INTRODUCTION TO PROGRAMMING
DM550, DM857 (Fall 2018)

Exam project: Part I — Deadline: 23h59 on Monday, November 12th, 2018

Overview

In this part, your task is to develop the top-level loop of the simulator. Your program needs to work with and maintain two objects: (a) a queue of the events that need to be simulated, which is an instance of class `EventQueue`, and (b) the current population, containing all individuals that are currently alive, and which is an instance of class `Population`.

The simulation depends on a number of parameters (see below for a detailed list). You should start by interacting with the user to obtain this parameters. The list of cities should be read using the appropriate method in class `CityGenerator`.

You then need to setup the simulation by generating an appropriate amount of `Individual`s and adding them to the population. For each `Individual`, you also need to add three events to the queue of pending events – namely, its next mutation, reproduction and death events. The documentation for class `Event` describes the methods you can use to create these events. To generate the time at which the next event of a given kind will happen is obtained, you need to call method `double getRandomTime(double averageTime)` from the class `RandomTools` with the following parameters:

- for mutation: $(1 - \log(\mathsf{fit}(p))) \times M$
- for reproduction: $(1 - \log(\mathsf{fit}(p))) \times \frac{N}{N_{\max}} \times R$
- for death: $(1 - \log(1 - \mathsf{fit}(p))) \times D$

where $\mathsf{fit}(p)$ is the fitness of the individual's path (check the documentation for class `Population`), N is the current size of the population, and M, R, D and $N_{\max}$ are parameters of the simulation.

In each iteration of the main loop, you select the next event from the queue and simulate it. Each type of event is simulated as follows, where p is the path stored in the individual affected by the event.

Mutation. With probability $(1 - \mathsf{fit}(p))^2$, we switch the order of two random cities in the individual's path. If the mutation occurs, there is a 30% chance that it is immediately followed by a second mutation, and (if this happens) a 15% chance that a third mutation occurs.

Afterwards, we add the next mutation for the mutated individual to the event queue. The average time until the next mutation is $(1-\log(\mathsf{fit}(p)))\times M$, if at least one mutation happened, or $(1-\log(\mathsf{fit}(p)))\times M/10$, otherwise.

Reproduction. We create a new individual, which differs from its parent by a mutation.

The next reproduction event for the individual that reproduced is added to the event queue. The average time until this event is $(1 - \log(\mathsf{fit}(p)))(\frac{N}{N_{\max}}) \times R$. Furthermore, we add mutation, reproduction and death events for the newly created individual. The average time until each of these events is $(1 - \log(\mathsf{fit}(p))) \times M$, $(1 - \log(\mathsf{fit}(p)))(\frac{N}{N_{\max}}) \times R$ and $(1 - \log(1 - \mathsf{fit}(p))) \times D$, respectively.

Death. With probability $1 - \mathsf{fit}(p)^2$ this individual dies, and is removed from the population. If this does not happen, the next death event for the individual is added to the event queue. The average time until this event is $(1 - \log(1 - \mathsf{fit}(p))) \times D$.

Epidemic. When the size of the population exceeds a value $N_{\max}$, there is an epidemic. The 5 fittest individuals always survive an epidemic, while the 5 least fit always die. The remaining ones have a probability $\mathsf{fit}(p)^2$ of surviving.

The actual time at which the next event of a given kind will happen is again obtained by calling method `double getRandomTime(double averageTime)` from the given class `RandomTools` with the parameter given. For events that have a given probability of happening, method `boolean getRandomEvent(double probability)` returns whether they happen or not.

When the simulation terminates, you should print the best path found and its cost.

Parameters and execution mode

The simulation depends on a number of parameters that should be interactively requested from the user:

- the initial size of the population;
- the maximum size of the population $N_{\max}$;
- the mutation interval normalization term M;
- the reproduction interval normalization term R;
- the death interval normalization term D;
- the comfort normalization term ω (used in some methods in class `Population`);
- the total simulation time.

The simulation can be run in four different modes:

1. with a report at every t units of time;
2. with a report after every n simulated events;
3. in verbose mode;
4. in silent mode.

In report mode, you print a message at the specified intervals indicating the current time, the number of simulated events so far, the current size of the population, and information about the best path found so far. In verbose mode, you report on each event you simulate. In silent mode, there are no messages during the simulation (but you still show the final result).

Expected results

You must hand in a class `Simulator.java` implementing the functionality described above. Your class should be a client of the remaining classes, which are provided in compiled form. It should run directly from the command line, and ask for any relevant input using only command-line interaction.

You should also hand in a report describing the algorithm implemented, any design choices that you think are important to document, and examples. The report will be the basis for the evaluation.

Suggestions

It is recommended that you initially develop your program to run always in verbose mode. Afterwards, you can add an interactive block to query the user for the desired mode (and its parameters), and extend your code to report at the appropriate places.

Remember also to take into account the following points.

- The event queue may include events related to individuals that have already died. Remember to check whether the individual is in the population before simulating.

Testing and examples

Here are some example executions. Consider using the values given for the parameters. Then experiment with your own set of cities and try to adjust the parameters to get the best results.

Example execution

```
Size of the initial population: 15
Maximum size of the population: 200
Value of the parameter omega: 0,001
Value of the parameter D: 30
Value of the parameter M: 3
Value of the parameter R: 15
Time to run the simulation: 10
Simulation mode (1: every t units; 2: every n events; 3: verbose; 4: silent): 1
Interval between observations: 3
Enter 1 to use the default list of cities, or 2 to read a custom list from a file.
1
Observation
-----------
Current time: 3.020733797130325
Events simulated: 253
Population size: 128
Best path: E; A; C; I; D; F; L; B; O; M; H; K; N; G; J (cost: 142.6236457859882)

Observation
-----------
Current time: 6.00182354191895
Events simulated: 823
Population size: 149
Best path: G; J; I; N; D; L; C; A; K; F; E; M; H; O; B (cost: 103.01769938053293)

Observation
-----------
Current time: 9.001575368290126
Events simulated: 1448
Population size: 160
Best path: J; N; I; D; K; G; F; E; H; M; L; B; C; O; A (cost: 94.73574733192301)

The best path is: J; N; I; D; K; G; F; E; H; M; L; B; C; O; A (cost: 94.73574733192301)
```

Example execution

```
Size of the initial population: 10
Maximum size of the population: 20
Value of the parameter omega: 0,001
Value of the parameter D: 30
Value of the parameter M: 3
Value of the parameter R: 15
Time to run the simulation: 10
Simulation mode (1: every t units; 2: every n events; 3: verbose; 4: silent): 2
Interval between observations: 100
Enter 1 to use the default list of cities, or 2 to read a custom list from a file.
1
Observation
-----------
Current time: 2.831182575196879
Events simulated: 100
Population size: 20
Best path: B; J; G; M; N; C; K; H; I; L; E; F; O; D; A (cost: 122.47792824886713)

Observation
-----------
Current time: 4.718226016346385
Events simulated: 200
```

```
Population size: 21
Best path: B; J; G; M; N; C; K; H; I; L; E; F; O; D; A (cost: 122.47792824886713)

Observation
-----------
Current time: 6.569363941993142
Events simulated: 300
Population size: 18
Best path: I; G; J; H; M; E; F; K; D; B; N; O; L; C; A (cost: 105.27643667871594)

Observation
-----------
Current time: 8.39405158395362
Events simulated: 400
Population size: 16
Best path: I; B; J; H; M; E; F; K; D; G; N; O; L; C; A (cost: 94.01139876752545)

The best path is: I; B; J; H; M; E; F; K; D; G; N; O; L; C; A (cost: 94.01139876752545)
```

Example execution

```
Size of the initial population: 20
Maximum size of the population: 100
Value of the parameter omega: 0,001
Value of the parameter D: 30
Value of the parameter M: 3
Value of the parameter R: 15
Time to run the simulation: 0,5
Simulation mode (1: every t units; 2: every n events; 3: verbose; 4: silent): 3
Enter 1 to use the default list of cities, or 2 to read a custom list from a file.
1
Simulating: Event type: mutation; time: 0.010715420094257851
Simulating: Event type: mutation; time: 0.03847471052998163
Simulating: Event type: reproduction; time: 0.049400297721669574
Simulating: Event type: reproduction; time: 0.12089797895617278
Simulating: Event type: mutation; time: 0.13785715877508226
Simulating: Event type: mutation; time: 0.16433516925298539
Simulating: Event type: mutation; time: 0.18177181255312852
Simulating: Event type: reproduction; time: 0.2121504396156783
Simulating: Event type: reproduction; time: 0.24418048610279788
Simulating: Event type: mutation; time: 0.2711525098044424
Simulating: Event type: reproduction; time: 0.27207271348913525
Simulating: Event type: mutation; time: 0.2788376210733896
Simulating: Event type: reproduction; time: 0.28791799672494767
Simulating: Event type: mutation; time: 0.3003712756898077
Simulating: Event type: mutation; time: 0.3116556856098624
Simulating: Event type: reproduction; time: 0.312602154933153
Simulating: Event type: reproduction; time: 0.3303489576431477
Simulating: Event type: mutation; time: 0.33563047962989123
Simulating: Event type: mutation; time: 0.39214329491123084
Simulating: Event type: reproduction; time: 0.39851954075746043
Simulating: Event type: mutation; time: 0.42926236824163305
Simulating: Event type: reproduction; time: 0.43140687419355106
Simulating: Event type: reproduction; time: 0.46992855827792923
Simulating: Event type: reproduction; time: 0.4962597549619342
Simulating: Event type: reproduction; time: 0.4966445201435269
The best path is: N; A; J; D; C; F; K; O; B; L; G; E; M; H; I (cost: 113.43904346342686)
```

Example execution

```
Size of the initial population: 15
Maximum size of the population: 1000
Value of the parameter omega: 0,001
Value of the parameter D: 30
Value of the parameter M: 3
Value of the parameter R: 15
Time to run the simulation: 100
Simulation mode (1: every t units; 2: every n events; 3: verbose; 4: silent): 4
Enter 1 to use the default list of cities, or 2 to read a custom list from a file.
1
The best path is: L; G; K; F; E; M; H; O; N; I; A; B; C; J; D (cost: 67.63170811883303)
```