Command-line programs tutorial

Command-line programs interact with a user by prompting a user for input, doing something with that input, and then displaying the result.

In this tutorial, you'll write a command-line program that converts kilometers to miles. Here's an example of what the program's output will look like when it's finished:

```
Enter a kilometer value to start at: 0
Enter a kilometer value to end with: 20
How many should it increment by: 5

Going from 0km to 20km in increments of 5km.

0km is 0.0mi.
5km is 3.106855mi.
10km is 6.21371mi.
15km is 9.320565mi.
20km is 12.42742mi.
```

To get started, import this project into IntelliJ. You'll write your code in the `src/main/java/com/techelevator/KilometerConverter.java` file.

Step One: Add an empty `main` method

To run a Java class, it must have a `main` method, so begin by adding one to the `KilometerConverter` class:

```
public class KilometerConverter {

    public static void main(String[] args) {

    }

}
```

Step Two: Create a `Scanner` to read from `System.in`

You'll need an instance of the `Scanner` class to read what the user types in, so create one by calling the constructor and storing a reference to the new object in a variable named `input`:

```
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

}
```

Step Three: Ask the user for three values

Ask the user for the starting kilometer value by following these three steps:

1. Display a prompt so the user knows what to enter.
2. Use the `Scanner` to read the `String` the user types.
3. Convert the `String` to an `int` and store it in a variable.

Add these lines of code:

```
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.print("Enter a kilometer value to start at: ");
    String value = input.nextLine();
    int kilometerStart = Integer.parseInt(value);

}
```

Repeat those same three steps two more times to ask the user for the ending kilometer value and the amount to increment by:

```
System.out.print("Enter a kilometer value to end with: ");
value = input.nextLine();
int kilometerEnd = Integer.parseInt(value);

System.out.print("How many should it increment by: ");
value = input.nextLine();
int incrementBy = Integer.parseInt(value);
```

Next, display the values the user has provided like this:

```
System.out.println("Going from " + kilometerStart + "km to " + kilometerEnd +
        "km in increments of " + incrementBy + "km.");
```

If you run the program at this point, and enter the values 0, 20, and 5, you'll see this output:

```
Enter a kilometer value to start at: 0
Enter a kilometer value to end with: 20
How many should it increment by: 5
Going from 0km to 20km in increments of 5km.
```

Step Four: Convert the entered values and display the result

Printing out each of the conversion values goes back to what you've learned previously: writing arithmetic logic. Writing repetitive statements and incrementing by the same amount until a value is reached sounds like a for loop.

Remember the structure of the for loop:

```
for (initializer; condition; increment/decrement) {
    statement or block of code to run when condition is true
}
```

For loops can start anywhere and they can increase, or decrease, by however much is necessary. For this conversion program, write a for loop that goes from your start to finish variables:

```java
for (int km = kilometerStart; km <= kilometerEnd; km += incrementBy) {

}
```

This for loop starts the variable `km` at whatever value `kilometerStart` holds. It runs as long as `km <= kilometerEnd`. After each iteration, `km` increments by whatever value `incrementBy` holds.

Inside the loop, convert the number of kilometers stored in `km` to the equivalent number of miles by multiplying by `0.621371`:

```java
for (int km = kilometerStart; km <= kilometerEnd; km += incrementBy) {
    double miles = km * 0.621371;

}
```

To display the result of the conversion, print it out by adding one more line inside the for loop:

```java
for (int km = kilometerStart; km <= kilometerEnd; km += incrementBy) {
    double miles = km * 0.621371;
    System.out.println(km + "km is " + miles + "mi.");
}
```

If you run the program again, you'll see output that resembles the example at the beginning of this file.

Step Five: Move the calculation to a separate method

Rather than keeping the details of the kilometers to miles calculation inside the for loop inside the `main` method, move it to its own method.

After the `main` method, add the following code. While you're at it, you can also make sure that `0.621371` isn't a magic number like this:

```java
public static double kilometersToMiles(int kilometers) {
    final double MILES_PER_KILOMETER = 0.621371;
    return kilometers * MILES_PER_KILOMETER;
}
```

Finally, update the for loop in the main method to call the new `kilometersToMiles` method. Your `main` method should look like this:

```java
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.print("Enter a kilometer value to start at: ");
    String value = input.nextLine();
    int kilometerStart = Integer.parseInt(value);

    System.out.print("Enter a kilometer value to end with: ");
    value = input.nextLine();
    int kilometerEnd = Integer.parseInt(value);

    System.out.print("How many should it increment by: ");
    value = input.nextLine();
    int incrementBy = Integer.parseInt(value);

    System.out.println("Going from " + kilometerStart + "km to " + kilometerEnd +
        "km in increments of " + incrementBy + "km.");

    for (int km = kilometerStart; km <= kilometerEnd; km += incrementBy) {
        double miles = kilometersToMiles(km);
        System.out.println(km + "km is " + miles + "mi.");
    }
}
```

This may seem like more work, but practicing this approach encourages good coding habits and can make less work for you later on. Some of the benefits are:

1. Testable code - Isolating code into small methods allows you to write tests that validate that the formula calculates correctly.
2. Readable code - Over time, code becomes more readable when it does less work. The for loop doesn't have to do much with `km` except call a function that converts it into miles.

Next steps

If you want to enhance the program's functionality and continue with experimenting and adding features, consider trying these challenges:

- Run the program infinitely until the user indicates they want to exit.
- Add other conversion units, like miles to kilometers, feet to inches, or kilometers to yards.
- Validate user input, ensuring they enter positive numbers and that it goes from small to large.

- What happens if the user enters something that isn't a number?