Hello, World

Spec FAQ Project Submit

Expand All Collapse All

General

- **Which Java programming environment should I use?**

  We recommend our customized IntelliJ programming environment for Mac OS X, Windows, and Linux. If you followed our step-by-step instructions, then everything should be configured and ready to go.

  If you prefer to use another environment (such as Eclipse or Sublime), that's fine. On the first three assignments, you'll need to enter command-line arguments. After that, you will need to add our textbook libraries stdlib.jar to your classpath and access to the command line (for redirection of standard input and standard output).

- **Which version of Java must I use?**

  Any version of Java 8, Java 9, Java 10, or Java 11 should be fine. Our autograder uses Java 8, so, if you use a more recent version, be sure to use only Java 8 features. We have configured the *IntelliJ* projects to enforce this constraint.

- **What's the Project link?**

  It contains an *IntelliJ* project that you can use to develop your program. It may also contain supplemental data files (which you will use even if you do not use *IntelliJ*).

- **Will I receive a deduction if I don't adhere to the course rules for formatting and commenting my code?**

  The autograder (and *IntelliJ*) provide style feedback to help you become a better programmer. The autograder, however, does not typically deduct for stylistic flaws.

- **Can I use Java constructs (such as loops and conditionals) before they are introduced in the course?**

  No. Do not use Java features until we have introduced them in the course.

- **What should my program do if the user specifies an invalid input?**

  In general, professional programmers must write code to carefully check and validate any user input. However, in this course, you may assume that any user inputs are in the specified format. For example, you may assume that the command-line arguments to `CMYKtoRGB` are four real numbers between 0.0 and 1.0 (and not, say, two arbitrary strings).

- **How do I create a zip file for submission to Coursera?**

  Here are three approaches:

  - *Mac OS X.*
    - Select the required files in the *Finder*.
    - Right-click and select *Compress 5 Items*.
    - Rename the resulting file to `hello.zip`.

  - *Windows.*
    - Select the required files in Windows Explorer.
    - Right-click and select *Send to -> Compressed (zipped) folder*.
    - Rename the resulting file to `hello` (the .zip extension is automatic).

  - *Command line (Linux or Mac OS X or Windows Git Bash).*
    - Change to the directory containing the required .java files.
    - Execute the command: `zip hello.zip *.java`.

HelloWorld

- **I don't understand all of the jargon in `HelloWorld.java`. Should I continue with the course?**

  Don't worry. Java needs a lot of boilerplate code to get started. You'll learn the meaning of `public`, `static`, `void`, `class`, and `main` during this course.

HelloGoodbye

- **I get the error `Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0`. What does this mean?**

  Did you forget to type a command-line argument when you executed the program?

RightTriangle

- **How do I get Java to print `true` or `false` without an if–else statement?**

  If `b` is a `boolean` variable, then `System.out.println(b)` prints `true` or `false`, according to its value.

- **How should I compute a^2 ?**

Use the expression `a * a`. In Java, the `^` operator means *exclusive or*, not exponentiation. You could use `Math.pow(a, 2)`; however, that converts `a` to a `double` before squaring, which is considered poor style.

▼ Should I write one single complex boolean expression to check for valid right triangles?

No. That would make your program difficult to read, debug, and maintain. Instead, break up a complicated expression into individual components. For example, define a boolean variable `arePositive` to be `true` if a, b, and c are all positive, and `false` otherwise. Then, you can use this boolean variable (and others) to define a boolean variable `isRightTriangle` that checks for valid right triangles.

▼ Should I worry about producing intermediate integers that are too large to fit in an `int`?

Good question. In general, such *arithmetic overflow* can be the source of subtle bugs. If you succeed in handling it in this program, you'll be rewarded by passing a bonus test.

GreatCircle

▼ How do I write and debug a program that involves a complicated formula?

First, decompose a formula into smaller pieces and store each piece in a separate variable. Next, pick input values for which computing the pieces by hand are relatively easy. Finally, check that each piece matches the value you expected. For example, the great-circle distance between (60°, 15°) and (120°, 105°) is approximately 4,604.53989 kilometers.

$$\boxed{\begin{aligned}(x_1, y_1) &= (\ 60^\circ,\ 15^\circ) &&= (\tfrac{\pi}{3}\text{ rad},\ \tfrac{\pi}{12}\text{ rad})\\ (x_2, y_2) &= (120^\circ,\ 105^\circ) &&= (\tfrac{2\pi}{3}\text{ rad},\ \tfrac{7\pi}{12}\text{ rad})\end{aligned}}$$

$$\overset{\overset{4604.53989\ldots}{\downarrow}}{\mathit{distance}} = \underset{\underset{12742.0}{\uparrow}}{\boxed{2r}} \cdot \overset{\overset{\arcsin\sqrt{\frac{1}{8}} = 0.361367\ldots}{\downarrow}}{\boxed{\arcsin\left(\sqrt{\underset{\underset{1/4}{\uparrow}}{\boxed{\sin^2\left(\frac{x_2 - x_1}{2}\right)}} + \underset{\underset{-1/4}{\uparrow}}{\boxed{\cos x_1 \cdot \cos x_2}} \cdot \underset{\underset{1/2}{\uparrow}}{\boxed{\sin^2\left(\frac{y_2 - y_1}{2}\right)}}}\right)}}$$

▼ Which `Math` library functions should I use?

Use whichever functions from Java's Math library ☕ that you need. The functions `Math.toRadians()`, `Math.sqrt()`, `Math.pow()`, `Math.sin()`, `Math.cos()`, and `Math.asin()` are particular relevant here.

▼ My output matches the sample output in the assignment specification, except in the very last digit or two. What causes these tiny discrepancies?

Computers work with limited precision, so algebraically equivalent expressions can produce slightly different results. Typically, the autograder ignores such tiny discrepancies.

▼ Can I use a different formula to compute the great-circle distance, such as one based on the spherical law of cosines?

No. Please use the formula provided. There are some differences due to floating-point precision, especially for antipodal points or points that are very close. The Haversine formula is more resilient to floating-point precision issues than the one based on the spherical law of cosines.

CMYKtoRGB

▼ How do I round a floating-point number to the nearest `int` value?

Use `Math.round(x)` ☕ to round `x` to the nearest `long` value; then cast the result to an `int`. If there is a tie, `Math.round()` will round toward positive infinity.

▼ I've seen a different formula for converting from CMYK to RGB. What's the difference?

Technically, you can't convert from CMYK to RGB because the mapping depends on the underlying *color spaces* (such as sRGB and SWOP). Color spaces map RGB or CMYK values to actual colors.