CS 150: Project I - Amusement Park Configuration

C.W. Liew

Version as of: 12:33 Wednesday 14th February, 2018

Due: 11:55pm, Friday March 9, 2018

1 Introduction

Your project is to design a simulation so that we can determine the optimal combination of features for an amusement park. The park will have rides of different types and each ride will have its own attractiveness, length of time and other features. The goal of the program is to allow the designer to answer some questions like those listed in Section 4

2 Project Description

The simulation will have to consider the following:

- For each type of ride:
  - what age group(s) does it appeal to? (adult, young adult, teenage, adolescent, kid, toddler)
  - how much space does it need?
  - how long is a ride?
  - what is the capacity of the ride? (how many people are simultaneously on the ride)
  - how much space is allocated for riders to queue?
- For each group of visitors:
  - type of group - individual, friends, family

    For each group of friends:
    - how many are in the group?
    - what age group are they in (teenage, young adult, adult)?

    For each family:
    - assume there are 2 adults (parents)
    - how many children are in the family?
    - what is the age of each child?

Parameters governing the current simulation are:

- frequency of arrival of groups
- distribution of groups (individual, friends, family)
- number of children
- age distribution of children

Simplifying Assumptions

The assumptions that we make to simplify the program are:

- all visitors pay a single price for each individual
- it takes one time step to load a ride
- the same distribution of visitors will arrive throughout the day
- each person takes up one space
- the cost for the park is the total space taken up by the rides (and the lines)

The information that you will need for the simulation includes:

- the size of the park (number of spaces)
- the frequency of arrival of groups
- the profile of each group

To see how to generate random numbers with a Gaussian (normal) distribution see `http://www.javapractices.com/topic/TopicAction.do?Id=62`.

Rules Governing The Program

The rules underlying the program are:

1. there is no space required for moving between rides as movement between queues is instantaneous,
2. each person will only try each ride either 0 or 1 times,
3. groups will ride together,
4. older people can go on rides for younger people but not vice versa. However, they cannot go on any rides designed for people that are younger than members of the group. For example: if a family consists of 2 adults and 2 teenagers they cannot go on a ride that is designed for adolescents.
5. the age groups that we have (in descending order): adults, young adults (18-22), teenagers (13-18), adolescents (5-12), toddlers (< 5)
6. groups leave the park when either they have exhausted the rides they want, or there is no space for them to join the queue for a ride

2.1 Program Behavior

We are going to simulate the behavior of the amusement park and visitors for some number of time steps. At each time step:

- some number of groups will arrive
- each ride will progress one step; if the end of the ride has arrived the passengers will be unloaded; if the ride discharged on the previous step, it will now load for the next step
- passengers will go from one ride to another in zero time steps
- the group leaves when they have either exhausted all the rides they want, or there is no space for them to join the queue for a ride.

You can assume that the simulation runs for 1000 time units.

2.2 Program Inputs

The following inputs will be through the $args[]$ array in $main()$ in the following order:

- name of the config file
- name of the parameter file

The config file will contain the information described in Section 2. The format of the files is specified below. An example of each file is available on moodle in the section for Project 1.

2.2.1 Configuration File

The configuration file holds information about 3 components - park, ride, people. Each component is described by a name (on one line) followed by key value pairs on subsequent lines. The information about a component ends with a blank line. There is only one park and one people component but there can be multiple ride components. They can all occur in any order.

2.2.2 Parameters File

Similarly the parameters file contains the parameters of the simulation that we will (possibly) change with every run of the program. It specifies how many of each kind of ride that we wish to have in the park. The information is specified as:

```
ride

$<$type of ride$>$
$<$number of rides of this type$>$
```

Each block terminates with a blank line.

Project Constraints

The following constraints apply to the project:

1. The project is to be completed individually. The only person you can consult is the instructor.
2. Each configuration of parameters should be run *at least* 5 times with different random seeds to obtain an ”average” value.
3. You are only to use containers from the Java Collections Framework. The only exception is when you use a method from an API that requires the use of arrays (e.g., main()).

3 Simplifications

You can simplify your project in one (or more) of the following ways:

- the rides are equally attractive to all age groups
- there are only 2 types of rides
- there are only 2 objects of each kind of ride
- the waiting lines for each ride have an unlimited capacity (the park also has unlimited capacity)

Each simplification is considered to be a loss of functionality and the appropriate number of points will be deducted.

4 Report

Your simulation and report should try to answer questions (you should design your own questions) like the following:

1. What is the average wait time for rides?
2. What is the max wait time for rides?
3. What configuration has a ¿ 70% chance of no one leaving early?
4. How much money will each configuration (of rides) make per day?
5. What is the set of features that will maximize the number of people in the park per day?
6. What configurations will make it so that no one will leave early due to a lack of space for rides?

Grading

Your project will be graded on the following criteria (assuming the program compiles and runs):

1. ”the story” (2.5%)
2. class diagrams (2.5%)
3. correctness and functionality of the program (35%)
4. documentation (methods and classes) (10%)
5. unit testing (15%)
6. object oriented design (15%)
7. report (20% - 10% for the draft and 10% for the final)

4.1 Documentation Requirements

The README file should contain instructions as to how to run the program, including where to start.

The java files should have the following:

1. Each class should have a description of its purpose/functionality.
2. Each method should have a header describing the method and its functionality. Set/Get methods need not be documented.
3. Each large block within a method (a large loop/conditional) should have some comments describing the purpose of the block.

4.2 Testing Requirements

Each class should have an associated test class. Each method should be tested for:

- normal case
- boundary conditions
- invalid values in the case where the method is specified to handle them.

5 Submission

The project submission is in 4 parts:

1. (2.5 pts) - due Wednesday, Feb 28: the ”story” of your program
2. (2.5 pts) - due Friday, March 2: class diagrams for your program
3. (85 pts) - due Friday, March 9: the rest including a draft report

4. (10 pts) - due Wednesday, March 21: the final report

s

Your submission for Part 3 will be composed of the following:

1. source files (*.java) that are commented and have javadoc directives
2. test files, one test file per class
3. a README.txt file that contains instructions on how to run your program
4. a draft of the project report (see project report guidelines `www.cs.lafayette.edu/~liew/courses/cs150/writeup-guidelines.pdf`) - 50% of the report grade is assigned to this. The remaining 50% will be given to a final report. The final report will be due several days after the draft is corrected and returned.