Lecture 04
Math functions, Chars, and Strings
Math Class

• Talked about it a bit with Math.random()

• Built-in to Java.lang, so don’t have to import it.

• Static, so we don’t need an instance of it.

• What other methods and constants are there?

• To the Javadoc!
Trig Methods

• sin(radians) -> Sine of an angle in radians
• cos(radians) -> Cosine of an angle in radians
• tan(radians) -> Tangent of an angle in radians
• toRadians(degrees) -> Angle in radians of angle in degrees
• toDegree(radians) -> Angle in degrees of angle in radians
• asin(a) -> angle in radians for the inverse of sine
• acos(a) -> angle in radians for the inverse of cosine
• atan(a) -> angle in radians for the inverse of tangent
Radians and Degrees

- 1 degree = $\frac{\pi}{180}$ radians
- 90 degrees = $\frac{\pi}{2}$ radians
• **Note:** This is not a geometry class! I'll give you formulas if we need them, you won't have to figure them out on your own.
Exponent Methods

• \( \exp(e) \rightarrow e^x \)
• \( \log(e) \rightarrow \text{natural log of } x \)
• \( \log_{10}(e) \rightarrow \text{base 10 log of } x \)
• \( \text{pow}(a, b) \rightarrow a^b \)
• \( \sqrt{x} \rightarrow \text{square root of } x \)
Rounding Methods

- ceil(x) -> rounded UP to nearest int
- floor(x) -> rounded DOWN to nearest int
- rint(x) -> rounded to nearest int. if right in the middle, rounds to even int
- round(x) -> rounded to nearest int. (rounds up from .5)
public class RoundingMethods {

    public static void main(String[] args) {
        double num = -2.1;

        System.out.println(Math.ceil(num));
        System.out.println(Math.floor(num));
        System.out.println(Math.rint(num));
        System.out.println(Math.round(num));
    }
}

min, max, abs

- min and max of 2 numbers
- ex. Math.max(2,3)
- Math.abs(-2.1)
min, max, abs

When would you use these?

• min and max are useful when sorting

• abs can be used for getting distance between points
Chars

• 1 single character

• Character literal in single quotation marks. (if you use "'", it's not a char anymore, it's a String)

• ex. char letter = 'B';

• ex. char nextChar = '7';
Special Characters: Escape sequences

- Super important when parsing data
- It's how you put special characters into Strings
- Names come from old printer commands
- even though they are technically 2 characters, they are 1 Char each
Escape sequences

- `\b` -> Backspace
- `\t` -> Tab
- `\n` -> Linefeed
- `\f` -> Formfeed
- `\r` -> Carriage Return
- `\` -> Backslash
- `"` -> Double quote
public class SpecialChars {
    public static void main(String[] args) {

        System.out.println("Tab \t example");
        System.out.println("Linefeed \n example");
        System.out.println("Backslash \ \ example");
        System.out.println("Double quote " example");
    }
}
\ \\ \ ------ \ BACKSLASH
\\\\---- \ REAL BACKSLASH
\\\\\\--- \ REAL REAL BACKSLASH
\\\\\\\\---- \ ACTUAL BACKSLASH, FOR REAL THIS TIME
\\\\\\\\\\-- \ ELDER BACKSLASH
\\\\\\\\\\\\--- \ BACKSLASH WHICH ESCAPES THE SCREEN AND ENTERS YOUR BRAIN
\\\\\\\\\\\\\\--- \ BACKSLASH SO REAL IT TRANSCENDS TIME AND SPACE
\\\\\\\\\\\\\\\\--- \ BACKSLASH TO END ALL OTHER TEXT
\\\\\\\\\\\\\\\\\\\\--- \ THE TRUE NAME OF BA'AL, THE SOUL-EATER
Helpful methods in Character class

- isDigit(ch) -> is true if a number
- isLetter(ch) -> is true if a letter
- isLetterOrDigit(ch) -> is true if letter or digit (not escape character)
- isLowerCase(ch) -> is true if lowercase letter
- isUpperCase(ch) -> is true if uppercase letter
- toLowerCase(ch) -> returns the lowercase of the character
- toUpperCase(ch) -> returns the uppercase of the character
Helpful methods in Character class

Character.isDigit('a')
Strings

- Used to represent a string of characters

- Syntax:
  
  - String exampleString = "Hey, this thing is a String";

- Not a primitive type (like int or double), but reference type
Useful methods for Strings

- length(): number of characters in a string
- charAt(index): returns the character at the point asked
- concat(s1): concatenates this string with s1
- toUpperCase(): returns string, but all letters uppercase
- toLowerCase(): returns string, but all letters uppercase
- trim(): returns a new string with no whitespace in front or behind
Get first and last character of a string
public class GetCharactersFromString {
    public static void main(String[] args) {
        char firstChar;
        char lastChar;
        String exampleString = "Hey, this thing is a String";

        firstChar = exampleString.charAt(0);
        lastChar = exampleString.charAt(exampleString.length()-1);

        System.out.println("The first character is " + firstChar + " and the last is " + lastChar);
    }
}
Reading in Strings and Chars

- Use Scanner object to read in strings using `next()` and `nextLine()` methods

- to get a char, use `nextLine()`, then `charAt(0)` on the string
Comparing Strings

- `equals(s1)`
  - Returns true if string is equal to `s1`

- `equalsIgnoreCase(s1)`
  - Same as above, but case insensitive

- `compareTo(s1)`
  - Compare strings and return an int >0, 0, or <0

- `compareToIgnoreCase(s1)`
  - Same as above, but case insensitive

- `startsWith(prefix)`
  - Returns true if string starts with prefix

- `endsWith(suffix)`
  - Returns true if string ends with suffix

- `contains(s1)`
  - Returns true if `s1` in inside string
Testing if strings are the same

```java
import java.util.*;

public class GuessAWord {

  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    String secretWord = "secret";
    String userGuess;

    System.out.println("Try to guess the word I'm thinking");
    userGuess = input.nextLine();

    if (secretWord == userGuess) {
      System.out.println("You guessed it!");
    } else {
      System.out.println("Sorry, that's not it.");
    }
  }
}
```
Testing if strings are the same

```java
import java.util.*;

public class GuessAWord {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String secretWord = "secret";
        String userGuess;

        System.out.println("Try to guess the word I'm thinking");
        userGuess = input.nextLine();

        if (secretWord.equals(userGuess)) {
            System.out.println("You guessed it!");
        } else {
            System.out.println("Sorry, that's not it.");
        }
    }
}
```
Substrings

• `substring(beginIndex)` -> Returns section of the string starting at `beginIndex`, to the end of the string

• `substring(beginIndex, endIndex)` -> Returns section of the string starting at `beginIndex`, to `endIndex - 1`
Substrings

```java
class GetSubstring {
    public static void main(String[] args) {
        String exampleString = "Hey, this thing is a String";
        String sub;

        sub = exampleString.substring(21);
        System.out.println(sub);

        sub = exampleString.substring(10, 15);
        System.out.println(sub);
    }
}
```
Searching Strings

- `indexOf(ch)`, `indexOf(s)`
- `indexOf(ch, fromIndex)`, `indexOf(s, fromIndex)`
- `lastIndexOf(ch)`, `lastIndexOf(s)`
- `lastIndexOf(ch, fromIndex)`, `lastIndexOf(s, fromIndex)`
public class SearchString {
    public static void main(String[] args) {
        String exampleString = "Hey, this thing is a String";

        System.out.println(exampleString.indexOf("this"));
        // System.out.println(exampleString.indexOf("th"));
        // System.out.println(exampleString.indexOf("a"));
        // System.out.println(exampleString.indexOf("z"));
    }
}
Strings to numbers, numbers to strings

```java
int numInt = Integer.parseInt(intString);
double numDouble = Double.parseDouble(doubleString);

String s = num + "";
```
Challenge: Small data parsing at AMNH
Challenge: Small data parsing at AMNH

• For an interactive media piece in our last show, I needed to parse a list of foods in the following format:

• [G,B]-[FoodName]

• ex. G-Broccoli, B-Chips
Challenge: Small data parsing at AMNH

• Write a program that will parse one line of that file:
  • G-Carrots

• And report to the user what the food was, and whether it’s a good food, or a bad food
public class ParseFood {
    public static void main(String[] args) {
        String foodText = "G-Carrots";
        int dashIndex;
        String goodOrBad;
        String foodName;

        dashIndex = foodText.indexOf('-');
        goodOrBad = foodText.substring(0,dashIndex);

        System.out.print(foodText.substring(dashIndex+1) + " are ");
        if (goodOrBad.equals("G")){
            System.out.println("good food!");
        } else {
            System.out.println("bad food!");
        }
    }
}