Collections and Generics

Overview

In this lab, you will make use of various Java collection classes in a standalone application. If time permits, you will implement a generic helper method to display different types of items in a collection.

Source folders

Student project:	`StudentCollectionsGenerics`
Solution project:	`SolutionCollectionsGenerics`

Roadmap

There are 5 exercises in this lab, of which the last two exercises are "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Using a simple list
2. Using a `LinkedList`
3. Using a `TreeMap`
4. Implementing a generic method
5. Additional suggestions

Familiarization

Open your student project and take a look at the code in `UsingCollections.java`. There are four methods, with liberal TODO comments indicating where you need to add your code later:

- `main()`
  Calls three methods (listed below), to perform collection-based work.
- `manageFootballTeams()`
  Allows the user to manipulate a list of football teams (i.e. `String`s).
- `manageSalaries()`
  Allows the user to manipulate a list of salaries (i.e. `Double`s).
- `manageEmployees()`
  Allows the user to manipulate a map of `Employee` objects.

You might also want to take a quick look in `Helper.java`, which provides some `static` helper methods to get input from the user.

Exercise 1: Using a simple list

In `manageFootballTeams()`, add code where indicated by the TODO comments, to manipulate a list of `String`s. You can use either an `ArrayList` or a `LinkedList` in this exercise, because both classes implement the necessary list-related behaviour.

Run the application, and make sure all the options work as expected.

Exercise 2: Using a LinkedList

In `manageSalaries()`, add code where indicated by the TODO comments, to manipulate a list of `Double`s. You must use a `LinkedList` in this exercise. Also note that you must declare it as `LinkedList<Double>` rather than as a `LinkedList<double>` - why is this?

In `main()`, uncomment the call to `manageSalaries()`, and then run the application. Make sure all the options work as expected.

Exercise 3: Using a TreeMap

In this exercise, you'll use a `TreeMap` to manage a collection of `Employee` objects. To get started, take a look at the `Employee` class in `Employee.java` and note the following points:

- Each employee has an ID (string), a name, and a salary.
- The constructor initializes a new `Employee` object from the keyboard, for simplicity.
- The `getId()` method returns the employee's ID.
- The `toString()` method returns a textual representation of an `Employee` object.
- The `equals()` method determines whether an `Employee` object "is equal to" another object. This will be useful later, when you need to ascertain whether an `Employee` object is already in the `TreeMap`.

Now switch back to `UsingCollections.java` and locate the `manageEmployees()` method. Add code where indicated by the TODO comments, to manipulate a `TreeMap` of employees. In the `TreeMap`, the keys should be the employee IDs, and the values should be the `Employee` objects themselves.

In `main()`, uncomment the call to `manageEmployees()`, and then run the application. Make sure all the options work as expected.

Exercise 4 (If time permits): Implementing a generic method

Take a closer look at your "display" code in each of the methods you've just written. You should find quite a lot of similarity in each case… With this in mind, refactor your code by implementing a generic helper method in `Helper.java` to display the items (of a given type) in any kind of collection.

Suggestions and requirements:

- Name the method `displayCollection()`.
- The method should be flexible enough to take any kind of collection, containing any type of items.
- In the method, display a message indicating the actual type of collection passed in (e.g. `ArrayList`, `LinkedList`, etc.).
- Then display each item in the collection (along with its type, e.g. `String`, `Double`, etc.)

Exercise 5: Additional suggestions (if time permits)

Consider how you might make use of collections in the previous applications you've written so far during the course.