EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Modellierung & Simulation I**
**WS 2018/2019**
**Prof. Dr. M. Menth**
**D. Merling**

# Exercise 2

22. Oktober 2018

Abgabe: 29. Oktober 2018, 12:00:00 Uhr

For the following tasks, gap source code packages are provided. Please download the corresponding archive *exc2_sourcecode.zip* in Moodle!

## Problem 2.1: *Statistical Counters*

Statistical counters are used to observe metrics of interest at particular instants within a simulation. They yield statistical measures like mean, variance, and standard deviation about the observed metric.

1. Complete the class implementations of *Counter*, the subclass *DiscreteCounter* for sampling a stochastic process at discrete time instants (see Section 1.5.1 in the course syllabus) and the subclass *ContinuousCounter* for sampling a stochastic process over an entire time interval (see Section 1.5.3.2 in the course syllabus) marked with *TODO* and listed below:

    - *Counter*
        - *increaseSumPowerOne*
        - *increaseSumPowerTwo*
        - *count*
        - *getStdDeviation*
    - *DiscreteCounter*
        - *count*
        - *getMean*
        - *getVariance*
    - *ContinuousCounter*
        - *count*
        - *getMean*
        - *getVariance*

    10 Points

2. Complete the class implementations of *Histogram*, the subclass *DiscreteHistogram* for sampling a distribution at discrete time instants (see Section 1.5.2 in the course syllabus) and the subclass *ContinuousHistogram* for sampling a distribution over an entire time interval (see Section 1.5.4 in the course syllabus) marked with *TODO* and listed below:

   - *Histogram*
     - *getBinNumber*
     - *incrementBin*
   - *DiscreteHistogram*
     - *count*
   - *ContinuousHistogram*
     - *count*

10 Points

## **Problem 2.2:** *Usage of statistical counters*

We extend the simulation by some quantitative results, i.e., we want to simulate the waiting time of customers, the number of customers in the queue, and the server utilization. The implementation should be done in a class *SimulationStudy* in package *study*. Additionally, you have to add Code to the *Simulator* (see *TODOs*) itself. The approach should be done analogous to the sample solution of the tutorial.

1. Briefly explain the functionality and structure of the DES. 5 Points
2. Apply a *DiscreteCounter* and a *DiscreteHistogram* to track the waiting and the service time of a customer! 5 Points
3. Apply a *ContinuousCounter* and a *ContinuousHistogram* to track the queue occupancy and the server utilization over time! 5 Points
4. The interarrival time of customers is $10\ s$. Use a constant service time of $\{9, 10, 11\}\ s$ and a constant simulation time of $T = 10^{\{4,5\}}\ s$ (meaning $6$ different simulation runs) and provide the mean and the coefficient of variation for
   - the waiting time per customer,
   - the service time per customer,
   - the queue occupancy over time, and
   - the server utilization over time! 10 Points
5. Present the histogram for the waiting and the service time per customer in an adequate way and explain the results! 10 Points
6. Conduct the experiment again for a constant service time of $9\ s$ and a constant simulation time of $T = 10^4\ s$. Use now a *DiscreteCounter* instead of a *ContinuousCounter* for tracking the queue occupancy and the server utilization whenever they change in the simulation! Compare the mean values and the coefficient of variation with the results above and explain your observation! 5 Points

## **Problem 2.3:** *Random Variables*

1. Complete the class implementations of the class *StdRNG* in package `simulation.lib.rng` that uses the random number generator `java.util.Random` marked with *TODO* and listed below:
   - *setSeed*
   - *rnd* 5 Points

2. Derive subclasses of *RandVar* in package `simulation.lib.randVars.continuous` for the following distributions:
   - Uniform (see Section 3.2.1 in the course syllabus)
   - Exponential (see Section 3.2.2 in the course syllabus)
   - $k$-Erlang (see Section 3.2.3 in the course syllabus)
   - Hyperexponential $\underline{H_2}$ (see Section 3.2.4 in the course syllabus)

   If an abstract class method does not make sense to be implemented in a particular RandVar class, an *UnsupportedOperationException* should be thrown. 20 Points

3. Implement a class *RandVarTest* in package `study` to apply the statistical counters in order to test your implementation of the *RandVar* subclasses for the following test secenarios:
   - Uniform ($mean = 1$, $c_{var} = \{0.1, 1, 2\}$)
   - Exponential ($mean = 1$, $c_{var} = \{0.1, 1, 2\}$)
   - $k$-Erlang ($mean = 1$, $c_{var} = \{0.1, 1, 2\}$)
   - Hyperexponential $\underline{H_2}$ ($mean = 1$, $c_{var} = \{0.1, 1, 2\}$)

   Which of the proposed distributions cannot be instantiated? Explain why! Use *DiscreteCounter* instances to monitor the statistical properties mean, variance and coefficient of variation of a sample size $n = 10^6$ for all random variables! 15 Points

4. **[Bonus]** Use *DiscreteHistogram* instances with a lower bound of 0 and an upper bound of 10 and 100 intervals to visualize the obtained distributions! Use the *report* class method and a visualization tool of your choise (Excel, LibreOffice, R, matplotlib, ...) to provide appropriate diagrams! *(The solution will be presented and discussed collectively in the following exercise lesson.)* +15 Points

Total: 115 Points