

Method	Description	Example
`abs`	absolute value	`Math.abs(-308)` returns `308`
`ceil`	ceiling (rounds upward)	`Math.ceil(2.13)` returns `3.0`
`floor`	floor (rounds downward)	`Math.floor(2.93)` returns `2.0`
`max`	maximum of two values	`Math.max(45, 207)` returns `207`
`min`	minimum of two values	`Math.min(3.8, 2.75)` returns `2.75`
`pow`	power (general exponentiation)	`Math.pow(3, 4)` returns `81.0`
`random`	random value	`Math.random()` returns a random `double` value k such that $0.0 \leq k < 1.0$
`round`	round real number to nearest integer	`Math.round(2.718)` returns `3`
`sqrt`	square root	`Math.sqrt(2)` returns `1.4142135623730951`

Table 3.3 Useful Methods of `String` Objects

Method	Description	Example (assuming `s` is `"hello"`)
`charAt(index)`	character at a specific index	`s.charAt(1)` returns `'e'`
`endsWith(text)`	whether or not the string ends with some text	`s.endsWith("llo")` returns `true`
`indexOf(text)`	index of a particular character or `String` (`-1` if not present)	`s.indexOf("o")` returns `4`
`length()`	number of characters in the string	`s.length()` returns `5`
`startsWith(text)`	whether or not the string starts with some text	`s.startsWith("hi")` returns `false`
`substring(start, stop)`	characters from start index to just before stop index	`s.substring(1, 3)` returns `"el"`
`toLowerCase()`	a new string with all lowercase letters	`s.toLowerCase()` returns `"hello"`
`toUpperCase()`	a new string with all uppercase letters	`s.toUpperCase()` returns `"HELLO"`

Useful Methods of `Scanner` Objects

Method	Description
`next()`	Reads and returns the next token as a `String`
`nextDouble()`	Reads and returns a `double` value
`nextInt()`	Reads and returns an `int` value
`nextLine()`	Reads and returns the next line of input as a `String`
`hasNext()`	Returns `true` if there is another token to be read
`hasNextDouble()`	Returns `true` if there is another token to be read and if it can be interpreted as a `double`
`hasNextInt()`	Returns `true` if there is another token to be read and if it can be interpreted as an `int`
`hasNextLine()`	Returns `true` if there is another line of input to be read

Table 4.5 Useful Methods of the `Character` Class

Method	Description	Example
`getNumericValue(ch)`	Converts a character that looks like a number into that number	`Character.getNumericValue('6')` returns `6`
`isDigit(ch)`	Whether or not the character is one of the digits `'0'` through `'9'`	`Character.isDigit('X')` returns `false`
`isLetter(ch)`	Whether or not the character is in the range `'a'` to `'z'` or `'A'` to `'Z'`	`Character.isLetter('f')` returns `true`
`isLowerCase(ch)`	Whether or not the character is a lowercase letter	`Character.isLowerCase('Q')` returns `false`
`isUpperCase(ch)`	Whether or not the character is an uppercase letter	`Character.isUpperCase('Q')` returns `true`
`toLowerCase(ch)`	The lowercase version of the given letter	`Character.toLowerCase('Q')` returns `'q'`
`toUpperCase(ch)`	The uppercase version of the given letter	`Character.toUpperCase('x')` returns `'X'`