| | |
|---|---|
| What to Submit | Commit your code to Github as a project named **`lab11`**. |
| Starting code | There is starter code for the project on Github. Download a "zip" file of the starter code and use it to initialize your project. |

## Problem 1. Eliminate Duplicate Code in TaskTimer

The starter code contains `TaskTimer.java` in the tasktimer package.

TaskTimer computes how long it takes to perform some different tasks. But there is lots of duplicate code. Restructure the code to eliminate duplication. Make these improvements:

1.1 Several tasks contains the same code for creating an InputStream to read the wordfile.txt. Move that code to a separate class named Dictionary.java and create a static method getWordsAsStream() that returns an InputStream to read from the file. The tasks will call Dictionary.getWordsAsStream when they need the word list.

This uses the *Single Responsibility Principle*. Managing the words file is a separate responsibility from running the tests, so it should be done by a separate class.

1.2 Every task contain some code that is always performed the same.

The code for computing the time used by each task is always the same. The task varies. Apply the principle "*Separate the part that varies from the part that remains the same"*. Create a separate class for each task and make the classes implement *Runnable*. (See diagram below.)

Read "*Let's Remove Duplicate Code*" by Thai Pangsakulyanont: **`https://goo.gl/TGiUqC`**

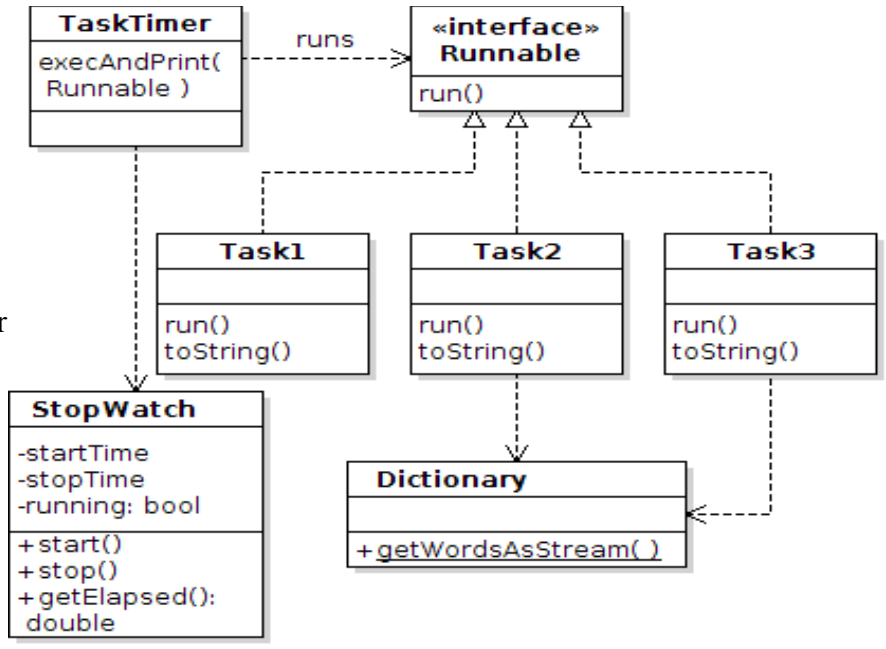
In TaskTimer you should create a method execAndPrint method that runs the task (*Runnable*) and then prints how much time was used:

*execAndPrint( Runnable task )*
*print description of task using toString*
*compute start time*
*run the task*
*compute stop time*
*print elapsed time*

1.3 In each task class, use the constructor to initialize the task, and write a toString() that describes the task. The TaskTimer will invoke toString() of the task.

1.4 Run the code. It should work the same as the original code, but not it is simpler and clearer.

1.5 *Encapsulate the job of computing elapsed time:* Create a separate StopWatch class, with the methods should in the diagram. The TaskTimer execAndPrint should use StopWatch to compute elapsed time.



## Problem 2. Using Lambdas

In the lab11 student package is code for reading a list of students from the Registrar.
We want to print reminders for when a student is having a birthday soon.
The StudentApp class has simple code for this:

```
public class StudentApp {

public void filterAndPrint( List<Student> students, int month ) {
   for(Student student : students ) {
      if (student.getBirthday().getMonthValue() == month)
                  System.out.println( student );
   }
}
```

Your task is to make this code more flexible and to write Lambdas for common tasks. You will optionally modify the code to use streams instead of a loop.

2.1 Modify filterAndPrint to accept a Predicate (java.util.function.Predicate) for selecting students by month. The new method signature should be:

```
filterAndPrint( List<Student> students, Predicate<Student> filter );
```

Use the Predicate (`filter`) in the "if" statement of filterAndPrint. When the Predicate (`filter`) is true, print the student.

2.2 In the main() method, define a Predicate<Student> that is true if the student's birthday is this month (**not** 5). Note that LocalDate.getMonthValue() returns month = 1 for Jan, 2 for Feb, etc (unlike Calendar). Also write your Predicate code here:

_____

_____

2.3 Modify filterAndPrint so that instead of calling System.out.println( student ) that it accepts a Consumer for the action to perform when a Student matches the predicate. The new method should look like this:

```
filterAndPrint( List<Student> students, Predicate<Student> filter,
                Consumer<Student> action ) {
   . . . // code omitted
   if ( filter is true ) action.accept( student );
```

2.4 In the main() method, define a Consumer<Student> using Lambda expression that prints a nicely formatted reminder of the student's birthday. The reminder should look like this:

```
Fatalai Jon will have birthday on 20 May.
```

Write your Consumer code here:

_____

_____

_____

2.5 In the main method, sort the students by first name. Write a *java.util.Comparator* as a Lambda expression. Note that Comparator requires one method (using Student as the type parameter):

public int compare(Student a, Student b)

return < 0 if **a** is before **b** in sort order
return = 0 if **a** and **b** have same sort order
return > 0 if **a** is after **b** in sort order

Use your Comparator as parameter to java.util.Collections.sort().

Your code should look like this:

```
Comparator<Student> byName = (a, b) -> ____________________________
```

2.6 (Optional) Can you write a Comparator to sort students by *birthday*, ignoring the year? That way, filterAndPrint will show students ordered by birthday. Your lambda code should look like:

```
Comparator<Student> byBirthday = (a, b) -> ________________________
```

2.7 (Optional) In the filterAndPrint method, replace the for loop with a stream, using list.stream(). Filter and sort the students all on one line (don't sort the list in main). Define the byBirthday comparator in filterAndPrint.

Your code will look something like this:

```
list.stream().filter( predicate ).sorted( byBirthday ).forEach(consumer)
```

2.8 (Optional) We really want to know when a birthday is coming *soon*. Can you write a Predicate to test if a student's birthday is in the *next 2 weeks?*

## Problem 3: Using LocalDate

*Please do this problem individually.*

*Do not ask your friends for help and do not copy someone else's code.*

The student's birthday is an object of type LocalDate.

In the student class, the setBirthday(String date) parses a String of the form "*day/month/year*" (for example: "25/12/1998") to create a LocalDate object.

It uses complex and ugly logic. Rewrite this method so that LocalDate will parse the string itself.

3.1 Look at the *factory methods* for the LocalDate class, using the Java API. Can you find a method that will parse a String and create a LocalDate object?

3.2 Read the Javadoc for DateTimeFormatter and the static ofPattern(String) method. This method creates DateTimeFormatter objects (a *simple factory method*). How can you use this method to create a DateTimeFormatter for dates of the form ("day/month/year").