P04 Wardrobe Manager 2.0

Overview

The wardrobe is back, and it's New and Improved!

In P01, you maintained the clothing list by using an oversize array as an argument to some utility methods, which trusted that the array was being correctly maintained according to oversize array protocols – compact, accurate size variable.

Now that we've unlocked Object Oriented Design, we're going to ***encapsulate*** all of the wardrobe information in a class (and all of the clothing information in another class!) so we can be absolutely sure that it's being maintained properly. If someone tries to do something they're not allowed to do with the wardrobe, well, that's what exceptions are for!

Additionally, it's useful to keep a list of clothing even when the program isn't actively running, so this version of the Wardrobe Manager will include the ability to save wardrobe contents to a file (and read in an existing file – with some proofreading, of course).

Grading Rubric

5 points	**Pre-assignment Quiz**: accessible through Canvas until 11:59PM on **02/25**.
+5%	**Bonus Points**: students whose ***final*** submission to Gradescope is before **5:00 PM Central Time** on **WED 02/28** *and* who pass ALL immediate tests will receive an additional 2.5 points toward this assignment, **up to a maximum total of 50 points**.
20 points	**Immediate Automated Tests**: accessible by submission to Gradescope. You will receive feedback from these tests *before* the submission deadline and may make changes to your code in order to pass these tests. Passing all immediate automated tests does **not** guarantee full credit for the assignment.
15 points	**Additional Automated Tests**: these will also run on submission to Gradescope, but you will not receive feedback from these tests until after the submission deadline.
10 points	**Manual Grading Feedback**: TAs or graders will manually review your code, focusing on algorithms, use of programming constructs, and style/readability.
50 points	**MAXIMUM TOTAL SCORE**

Learning Objectives

After completing this assignment, you should be able to:

- **Implement** a simple object with basic accessor and mutator methods
- **Communicate** error states using exceptions and appropriate messages, and **explain** how to appropriately handle thrown exceptions.
- **Explain** the different requirements of using a static vs. a non-static method
- **Incorporate** exception handling into your tester methods so that they can survive unusual circumstances and receive communication about error states from other methods without crashing

Additional Assignment Requirements and Notes

Keep in mind:

- Pair programming is **ALLOWED** for this assignment, BUT you must have registered your partnership before this specification was released. If you did not do so, you must complete this assignment individually.

- The ONLY external libraries you may use in your program are:
  Clothing.java `- java.time.LocalDate`
  Wardrobe.java `- java.util.{NoSuchElementException, Scanner},`
  `java.text.ParseException, java.time.LocalDate,`
  `java.io.{File, FileNotFoundException, FileWriter, PrintWriter,`
  `IOException}`
  Tester.java `- java.util.Arrays, java.time.LocalDate,` any relevant exceptions

- Use of *any* other packages (outside of java.lang) is NOT permitted.

- You are allowed to define any **local** variables you may need to implement the methods in this specification (inside methods). You are NOT allowed to define any additional instance or static variables or constants beyond those specified in the write-up.

- You are allowed to define additional **private** helper methods.

- Only the `WardrobeManagerTester` class may contain a main method.

- All classes and methods must have their own Javadoc-style method header comments in accordance with the CS 300 Course Style Guide.

- Any source code provided in this specification may be included verbatim in your program without attribution.

- All other sources must be cited explicitly in your program comments, in accordance with the Appropriate Academic Conduct guidelines.

- Any use of ChatGPT or other large language models ***must be cited*** AND your submission MUST include a file called `log.txt` containing the full transcript of your usage of the tool. Failure to cite or include your logs is considered academic misconduct and will be handled accordingly.

- **Run your program locally before you submit to Gradescope**. If it doesn't work on your computer, *it will not work on Gradescope*.

Need More Help?

Check out the resources available to CS 300 students here:
https://canvas.wisc.edu/courses/398763/pages/resources

CS 300 Assignment Requirements

You are responsible for following the requirements listed on both of these pages on all CS 300 assignments, whether you've read them recently or not. Take a moment to review them if it's been a while:

- Appropriate Academic Conduct, which addresses such questions as:
  - How much can you talk to your classmates?
  - How much can you look up on the internet?
  - How do I cite my sources?
  - and more!

- Course Style Guide, which addresses such questions as:
  - What should my source code look like?
  - How much should I comment?
  - and more!

Getting Started

1. Create a new project in Eclipse, called something like **P04 Exceptional Wardrobe**[1].
   a. Ensure this project uses Java 17. Select "JavaSE-17" under "Use an execution environment JRE" in the New Java Project dialog box.
   b. Do **not** create a project-specific package; use the default package.

2. Download two (2) Java source files from the assignment page on Canvas:
   a. **WardrobeManagerTester.java** (includes a main method)
   b. **WardrobeManager.java** (includes a main method)

[1] Because it now has exceptions, you see.

3. Create two (2) Java source files within that project's src folder:
    a. **Wardrobe.java** (does NOT include a main method)
    b. **Clothing.java** (does NOT include a main method)

1. Wardrobe Manager Tester

The **WardrobeManagerTester.java** provided to you includes all REQUIRED tester methods. Two of them have been implemented in full:

- `testClothingConstructorAndGetters()`
- `testClothingConstructorExceptions()`

This program features some expected behavior that involves throwing exceptions, and these two tester methods demonstrate how you should write testers to validate that kind of behavior.

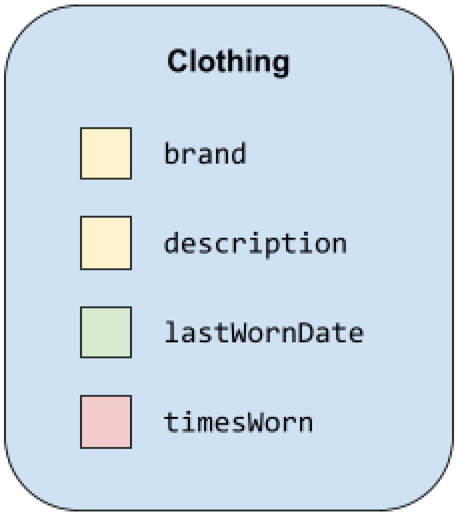
Any tester method with the word **"Exceptions"** in it must *only* feature test cases in which correct implementations should throw a specific exception. Any other tester method tests *only* normal functionality, including edge cases.

As with P01, we recommend writing at least one relevant test before you implement the class/method it would test. We speak from experience here, as debugging 1 thing is WAY easier than debugging 10 things that all depend on each other.

You may add additional tester methods as you see fit, as long as they are private.

2. The `Clothing` data type

Where P01 represented clothing as an array of 3 Strings, this program represents each piece of clothing as its own custom-defined object. In addition to the `brand` (still a String), `description` (also still a String), and `lastWornDate` (now represented as a LocalDate object - see below) we will *also* keep track of how many times a piece of clothing has been worn. Implement the class according to these Javadocs.



As in P01 two pieces of clothing are considered the "same" if the `brand` and `description` match, ignoring case. These are the ONLY data which are considered for equality; number of times worn or last worn date may be different and the two objects will still be considered equal. ***This has important implications for your tester methods***! **Don't** rely only on `equals()` to verify that you have the correct Clothing object.

2.1 An aside: Java's LocalDate object

Java provides an incredibly useful object called LocalDate, which we use to store the `lastWornDate` of each Clothing object. Familiarize yourself with the documentation, as it will be helpful across many parts of this program. Some highlights:

- LocalDates are easy to create using the static method `LocalDate.of(year, month, day)`.
- You can tell if a particular LocalDate is equal to another using the `equals()` method, but conveniently LocalDates also have an `isBefore(LocalDate otherDate)` method if you want to know which date happened earlier.
- Months in LocalDate are 1-12, and the first day of each month is day 1. You can access these values from a LocalDate using the `getMonthValue()` and `getDayOfMonth()` methods.

3. Wardrobe

As in P01 this design uses a compact oversized array to hold all of our Clothing, but instead of relying on the tester class to maintain it, you'll encapsulate the relevant information and methods in an object. Go forth and read the javadocs.

A note about ParseException (which is thrown from `parseClothing()`): It has only a 2-argument constructor. For the purposes of this program, you can set the second argument to be 0 at all times.

The compact, oversize `wardrobe` array should have nulls in elements not being used; for full credit you must REMOVE all data not currently being used. For example, if `wardrobeSize` is 3 and `wardrobe.length` is 7 then the array should be as follows: `{clothing1, clothing2, clothing3, null, null, null, null}`

4. OPTIONAL: WardrobeManager

As a demonstration of why you might want a method to throw exceptions in practice (that is, to inform another class/part of your program that something went wrong) we have provided a driver class (WardrobeManager.java) *in full*. You don't have to add anything to this; it should interface with your Clothing and Wardrobe objects as you've written them!

You can see an example of the driver running successfully in the sampleRun.txt file we've provided on the assignment page.

IMPORTANT: do NOT use the driver as a substitute for writing tests. The sample run does not show/check all possible relevant situations. If you haven't finished your tester file yet, GO DO THAT.

Assignment Submission

Hooray, you've finished this CS 300 programming assignment!

Once you're satisfied with your work, both in terms of adherence to this specification and the academic conduct and style guide requirements, submit your source code through Gradescope.

For full credit, please submit the following files (**source code**, *not* .class files):

- Clothing.java
- Wardrobe.java
- WardrobeManagerTester.java

You do NOT need to submit the WardrobeManager source code; it does not include any of YOUR work.

Additionally, if you used any generative AI at any point during your development, you ***must*** include the full transcript of your interaction in a file called

- log.txt

Your score for this assignment will be based on the submission marked "**active**" prior to the deadline. You may select which submission to mark active at any time, but by default this will be your most recent submission.

Students whose final submission (which must pass ALL immediate tests) is made before 5pm on the Wednesday before the due date will receive an additional 5% bonus toward this assignment. Submissions made after this time are NOT eligible for this bonus, but you may continue to make submissions until 10:00PM Central Time on the due date with no penalty.

Copyright notice