INTRODUCTION TO PROGRAMMING
DM550, DM857 (Fall 2018)

Exam project: Part II — Deadline: 23h59 on Monday, December 3rd, 2018

Overview

In this part, your task is to develop the low-level classes required to run the simulator: classes `City`, `Individual` and `Event`.

Class `City`

Objects of this class correspond to concrete cities. Each city has a name and two coordinates (both represented as doubles). This class should provide the following methods:

- a constructor taking the city's name and its coordinates as parameters and returning a new city;
- methods `double x()` and `double y()` returning this city's horizontal and vertical coordinates, respectively;
- a method `String name()` returning this city's name;
- a method `double distanceTo(City other)` returning the Euclidean distance from this city to `other`.

Class `Event`

Objects of this class represent events to be simulated. Each event is characterized by a type (mutation, death or reproduction), the time at which it should be simulated, and the individual it affects. This class should provide the following methods:

- a constructor taking the event's type, timestamp and a reference to the individual it affects and returning a new event;
- methods `Individual individual()`, `double time()` and `char type()` returning each of the corresponding data about this event;
- a method `String toString()` returning a textual representation of this event.

Your implementation should encapsulate the concrete representation for the different types of events.

Class `Individual`

Objects of this class represent individuals in the simulation, which are characterized by a path between a set of cities. This class should provide the following methods:

- a constructor that takes as argument a list of cities and creates a new individual whose path is a random permutation of the list of cities;
- a method `City[] path()` that returns a copy of this individual's path;
- a method `double cost()` that returns the cost assigned to this individual's path;
- a method `void mutate()` that performs a mutation of this individual;
- a method `Individual reproduce()` that returns a new `Individual` differing from this one by a mutation.

For the constructor, you need a way to create a random permutation of the given array. One possibility is the following algorithm: starting from the *end* of the array, permute the element at the current index with a randomly chosen one in a position with lower index.

For mutating a path, you choose two random cities in the path and simply switch them. You do not need to worry about the possibility of switching a city with itself.

Some steps of the implementation require you to generate a random integer uniformly distributed in the range $[0, n-1]$. This can be done by a call to method `int getRandomValue(int n)` in class `RandomUtils`.

Expected results

You must hand in classes `City.java`, `Event.java` and `Individual.java` implementing the contracts described above. Your classes should be providers for the remaining classes, which are provided in compiled form. An implementation of class `Simulator.java` is also provided; your classes should be compatible both with your own implementation and with the one given.

You should also hand in a report describing the design of your classes – your choice of attributes, whether you provide any additional getters and setters, which universal methods you override, and any other design choices that you think are important to document. The report will be the basis for the evaluation.

Suggestions

It is recommended that you develop and test the three classes independenly. You should test them by dedicated code, and you should also check that they follow the contract by integrating them with the other given classes.