DrugLevelsMain.java

Friday, June 30, 2023 11:24 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```
 1: /** ************************************************************************
 2: *
 3: * Main Class ID:    DrugLevelsMain
 4: * Creation Date:    2023-01-09
 5: * Updated:          2023-02-21
 6: * Version:          v1.5.8
 7: *
 8: * Diagram Versions
 9: *   Causal Inference:   v1.0.2
10: *   Flow Diagram:       v1.2.2
11: *
12: * Programmer: Garrett A. Hughes
13: *
14: * Purpose: Show levels of a prescription drug in a patient's blood serum
15: *          over the course of ingestion and through the elimination period.
16: *
17: * I/O:  User Defined
18: *
19: * NOTE: This class (DrugLevelsMain) has its version number shown above.
20: *
21: *       The title and version number of this program are entered below as
22: *       static members of this class and are printed as part of the header
23: *       at runtime. The program version number changes whenever changes are
24: *       made to any of the classes.
25: *
26: *       If this program implements a model, then the programTitle entered
27: *       below, should be the same as the name of the model.
28: *
29: ************************************************************************** */
30:
31: /* Ruler
32:         1         2         3         4         5         6         7
33: 12456789012345678901234567890123456789012345678901234567890123456789012345678
34: */
35:
36: // from docs.oracle.com/javase/8/docs/api
37: import java.time.LocalDate;
38: import java.time.LocalTime;
39: import java.time.Instant;
40: import java.time.Duration;
41:
42:
43: import xsd.*;
44:
45: public class DrugLevelsMain
46: {
47:     // INSTANCE FIELDS
48:     static String programTitle = "Drug Levels";
49:     static String programCreationDate ="2023-01-02";
50:     static String lastModified = "2023-03-14";
51:     static String version = "v2.2.3";
52:     static String runDescription =
53:         "Stable Version, ran all feasible DT values";
54:     static LocalDate LD;
55:     static LocalTime LT;
56:
57:
```

```
58:     // INITIALIZATION BLOCKS
59:
60:
61:     // CONSTRUCTORS
62:
63:
64:     // METHODS
65:
66:
67:     public static void main(String[] args)
68:     {
69:         System.out.printf("\nProgram Title: %s \nCreated: %s",
70:             programTitle, programCreationDate);
71:         System.out.printf("\nLast Modified: %s", lastModified);
72:         System.out.printf("\nVersion: %s ",version);
73:
74:         System.out.printf("\n\nRun Description: %s", runDescription);
75:         System.out.printf ("\nDate: %s   \nTime: %s\n",
76:             LD.now().toString(),
77:             LT.now().toString());
78:
79:         // **********BEGIN optional timer
80:           Instant start = Instant.now();
81:
82:           // Insert timed code here
83:
84:           // Create objects
85:           EventGenerator evgn = new EventGenerator();
86:           MasterClockRate msclrt = new MasterClockRate();
87:           MasterClock mscl = new MasterClock();
88:           Patient pt = new Patient();
89:           Prescription pr = new Prescription();
90:           IngestionRate inrt = new IngestionRate();
91:           DrugLevel drlv = new DrugLevel();
92:           EliminationRate elrt = new EliminationRate();
93:           BloodSerumConcentration blsrcn = new BloodSerumConcentration();
94:
95:           evgn.updateLinks(msclrt, mscl, pt, pr, inrt, drlv, elrt, blsrcn);
96:
97:           // Begin Simulation
98:           evgn.simulate();
99:
100:          Instant end = Instant.now();
101:          // compute and output duration
102:          Duration elapsedTime = Duration.between(start,end);
103:          long milliSeconds = elapsedTime.toMillis();
104:          long seconds = elapsedTime.toSeconds();
105:          System.out.printf("\nelapsed time = %10d milliseconds",
106:          milliSeconds);
107:          System.out.printf("\nelapsed time = %10d seconds", seconds);
108:        // **********END optional timer
109:
110:
111:    } // END method: main
112:
113: } // END class: DrugLevelsMain
```

EventGenerator.java

Friday, June 30, 2023 9:24 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```
 1: package xsd;;
 2:
 3: /** ************************************************************************
 4: *
 5: * Class ID:         EventGenerator
 6: * Creation Date:    2023-01-09
 7: * Updated:          2023-03-14
 8: * Version:          v2.1.5
 9: *
10: * Programmer:       Garrett A. Hughes
11: *
12: * Purpose:          Initiate events for Drug Levels model
13: *
14: * I/O:
15: *   Input           User defined variables
16: *   Output          User defined at runtime
17: *
18: * NOTE: HOW TIME IS KEPT IN AN XSD MODEL
19: * The passage of time is recorded as ticks on a "MasterClock" object.
20: * Each tick represents the end of one time interval and the beginning of
21: * another. The interval of real time between ticks is kept in the variable
22: * referred to as "DT". You get to choose the value of DT. Note that DT
23: * is the smallest time increment that can be represented in any given model
24: * run. The units of time associated with DT can be of any convenient type:
25: * nanoseconds, hours, years, whatever. DT can be declared as any numerical
26: * primitive: usually a double or an integer.
27: *
28: ************************************************************************** */
29:
30: /* Ruler
31:         1         2         3         4         5         6         7
32: 12456789012345678901234567890123456789012345678901234567890123456789012345678
33: */
34:
35: // from docs.oracle.com/javase/8/docs/api
36: import java.time.Instant;
37: import java.time.Duration;
38: import java.io.*;
39:
40:
41: public class EventGenerator
42: {
43:     // INSTANCE FIELDS
44:     String name = "EventGenerator";
45:
46:     // object identifiers
47:     BloodSerumConcentration blsrcn;
48:     DrugLevel               drlv;
49:     EliminationRate         elrt;
50:     IngestionRate           inrt;
51:     MasterClockRate         msclrt;
52:     MasterClock             mscl;
53:     Patient                 pt;
54:     Prescription            pr;
55:
56:     // simulation runtime parameters
57:     int runtime;            // weeks
```

```
58:     int daysInWeek;     // days/week
59:     int runlength;          // ticks
60:
61:     double hoursInDay;          // hours/day
62:     double runtimeHours;        // hours
63:     double DT;                  // hours
64:
65:     // class variables
66:     int remainder;
67:     int numberTicksPerDay;
68:     int numberTicksPerHour;
69:     int ticksElapsed;           // dimensionless
70:     int elapsedTimeInDays;      // days
71:     int interarrivalTime;       // ticks
72:
73:
74:     double elapsedTimeInHours;  // hours
75:
76:     // ouput file
77:     File file;
78:
79:
80:     // INITIALIZATION BLOCK
81:     {
82:         DT = 1.0; // hours
83:         ticksElapsed = 1;
84:         hoursInDay = 24.0;  // hours/day
85:         runtime    = 4;     // weeks
86:         daysInWeek = 7;     // days/week
87:
88:         // convert runtime in weeks to runtime in hours
89:         runtimeHours =  runtime * daysInWeek * hoursInDay;
90:
91:         // compute runlength in ticks
92:         runlength = (int)Math.round(runtimeHours / DT);
93:
94:         // determine number ticks per day
95:         numberTicksPerDay = (int)Math.round(hoursInDay / DT);
96:
97:         // determine number ticks per hour
98:         numberTicksPerHour =(int)Math.round(1.0 / DT);
99:
100:        // define output file instance
101:        file = new File("DrugLevels_v2.2.3.txt");
102:
103:    }
104:
105:
106:    // CONSTRUCTORS
107:
108:
109:    // METHODS
110:     public void simulate()
111:    {
112:         // view runtime parameters
113:        System.out.println( "\nModel Parameters");
114:        System.out.printf( "    runtime:    %d (weeks)\n", runtime);
```

```
115:         System.out.printf( "    DT:         %f (hours)\n", DT);
116:         System.out.printf( "    runlength:  %d (ticks)\n" , runlength);
117:
118:         // view patient's prescription
119:         pr.showPrescription();
120:
121:         interarrivalTime = numberTicksPerDay/pr.frequency;
122:
123:         System.out.println("\nSIMULATION HAS BEGUN AT ticks = "
124:             + mscl.ticks + "\n");
125:
126:       // open file in which to write results
127:       try
128:         (FileWriter fw = new FileWriter(file);
129:         BufferedWriter bf = new BufferedWriter(fw);
130:         PrintWriter out = new PrintWriter(bf) )
131:       {
132:
133:         // update PrintWriter in DrugLevel object
134:         drlv.updatePrintWriter(out);
135:
136:         // write column headings to file
137:         out.printf("   time           drug    elimination  concentration\n");
138:
139:         do
140:         {
141:             // UPDATE RATES:
142:             inrt.updateIngestionRate();
143:             elrt.updateEliminationRate();
144:
145:
146:             // UPDATE LEVELS:
147:             drlv.updateDrugLevel();
148:
149:
150:             // UPDATE PASSIVE AUXILIARIES
151:             // passive auxiliaries not used to compute rates or levels
152:             blsrcn.updateBloodSerumConcentration();
153:
154:
155:             // OUTPUT SYSTEM STATES
156:             remainder = mscl.ticks % numberTicksPerHour;
157:             if ( remainder == 0)
158:             {
159:                 // compute elapsed time in hours
160:                 elapsedTimeInHours = mscl.ticks / numberTicksPerHour;
161:
162:                 out.printf
163:                 ("%7.0f  %13.6f  %13.6f  %13.6f\n",
164:                     elapsedTimeInHours, drlv.drugLevel, elrt.eliminationRate,
165:                     blsrcn.bloodSerumConcentration);
166:             }
167:
168:
169:             // END ONE TIME INTERVAL BEGIN THE NEXT
170:
171:             // UPDATE MasterClock
```

```
172:             if (mscl.ticks >= runlength) break;
173:             msclrt.updateMasterClockRate(ticksElapsed);
174:             mscl.updateMasterClock();
175:
176:         } while (true); // END do:
177:
178:
179:         System.out.println("\nSIMULATION HAS ENDED AT ticks = "
180:             + mscl.ticks);
181:
182:         System.out.printf("   time           drug        elim_rt
    cncntrtn\n");
183:
184:         System.out.printf(("%7.0f  %13.6f  %13.6f  %13.6f\n"),
185:                     elapsedTimeInHours, drlv.drugLevel, elrt.eliminationRate,
186:                     blsrcn.bloodSerumConcentration);
187:
188:       // close BufferedWriter and PrintWriter
189:             bf.close();
190:             out.close();
191:       }  // END - try
192:
193:       catch (IOException e)
194:       {
195:
196:       }  // END - CATCH
197:
198:     } // END method: simulate
199:
200:
201:     public void reportName()
202:     {
203:         System.out.println("\n" + name);
204:     }
205:
206:
207:     public void updateLinks(MasterClockRate msclrt, MasterClock mscl,
208:         Patient pt, Prescription pr, IngestionRate inrt, DrugLevel drlv,
209:         EliminationRate elrt, BloodSerumConcentration blsrcn)
210:     {
211:         this.msclrt = msclrt;
212:         this.mscl = mscl;
213:         this.pt = pt;
214:         this.pr = pr;
215:         this.inrt = inrt;
216:         this.drlv = drlv;
217:         this.elrt = elrt;
218:         this.blsrcn = blsrcn;
219:
220:         // update links of other objects
221:
222:         msclrt.updateLinks(mscl);
223:         mscl.updateLinks(msclrt);
224:         pt.updateLinks(this, pr, mscl, inrt, elrt);
225:         pr.updateLinks(pt);
226:         inrt.updateLinks(drlv, pr, pt, mscl, this);
227:         drlv.updateLinks(this, mscl, inrt, elrt, pt);
```

```
228:         elrt.updateLinks(this, drlv, mscl, pr);
229:         blsrcn.updateLinks(drlv, pt);
230:
231:     }
232:
233: } // END class: EventGenerator
234:
```

MasterClock.java

Friday, June 30, 2023 11:32 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```
 1: package xsd;
 2:
 3: /** ************************************************************************
 4: *
 5: * Class ID:         MasterClock
 6: * Creation Date:    2023-01-09
 7: * Updated:          2023-02-06
 8: * Version:          v1.2.8
 9: *
10: * Programmer:       Garrett A. Hughes
11: *
12: * Purpose:          Keep track of the time in  "tick" units.
13: *
14: *
15: * I/O:
16: *   Input           clock is updated by updateMasterClock() message
17: *                       which is sent from the MasterClockRate object
18: *
19: *   Output          on demand when receiving getMasterClockTime() message
20: *
21: ************************************************************************** */
22:
23: /* Ruler
24:         1         2         3         4         5         6         7
25: 12456789012345678901234567890123456789012345678901234567890123456789012345678
26: */
27:
28:
29: public class MasterClock
30: {
31:     // INSTANCE FIELDS
32:     String name = "MasterClock";
33:     MasterClockRate msclrt;
34:     int ticks = 0;              // time on Masterclock in ticks
35:
36:
37:     // INITIALIZATION BLOCKS
38:
39:
40:     // CONSTRUCTORS
41:
42:
43:     // METHODS
44:     public void reportName()
45:     {
46:         System.out.println("\n" + name);
47:
48:
49:     }public void updateLinks(MasterClockRate msclrt)
50:    {
51:         this.msclrt = msclrt;
52:
53:    } // END method: shareLinks
54:
55:
56:     public void updateMasterClock()
57:     {
```

```
58:          ticks += msclrt.ticksElapsed;
59:
60:      }
61:
62:      public int getMasterClockTime()
63:      {
64:          return ticks;
65:
66:      }
67:
68:
69: } // END class: MasterClock
70:
```

MasterClockRate.java

Friday, June 30, 2023 11:38 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```
 1: package xsd;
 2:
 3: /** ************************************************************************
 4: *
 5: * Class ID:         MasterClockRate
 6: * Creation Date:    2023-01-09
 7: * Updated:          2023-02-06
 8: * Version:          v1.2.3
 9: *
10: * Programmer:       Garrett A. Hughes
11: *
12:
13: *
14: * Purpose:  MasterClock mechanism for updating the time in "tick" units
15: *
16: * I/O:
17: *
18: ************************************************************************** */
19:
20: /* Ruler
21:         1         2         3         4         5         6         7
22: 12456789012345678901234567890123456789012345678901234567890123456789012345678
23: */
24:
25: // from docs.oracle.com/javase/8/docs/api
26: import java.time.Instant;
27: import java.time.Duration;
28:
29:
30: public class MasterClockRate
31: {
32:     // INSTANCE FIELDS
33:
34:     String name = "MasterClockRate";
35:     MasterClock mscl;
36:     int ticksElapsed;
37:
38:
39:     // INITIALIZATION BLOCKS
40:
41:
42:     // CONSTRUCTORS
43:
44:
45:     // METHODS
46:     public void reportName()
47:     {
48:         System.out.println("\n" + name);
49:
50:     }
51:
52:
53:    public void updateLinks(MasterClock mscl)
54:    {
55:         this.mscl = mscl;
56:
57:    } // END method: shareLinks
```

```
58:
59:
60:     public void updateMasterClockRate( int ticksElapsed)
61:     {
62:          this.ticksElapsed = ticksElapsed;
63:
64:     }
65:
66:
67: } // END class: MasterClockRate
68:
```

Patient.java

Friday, June 30, 2023 11:44 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```java
package xsd;

/** ***************************************************************************
*
* Class ID:         Patient
* Creation Date:    2023-01-16
* Updated:          2023-02-23
* Version:          v1.1.2
*
* Programmer:       Garrett A. Hughes
*
* Purpose:          Ingest a prescribed drug on a scheduled basis
*
*                   Eliminate that drug from their system based on the
*                     the drug's normal half life in a human body.
*
* I/O
*
********************************************************************************* */

/* Ruler
        1         2         3         4         5         6         7
12456789012345678901234567890123456789012345678901234567890123456789012345678
*/

public class Patient
{
    // INSTANCE FIELDS
    String name = "Patient";

    int startTime;          // ticks

    boolean active = false;

    double bloodVolume;     // liters
    double hoursInDay;      // hours/day

    EventGenerator evgn;
    Prescription pr;
    MasterClock mscl;
    IngestionRate inrt;
    EliminationRate elrt;

    Prescription prescrip;


    // INITIALIZATION BLOCKS
    {

        startTime = 0;      // ticks
        bloodVolume = 5.0;  //liters

    }


    // CONSTRUCTORS

```

```
58:
59:     // METHODS
60:     public void updateLinks(EventGenerator evgn, Prescription pr,
61:         MasterClock mscl, IngestionRate inrt, EliminationRate elrt)
62:     {
63:         this.evgn = evgn;
64:         this.pr = pr;
65:         this.mscl = mscl;
66:         this.inrt = inrt;
67:         this.elrt = elrt;
68:
69:     }
70:
71:     public boolean updatePatientActivity()
72:     {
73:         if (mscl.ticks >= startTime)
74:           active = true;
75:         return active;
76:
77:     }
78:
79:
80: } // END class: Patient
81:
```

Prescription.java

Friday, June 30, 2023 11:46 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```
 1: package xsd;
 2:
 3: /** ************************************************************************
 4: *
 5: * Class ID:         Prescription
 6: * Creation Date:    2023-01-16
 7: * Updated:          2023-03-05
 8: * Version:          v1.1.8
 9: *
10: * Programmer:       Garrett A. Hughes
11: *
12: * Purpose:          Provide description of current prescription for patient
13: *
14: ************************************************************************ */
15:
16: /* Ruler
17:         1         2         3         4         5         6         7
18: 12456789012345678901234567890123456789012345678901234567890123456789012345678
19:
20: */
21:
22: public class Prescription
23: {
24:     // INSTANCE FIELDS
25:     String name = "Prescription";
26:
27:     Patient pt;
28:
29:     String drugName =       "clonazepam";
30:     String drugHalflife =   "35 hours";
31:     String drugDose =       "0.25 milligrams";
32:     String drugFrequency =  "3x daily";
33:     double halflife; // hours
34:     double dose =   0.25;   // milligrams
35:     int frequency = 3;      // doses/day
36:
37:
38:     // INITIALIZATION BLOCK
39:     {
40:         halflife = 35.0;  // hours
41:
42:     }
43:
44:
45:     // CONSTRUCTORS
46:
47:
48:     // METHODS
49:      public void updateLinks(Patient pt)
50:     {
51:         this.pt = pt;
52:
53:     }
54:
55:
56:     public void showPrescription()
57:     {
```

```
58:          // show patient's current prescription
59:         System.out.println( "\nPatient's Current Prescription");
60:         System.out.println ("    name:      " + drugName);
61:         System.out.println ("    halflife:  " + drugHalflife);
62:         System.out.println ("    dose:      " + drugDose);
63:         System.out.println ("    frequency: " + drugFrequency);
64:
65:     }
66:
67: } // END class: Prescription
68:
```

DrugLevel.java

Friday, June 30, 2023 11:50 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```
 1: package xsd;
 2:
 3: /** ************************************************************************
 4: *
 5: * Class ID:         DrugLevel
 6: * Creation Date:    2023-01-02
 7: * Updated:          2023-03-09
 8: * Version:          v1.1.3
 9: *
10: * Programmer:       Garrett A. Hughes
11: *
12: * Purpose:          Keep track of total drug amount in body of patient while
13: *                   patient is ingesting that drug.  This component holds a
14: *                   state variable "drugLevel".
15: *
16: * I/O:
17: *
18: ************************************************************************** */
19:
20: /* Ruler
21:         1         2         3         4         5         6         7
22: 12456789012345678901234567890123456789012345678901234567890123456789012345678
23: */
24:
25: // from docs.oracle.com/javase/8/docs/api
26: import java.time.Instant;
27: import java.time.Duration;
28: import java.io.*;
29:
30:
31: public class DrugLevel
32: {
33:     // INSTANCE FIELDS
34:
35:     String name = "DrugLevel";
36:
37:     EventGenerator evgn;
38:     MasterClock mscl;
39:     IngestionRate inrt;
40:     EliminationRate elrt;
41:     Patient pt;
42:
43:     double drugLevel;
44:     double eliminationRate;
45:     double conversionFactor = 1000.0; // milligrams/liter to
    nanograms/milliliter
46:     double blsrcn; // nanograms/milliliter
47:     int elapsedTimeInHours;
48:
49:     PrintWriter out;
50:
51:
52:     // INITIALIZATION BLOCKS
53:
54:     {
55:         drugLevel = 0.0; // (milligrams)
56:
```

```
57:     }
58:
59:
60:     // CONSTRUCTORS
61:
62:
63:     // METHODS
64:
65:     public void updateLinks(EventGenerator evgn, MasterClock mscl,
66:         IngestionRate inrt, EliminationRate elrt, Patient pt)
67:     {
68:         this.evgn = evgn;
69:         this.mscl = mscl;
70:         this.inrt = inrt;
71:         this.elrt = elrt;
72:         this.pt = pt;
73:
74:     }
75:
76:
77:     public void updateDrugLevel()
78:     {
79:         if( (inrt.ingesting > 0.0) )
80:         {
81:             drugLevel = drugLevel  - (elrt.eliminationRate * evgn.DT);
82:
83:             elapsedTimeInHours = (mscl.ticks / evgn.numberTicksPerHour);
84:
85:
86:             eliminationRate = elrt.eliminationRate;
87:
88:             blsrcn = (drugLevel / pt.bloodVolume) * conversionFactor;
89:
90:             // output drug level afer elimination but before ingesting to
    avoid conflating the two
91:             out.printf("%7d  %13.6f  %13.6f  %13.6f\n", elapsedTimeInHours,
92:             drugLevel, eliminationRate, blsrcn);
93:
94:             drugLevel = drugLevel + inrt.ingesting;
95:
96:         }
97:         else
98:         {
99:             drugLevel = drugLevel - (elrt.eliminationRate * evgn.DT);
100:
101:        }
102:
103:    }
104:
105:
106:    public void updatePrintWriter(PrintWriter out)
107:    {
108:        this.out = out;
109:
110:    }
111:
112:
```

```
113: } // END class: DrugLevel
114:
```

IngestionRate.java

Friday, June 30, 2023 11:53 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```
 1: package xsd;
 2:
 3: /** ************************************************************************
 4: *
 5: * Class ID:         IngestionRate
 6: * Creation Date:    2023-01-16
 7: * Updated:          2023-03-06
 8: * Version:          v1.1.1
 9: *
10: * Programmer:       Garrett A. Hughes
11: *
12: * Purpose:          Ingest dose and increment drug level
13: *
14: * I/O:
15: *
16: ************************************************************************** */
17:
18: /* Ruler
19:         1         2         3         4         5         6         7
20: 12456789012345678901234567890123456789012345678901234567890123456789012345678
21: */
22:
23: // from docs.oracle.com/javase/8/docs/api
24: import java.time.Instant;
25: import java.time.Duration;
26:
27:
28: public class IngestionRate
29: {
30:     // INSTANCE FIELDS
31:
32:     String name = "IngestionRate";
33:     DrugLevel drlv;
34:     Prescription pr;
35:     Patient pt;
36:     MasterClock mscl;
37:     EventGenerator evgn;
38:
39:     double ingesting;  // milligrams
40:
41:
42:     // INITIALIZATION BLOCKS
43:
44:
45:     // CONSTRUCTORS
46:
47:
48:     // METHODS
49:
50:     public void updateLinks(DrugLevel drlv, Prescription pr, Patient pt,
51:             MasterClock mscl, EventGenerator evgn)
52:     {
53:         this.drlv = drlv;
54:         this.pr = pr;
55:         this.pt = pt;
56:         this.mscl = mscl;
57:         this.evgn = evgn;
```

```
58:
59:     }
60:
61:
62:     public void updateIngestionRate()
63:     {
64:         if (pt.active) // if patient is already active
65:         {
66:             if (mscl.ticks == evgn.runlength)
67:             // don't ingest anything at end of run
68:             {
69:                 ingesting = 0.0;
70:             }
71:             else if ((mscl.ticks - pt.startTime) % evgn.interarrivalTime == 0)
72:             {
73:                 ingesting = pr.dose;
74:             }
75:             else
76:             {
77:                 ingesting = 0.0;
78:             }
79:         }
80:         else  // modify patient status if criteria are met
81:         {
82:              if (pt.updatePatientActivity() == true)
83:                 ingesting = pr.dose;
84:         }
85:
86:     } // END method: updateIngestionRate()
87:
88:
89: } // END class: IngestionRate
90:
```

EliminationRate.java

Friday, June 30, 2023 11:55 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```
 1: package xsd;
 2:
 3: /** ************************************************************************
 4: *
 5: * Class ID:         EliminationRate
 6: * Creation Date:    2023-01-27
 7: * Updated:          2023-03-05
 8: * Version:          v1.0.5
 9: *
10: * Programmer:       Garrett A. Hughes
11: *
12: * Purpose:          Eiminate drug from patient's system.
13: *
14: *
15: * I/O:
16: *
17: ************************************************************************** */
18:
19: /* Ruler
20:         1         2         3         4         5         6         7
21: 12456789012345678901234567890123456789012345678901234567890123456789012345678
22: */
23:
24: // from docs.oracle.com/javase/8/docs/api
25: import java.time.Instant;
26: import java.time.Duration;
27:
28:
29: public class EliminationRate
30: {
31:     // INSTANCE FIELDS
32:
33:     String name = "EliminationRate";
34:
35:     DrugLevel drlv;
36:     MasterClock mscl;
37:     Prescription pr;
38:     EventGenerator evgn;
39:
40:     double eliminationFraction; // 1/hours
41:     double eliminationRate;     // milligrams/hour
42:
43:     // INITIALIZATION BLOCKS
44:
45:
46:     // CONSTRUCTORS
47:
48:
49:     // METHODS
50:     public void updateLinks(EventGenerator evgn, DrugLevel drlv,
51:         MasterClock mscl, Prescription pr)
52:     {
53:         this.evgn = evgn;
54:         this.drlv = drlv;
55:         this.mscl = mscl;
56:         this.pr = pr;
57:
```

```
58:     }
59:
60:
61:     public void updateEliminationRate()
62:     {
63:         /*
64:         The elimination rate depends on the halflife of the drug being
65:         ingested. The range of the halflife is given as 30 to 40 hours
66:         in NDA 017533 of the Food and Drug Administration dated
67:         October 2013. We use 35 hours for purposes of analysis.
68:
69:         Elimination is assumed to be continuous, therefore we compute
70:         the amount being eliminated at each DT of the model run using
71:         numerical integration. The elimination rate is assumed to be
72:         proportional to the drug level remaining in the patient's blood
73:         serum.
74:
75:         We compute the constant of proportionality, the
76:         eliminationFraction, from the drug halflife.
77:         */
78:
79:
80:         eliminationFraction = Math.abs(Math.log(0.5)/pr.halflife);
81:
82:         // We will return the negative sign when we use the
83:         // elimination rate to reduce the drug level
84:
85:         eliminationRate = drlv.drugLevel * eliminationFraction;
86:
87:
88:     }
89:
90:
91:
92:
93: } // END class: EliminationRate
94:
```

BloodSerumConcentration.java

Friday, June 30, 2023 11:05 AM

ModelingComplexSystems.net
Collection: Model Construction
Title: Methodology
Section: Drug Levels Model.Preview.code

```java
package xsd;

/** ************************************************************************
*
* Class ID:         BloodSerumConcentration
* Creation Date:    2023-02-21
* Updated:          2023-02-21
* Version:          v1.0.0
*
* Programmer:       Garrett A. Hughes
*
* Purpose:          Compute the concentration of the drug in the blood serum
*
* I/O:              no class members required
*
************************************************************************** */

/* Ruler
        1         2         3         4         5         6         7
12456789012345678901234567890123456789012345678901234567890123456789012345678
*/

// from docs.oracle.com/javase/8/docs/api
import java.time.Instant;
import java.time.Duration;


public class BloodSerumConcentration
{
    // INSTANCE FIELDS

    DrugLevel drlv;
    Patient pt;

    double bloodSerumConcentration;  // (nanograms/milliliter)


    // INITIALIZATION BLOCKS


    // CONSTRUCTORS


    // METHODS

    public void updateLinks(DrugLevel drlv, Patient pt)
    {
        this.drlv = drlv;
        this.pt = pt;

    }


    public void updateBloodSerumConcentration()
    {
        bloodSerumConcentration = drlv.drugLevel / pt.bloodVolume;

```

```
58:         // NOTE: concentration is currently in milligams/liter
59:         //       convert concentration to nanograms per milliliter
60:         //       multiply by 1000
61:
62:         bloodSerumConcentration *= 1000.0;
63:
64:     }
65:
66:
67:
68:
69: } // END class: BloodSerumConcentration
70:
```