Homework 6

CS 1331

Due Friday, October 25th at 11:59PM

1 Introduction

Hello everybody! Hope you guys had fun with abstract classes! This homework will focus on interfaces and error catching. For this homework, you will be writing a dynamic unit converter.

2 Problem Description

You are sick and tired of dealing with all these different units of length. Converting between miles, feet, kilometers, meters, and other measurements is tedious and time-consuming. Eventually you become so fed up that you want to create a calculator that can convert to different types of units, while also allowing the user to create more types of units during runtime.

Your job is to write the classes for Mile, Foot, and three other types of length measurements. These classes must implement Convertible, an interface that defines a list of methods that your converter will need to use. Users should also be able to create a new unit at runtime through the use of a class that can be customized by the user. You will also have to create three exceptions that are thrown if certain conditions are met.

Finally, you must write the driver class Main. Main will loop through, first asking the user what they would like to do. The user has a choice to: receive a cool fact about the unit, convert a unit to meters, convert a unit to a different unit, and add a new unit to the calculator. Your driver should be robust enough to correctly handle errors if the user inputs an invalid data type.

Here is an example of what your program should output (not including three custom Convertibles)!

3 Solution Output

```
$ java Main
Welcome to the converter!
What would you like to do?
1. Learn cool facts
```

```
2. Convert to meters
3. Convert to another unit
4. Add a temporary unit to converter
5. Exit this program

4
We offer these following units:
Mile
Foot

What is the name of the new unit?
Light Year
What is a cool fact about this unit?
It is very big.
How many meters are in 1 of this unit?
10000000000000000000000000000000000000

==================================================

What would you like to do?
1. Learn cool facts
2. Convert to meters
3. Convert to another unit
4. Add a temporary unit to converter
5. Exit this program

4
We offer these following units:
Mile
Foot
Light Year

What is the name of the new unit?
foot
What is a cool fact about this unit?
2
How many meters are in 1 of this unit?
7
DuplicateUnitException: We already support this operation.

==================================================

What would you like to do?
1. Learn cool facts
2. Convert to meters
3. Convert to another unit
4. Add a temporary unit to converter
5. Exit this program

1
We offer these following units:
Mile
Foot
Light Year

What unit would you like to learn a cool fact about?
Light Year
It is very big.

==================================================

What would you like to do?
1. Learn cool facts
2. Convert to meters
3. Convert to another unit
4. Add a temporary unit to converter
5. Exit this program
```

```
2
We offer these following units:
Mile
Foot
Light Year

What unit are you converting to meters?
Light Year
How many of these are you converting to meters?
1

Light Year: 1.0
The total number of meters is: 1.0E37

=====================================================

What would you like to do?
1. Learn cool facts
2. Convert to meters
3. Convert to another unit
4. Add a temporary unit to converter
5. Exit this program

2
We offer these following units:
Mile
Foot
Light Year

What unit are you converting to meters?
Furlong
How many of these are you converting to meters?
2
UnsupportedUnitException: We do not support this operation!

=====================================================

What would you like to do?
1. Learn cool facts
2. Convert to meters
3. Convert to another unit
4. Add a temporary unit to converter
5. Exit this program

4
We offer these following units:
Mile
Foot
Light Year

What is the name of the new unit?
Inch
What is a cool fact about the unit?
It is very small.
How many meters are in 1 of this unit?
.05
=====================================================

What would you like to do?
1. Learn cool facts
2. Convert to meters
3. Convert to another unit
4. Add a temporary unit to converter
5. Exit this program

5
```

4 Class Outline

Convertible classes:

- You are given the interface `Convertible`
- Need to fill out all appropriate methods in `Convertible`:
- `getConversionRate(), convertToMeters(), specialInformation(), getName()` `convertToDifferentMeasurement()`
- Should have one constructor that initializes all class-specific data
- UserCreatedConvertible
  - Should throw `UnsupportedRateException` if the rate is negative, stopping creation of the object.
  - Constructor should take in important data relevant to the new user created object.

Exceptions:

- You should write three new exceptions: `DuplicateUnitException, UnsupportedRateExcept` `UnsupportedUnitException.`
- All of these Exceptions should extend `RuntimeException` (meaning they are unchecked exceptions).
- These new exceptions should have appropriate error messages.

MetricConverter:

- Contains an array of `Convertibles` that signifies the units that the Converter supports.
- The constructor should instantiate the array and put the Convertible objects that are instantiated into the array.
- Any Exceptions that you throw will be handled in the Driver.
- The Converter needs 4 main functions:
  - `addUnit(double rate, String name, String fact)`
    - Adds a `UserCreatedConvertible` to the array.
    - If a `Convertible` of the same name already exists, throw a `DuplicateUnitException.`
    - If the user types in a negative rate, throw an `UnsupportedRateException.`
    - If the array is full, create a new array with twice the size, then copy the old elements of the array into the new one before adding the new UserCreatedConvertible.

- `convertToMeters(String name, double x)`
  - Uses the corresponding `Convertible` to return the converted value
  - Throws an `UnsupportedUnitException` if the unit does not exist.
- `convertToDifferentUnit(String baseUnit, String targetUnit, double x)`
  - Returns the conversion between the baseUnit and targetUnit
  - Throws an `UnsupportedUnitException` if either unit does not exist.
- `getCoolFact(String name)`
  - Returns the cool fact about the corresponding unit.
  - Throws an `UnsupportedUnitException` if unit does not exist.

Driver:

- The driver will create a `MetricConverter` and allow for user input
- The driver will keep looping until the user tells it to stop
  - The loop should have 5 options depending on user input:
    - Create a new unit
    - Convert to meters
    - Convert to another unit
    - Get a cool fact about a unit
    - Exit the program
  - If the user inputs the wrong data type, you should catch the error and restart the loop.
  - If the code encounters any of the excetpions that you wrote, print an appropriate message and restart the loop.

4.1 Tips

- `Convertible` is an interface that we give you. You must override all methods in the interface to correctly implement it.
- To reduce reusing code, it is possible for an abstract class to implement an interface, but this is not mandatory.
- Helper functions can make operations of `MetricConverter` much easier.
- Dont forget to import `InputMismatchException` located in `java.util.InputMismatchexcep`
  - This is the `Exception` thrown if the user inputs an invalid data type with `Scanner`.
- You cannot modify `Convertible`.

4.2 Program Requirements

In summary, your program should...

1. Create 6 classes implementing `Convertible`. 2 of them will be `Foot` and `Mile`, and one of them will be `UserCreatedConvertible`.
2. Write a `MetricConverter` that allows the link between different `Convertibles`.
3. Write a `Driver` named `Driver.java` that follows the specifications above.
4. Javadoc all methods and classes (except main())
5. Pass the Checkstyle jar.

4.3 Turn-in Procedure

Turn in the following files to T-Square. When youre ready, double-check that you have submitted and not just saved a draft. Due to the sheer amount of files you are submitting, you are allowed to submit they in a .zip archive if you wish (please do not use .rar or other compressed formats if at all possible).

- `Mile.java`
- `Foot.java`
- `UserCreatedConvertible.java`
- `MetricConverter.java`
- `Driver.java`
- `UnsupportedUnitException.java`
- `UnsupportedRateException.java`
- `DuplicateUnitException.java`
- `Convertible.java`
- Three of your own classes implementing `Convertible`.

5 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight Friday. Do not wait until the last minute!