Contents

1 Task 1

a) Passenger:

$$\begin{aligned}
Passenger &\stackrel{def}{=} (arrive, 0.5\lambda).Passenger_1 + (arrive, 0.5\lambda).Passenger_2 \\
Passenger_1 &\stackrel{def}{=} (deposit, 0.9d).(walk, \omega).(collect, c).Passenger_R \\
&+ (deposit, 0.1d).(walk, \omega).(search, s).(collect, c).Passenger_R \\
Passenger_2 &\stackrel{def}{=} (deposit, 0.9d).(deposit, d).(walk, \omega).(collect, c).(collect, c).Passenger_R \\
&+ (deposit, 0.1d).(deposit, d).(walk, \omega).(search, s).(collect, c).(collect, c).Passenger_R
\end{aligned}$$

b) Conveyor Belt:

$$ConveyorBelt \stackrel{def}{=} (deposit, \top).(screening, sc).(collect, \top).ConveyorBelt$$

c) To deal with the Bags, I modelled the Conveyer Belt with a capacity of 6 Bags as a FIFO queue.

$$\begin{aligned}
ConveyorBelt_0 &\stackrel{def}{=} (deposit, \top).ConveyorBelt_1 \\
ConveyorBelt_1 &\stackrel{def}{=} (deposit, \top).ConveyorBelt_2 + (screening, sc).(collect, \top).ConveyorBelt_0 \\
ConveyorBelt_2 &\stackrel{def}{=} (deposit, \top).ConveyorBelt_3 + (screening, sc).(collect, \top).ConveyorBelt_1 \\
ConveyorBelt_3 &\stackrel{def}{=} (deposit, \top).ConveyorBelt_4 + (screening, sc).(collect, \top).ConveyorBelt_2 \\
ConveyorBelt_4 &\stackrel{def}{=} (deposit, \top).ConveyorBelt_5 + (screening, sc).(collect, \top).ConveyorBelt_3 \\
ConveyorBelt_5 &\stackrel{def}{=} (deposit, \top).ConveyorBelt_6 + (screening, sc).(collect, \top).ConveyorBelt_4 \\
ConveyorBelt_6 &\stackrel{def}{=} (screening, sc).(collect, \top).ConveyorBelt_5
\end{aligned}$$

$$\begin{aligned}
Checkpoint &\stackrel{def}{=} (Passenger \parallel Passenger \parallel Passenger) \underset{L}{\bowtie} ConveyorBelt_0 \\
&\text{where } L = \{deposit, collect\}.
\end{aligned}$$

d) I used the ml-version of the Pepa Workbench. Therefor I couldn't write `0.9*d`, so I introduced `d` and `dr` file:

Checkpoint.pepa

```
%lambda = 0.05;
%d = 1;
%w = 0.2;
```

```
%c = 0.3;
%s = 0.02;
%sc= 0.1;

#Passenger  = (arrive, lambda).Passenger1 + (arrive, lambda).Passenger2;
#Passenger1 = (deposit, 0.9*d).(walk, w).(collect, c).Passenger
            + (deposit, 0.1*d).(walk, w).(search, s).(collect, c).Passenger;
#Passenger2 = (deposit, 0.9*d).(deposit, d).(walk, w).(collect, c).(collect, c).Passenger
            + (deposit, 0.1*d).(deposit, d).(walk, w).(search, s)
            .(collect, c).(collect, c).Passenger;

#ConveyorBelt0 = (deposit, infty).ConveyorBelt1;

#ConveyorBelt1 = (deposit, infty).ConveyorBelt2
               + (screening, sc).(collect, infty).ConveyorBelt0;
#ConveyorBelt2 = (deposit, infty).ConveyorBelt3
               + (screening, sc).(collect, infty).ConveyorBelt1;
#ConveyorBelt3 = (deposit, infty).ConveyorBelt4
               + (screening, sc).(collect, infty).ConveyorBelt2;
#ConveyorBelt4 = (deposit, infty).ConveyorBelt5
               + (screening, sc).(collect, infty).ConveyorBelt3;
#ConveyorBelt5 = (deposit, infty).ConveyorBelt6
               + (screening, sc).(collect, infty).ConveyorBelt4;
#ConveyorBelt6 = (screening, sc).(collect, infty).ConveyorBelt5;
(Passenger <>  Passenger <>  Passenger) < deposit, collect > ConveyorBelt0
```

The Result is:

pwb output

```
The model has 4367 states
The model has 14284 transitions
Writing the hash table file to Checkpoint.ml.hash
Exiting PEPA Workbench.
```

2 Task 2

a) I chose the application variant for my model, that consists of a source and a server. Each entity has one port and the passed events have tag 0 and no data. After problems with the Poisson distribution I remembered the Poisson limit law, which says that a binomial distribution (n, p) with small p and large n is nearly Poisson distributed with $\lambda = n \cdot p$. For souce code see on page 7.

b) During my tests I realised, that the variance of my binomial distribution was not λ as claimed by the Poisson distribution. The variance was instead about one magnitude smaller. I decided to take a exponential distribution but here is variance quadratic in relation to the mean value. Therefore are the calculated values only of partial relevance. For souce code see on page 9.

c) I used for the different runs a new random seed by calling the function `Sim_system.next_seed` several times. I performed 4 runs with 50000 events. Here one of the report files:

2b/sim_report

```
############################################################
#                                                          #
#                     SIMULATION REPORT                    #
#                                                          #
############################################################

```

```
Version: SimJava 2.0

Simulation date:         8. Dezember 2002
Simulation start time:   01:30:09 CET
Simulation end time:     01:30:56 CET

############################################################
#            Overall simulation run information            #
############################################################

Total simulated time:    3072103.689569396
Total transient time:    0.0
Total steady state time: 3072103.689569396
Transient condition:     None
Termination condition:   50000 event completions at Service
Output analysis method:  None

############################################################
#               Simulation run measurements                #
############################################################

------------------------ Customers -------------------------

- Generation rate

Sample mean:          0.016684332685132884
Event count:          51256

------------------------- Service --------------------------

- Utilisation

Sample mean:          0.7324082110014821

- Waiting time

Sample mean:          51.1544200634707
Sample variance:      2790.580476859551
Sample std deviation: 52.82594511089746
Maximum:              224.42379269504454
Minimum:              0.0

- Deflect rate

Sample mean:          4.0884036703072623E-4
Event count:          1256

```

Run	1	2	3	4	average
Deflects	1256	1309	1231	1278	1268
Deflection Rate	$4.088 \cdot 10^{-4}$	$4.234 \cdot 10^{-4}$	$4.01 \cdot 10^{-4}$	$4.15 \cdot 10^{-4}$	$4.120 \cdot 10^{-4}$

d) I changed line 99 in the source code of part b) to

```
         Server service = new Server("Service", 35.0, 20.0, 180.0);
```

2d/sim_report

```
############################################################
#                                                          #
#                    SIMULATION REPORT                     #
#                                                          #
############################################################

Version: SimJava 2.0

Simulation date:         8. Dezember 2002
```

```
Simulation start time:   01:26:54 CET
Simulation end time:     01:27:33 CET

############################################################
#           Overall simulation run information             #
############################################################

Total simulated time:    2996275.024126742
Total transient time:    0.0
Total steady state time: 2996275.024126742
Transient condition:     None
Termination condition:   50000 event completions at Service
Output analysis method:  None

############################################################
#              Simulation run measurements                 #
############################################################

------------------------ Customers -------------------------

- Generation rate

Sample mean:          0.016718091495822953
Event count:          50092

------------------------- Service --------------------------

- Utilisation

Sample mean:          0.5840745668900222

- Waiting time

Sample mean:          24.005333839884273
Sample variance:      1062.4730671367815
Sample std deviation: 32.59559889213238
Maximum:              214.74985523801297
Minimum:              0.0

- Deflect rate

Sample mean:          3.07047915358882E-5
Event count:          92
```

Run	1	2	3	4	average
Deflects	92	100	104	104	100
Deflection Rate	$3.07 \cdot 10^{-5}$	$3.32 \cdot 10^{-5}$	$3.47 \cdot 10^{-5}$	$3.46 \cdot 10^{-5}$	$3.33 \cdot 10^{-5}$

3 Task 3

a) Errors:

1. **Compilation**
2. **Structured walk through at high level:** Explanation to myself and realized that Disk request from Database server should go back to Database server rather than to the Application server.
3. **Tracing:** Application server seems to stop the flow of messages! Mean time error.
4. **Tracing:** I traced the System with a very high inter arrival time, to track one message. Obviously there was no second processing in the Database server.

5. **Tracing:** With a small inter arrival time the system stopped if the web server was full. U found the reason be looking through the WSQ code. There were the stop message leaved in the queue, therefore the queue thought to each time that the web server was full.

6. **Tracing:** Unfortunately I found a further error in the Queue or web server. Description: If the application server and the web server is full, than there exits a user `request` the web server and than the queue fires. After that the web server simulation thread is trapped into a waiting for a `start` message from the application server. During this time it don't process new `request` from the queue. So it happens that after getting that start message the application server but in the next step it there exits a next complete user `request` instead of processing the new incoming request. The web server don't become full after the new request. Therefore the WSQ don't get a new `stop` message later followed be a `start` message. The result is, that the queue stays full (not empty) during the web server has several free threads. I wrote several solutions for this problem, but I think this model is very complicated programmed, with a lot of site affects, so that no solution is perfect.

   My first solution contains a synchronisation and a buffering if `bf_out_requests` in the web server. That means each thread only send the request if the application server is not full, but without blocked waiting for a start message. Result: it worked, but the queue didn't become full really, so that I think it is not what you expect.

   My second solution is a further `if` case the WSQ. If there is a `request` in the queue and the web server is neither full in the knowledge of the Queu (`WSFULL`) nor too heavy loaded (more than 2 threads free), than flush a message.

For souce code see on page 10.

In all my following test I used a warmup time of 4000 and a total time of 500000. The warmup time come from a naiv observation of the animation, where all devices seem to be steady. The long running time come from a few tries with less time, where too less lost events are ocurred. My results with 3 different random seeds:

```
////////// SEED 1
Total simulated time:     500008.1586643648
Total transient time:     4013.6238648253425
Total steady state time: 495994.5347995394
Transient condition:      4000.0 units of elapsed simulated time
Termination condition:    500000.0 units of elapsed simulated time
.....

-------------------------- WSQ ---------------------------
- Loss rate
Sample mean:          8.064604989280044E-6
Event count:          4

////////// SEED 2
- Loss rate
Sample mean:          1.8144714683318705E-5
Event count:          9

////////// SEED 3
- Loss rate
Sample mean:          8.065284090293165E-6
Event count:          4
```

b) I have ran the simulation 5 times with different random seed with the same timing values like above and an inter arrival time of 300.

Run	1	2	3	4	5	average
Count	72	120	133	89	98	102.4
Mean	$1.45{\cdot}10^{-4}$	$2.42 \cdot 10^{-4}$	$2.68{\cdot}10^{-4}$	$1.79{\cdot}10^{-4}$	$1.98{\cdot}10^{-4}$	$2.06{\cdot}10^{-4}$

The mean lost rate is about 0.0002 request per ms and the variance is about ± 0.00005. This is about one magnitude greater than the old rate.

c) I added the following lines to the code:

3c/ECommerceSite.diff

```
*** 3_a/ECommerceSite.java  Tue Dec 10 13:46:42 2002
--- 3_c/ECommerceSite.java  Tue Dec 10 15:21:18 2002
***************
*** 85,86 ****
--- 85,87 ----
        stat.set_efficient("Loss rate");
+       stat.add_measure("WSQueue length", Sim_stat.STATE_BASED, 0.0); // added (continuous state measure)
        set_stat(stat);
***************
*** 126,127 ****
--- 127,129 ----
              num_requests--;
+             stat.update("WSQueue length", (double)num_requests, Sim_system.sim_clock());// added
              sim_trace(1, "P "+num_requests);
***************
*** 134,135 ****
--- 136,138 ----
              num_requests--;
+             stat.update("WSQueue length", (double)num_requests, Sim_system.sim_clock());// added
              sim_trace(1, "P "+num_requests);
***************
*** 142,143 ****
--- 145,147 ----
              num_requests++;
+             stat.update("WSQueue length", (double)num_requests, Sim_system.sim_clock());// added
              sim_trace(1, "P "+num_requests);
```

I have ran the simulation 3 times with different random seeds, with the same timing values like above, a queue length of 10 and an inter arrival time of 300. Here is a part of one report:

```
...
Total simulated time:    500015.1398919981
Total transient time:    4070.7831042164194
Total steady state time: 495944.35678778164
Transient condition:     4000.0 units of elapsed simulated time
Termination condition:   500000.0 units of elapsed simulated time
Output analysis method:  None
...
-------------------------- WSQ ---------------------------

- Loss rate

Sample mean:          6.855607798466968E-5
Event count:          34

- WSQueue length

Sample mean:          2.3890705110050816
Sample variance:      9.12398883021843
Sample std deviation: 3.0205941187485665
Maximum:              10.0
Minimum:              0.0
```

Run	1	2	3	result
Min	0	0	0	0
Max	10	10	10	10
Mean	2.39	3.26	3.80	3.15
σ	3.02	3.37	3.32	3.4

The mean queue lenght is about 3.2 and the standart variance is about 3.4. If the queue length would be normal distributed, than I would suggest a queue length of $\lceil 3.2 + 3\sigma = 13.4\rceil = \underline{14}$, because there is a 99.97% confidence interval of $[\mu - 3\sigma, \mu + 3\sigma]$. However, this is not the case, and the queue length don't realy help to improve the lost rate. The system is to slow, so that also a very large queue doesn't help. A long queue has the disatvantage of long waiting times. Maybe a queue length of 4 or 5 is optimal.

d) I added a new disk in the model. The database server got a new output port to the second disk. For the decision of the disks I have introduced a new random distribution object. This distribution is equal distributed in the interval $[0, 1)$. Therefore I check if the sample value is less than 0.6. In this case the request is for disk 1 otherwise for disk 2.

For souce code see on page 21.

My new results for 3 runs with the used timing parameters and 300 ia. time.

```
...
-------------------------- WSQ ---------------------------
- Loss rate

Sample mean:          1.915655427965474E-4
Event count:          95

- WSQueue length
Sample mean:          0.9828147138725555
Sample variance:      1.850704826064835
Sample std deviation: 1.360406125414332
Maximum:              4.0
Minimum:              0.0

--------------------- DatabaseDisk1 ----------------------

- Utilisation
Sample mean:          0.46912947946934297

--------------------- DatabaseDisk2 ----------------------

- Utilisation
Sample mean:          0.29755388097075847
```

Run	1	2	
Count	129	95	98
Mean	$2.60 \cdot 10^{-4}$	$1.92 \cdot 10^{-4}$	$1.98 \cdot 10^{-4}$

The lost rate is not much better with 2 disks, because the Application server is too slow.

A Souce code to task 2

A.1 Part a)

2a/Queue.java

```
/*
 * Author: Georg Martius
 * 2a: Queue
 */
```

```java

import eduni.simjava.*;
import eduni.simjava.distributions.*;

class Source extends Sim_entity {
  private Sim_port enqueue;
  private Sim_binomial_obj dist;
  private double lambda;
  private Sim_stat stat;

  public Source(String name, double lambda) {
    super(name);
    this.lambda = lambda;
    double p;
    enqueue = new Sim_port("enqueue");
    add_port(enqueue);
    /* fake the poisson distr with binomial distr with large n and small p
        limit theorem: if n big and p small then binomial(n,p) approx. poission(lambda = n*p)
        we keep p fixed and calc n = lambda / p  */
    p = 0.01;
    dist = new Sim_binomial_obj("lambda", p, (int)(lambda/p));
    add_generator(dist);
    stat = new Sim_stat();
    stat.add_measure("Generation rate", Sim_stat.RATE_BASED);
    set_stat(stat);
  }

  public void body() {
    while (Sim_system.running()) {
      stat.update("Generation rate", Sim_system.sim_clock());
      sim_schedule(enqueue, 0.0, 0);
      sim_pause(dist.sample());
    }
  }
}

class Server extends Sim_entity {
  private Sim_port arrival;
  private Sim_normal_obj dist;
  private Sim_stat stat;
  private double mu;
  private double vmu;

  public Server(String name, double mu, double vmu) {
    super(name);
    arrival = new Sim_port("arrival");
    add_port(arrival);
    this.mu = mu;
    this.vmu = vmu;
    dist = new Sim_normal_obj("serving", mu, vmu);
    add_generator(dist);
    stat = new Sim_stat();
    stat.add_measure(Sim_stat.UTILISATION);
    set_stat(stat);
  }

  public void body() {
    Sim_event next = new Sim_event();
    while (Sim_system.running()) {
      sim_wait(next);
      sim_process(dist.sample());
      sim_completed(next);
    }
  }
}

class Queue {
```

```
  public static void main(String args[]) {
    Sim_system.initialise();
    Source customers = new Source("Customers", 60.0);
    Server service = new Server("Service", 45.0, 10.0);
    Sim_system.link_ports("Customers", "enqueue", "Service", "arrival");
    Sim_system.set_termination_condition(Sim_system.EVENTS_COMPLETED, "Service", 0, 100, true);
    Sim_system.run();
  }
}

```

A.2 Part b)

2b/Queue.java

```
/*
 * Author: Georg Martius
 * 2a: Queue
 */

import eduni.simjava.*;
import eduni.simjava.distributions.*;

class Source extends Sim_entity {
  private Sim_port enqueue;
  private Sim_negexp_obj dist;
  private double lambda;
  private Sim_stat stat;


  public Source(String name, double lambda) {
    super(name);
    this.lambda = lambda;
    enqueue = new Sim_port("enqueue");
    add_port(enqueue);
    dist = new Sim_negexp_obj("lambda", lambda);

    add_generator(dist);
    stat = new Sim_stat();
    stat.add_measure("Generation rate", Sim_stat.RATE_BASED);
    set_stat(stat);
  }

  public void body() {

    while (Sim_system.running()) {
      stat.update("Generation rate", Sim_system.sim_clock());
      sim_schedule(enqueue, 0.0, 0);
      sim_pause(dist.sample());
    }
  }
}

class Server extends Sim_entity {
  private Sim_port arrival;
  private Sim_normal_obj dist;
  private Sim_stat stat;
  private double mu;
  private double vmu;

  class Deflect_Pred extends Sim_predicate{
    private double deflectRate;
    public Deflect_Pred(double _deflectRate){
      deflectRate = _deflectRate;
    }
    public boolean match(Sim_event ev){
```

```
      return ((Sim_system.sim_clock() - ev.event_time()) >= deflectRate);
    }
  }
  private Deflect_Pred deflect_pred;

  public Server(String name, double mu, double vmu, double deflect) {
    super(name);
    arrival = new Sim_port("arrival");
    add_port(arrival);
    this.mu = mu;
    this.vmu = vmu;
    deflect_pred = new Deflect_Pred(deflect);
    dist = new Sim_normal_obj("serving", mu, vmu);
    add_generator(dist);
    stat = new Sim_stat();
    stat.add_measure(Sim_stat.UTILISATION);
    stat.add_measure(Sim_stat.WAITING_TIME);
    stat.add_measure("Deflect rate", Sim_stat.RATE_BASED);
    set_stat(stat);
  }

  public void body() {
    Sim_event next = new Sim_event();
    int i;
    while (Sim_system.running()) {
      sim_get_next(next);
      while(deflect_pred.match(next)){
        stat.update("Deflect rate", Sim_system.sim_clock());
        System.out.println("Deflect with waiting time: " + (next.end_waiting_time() - next.event_time()));
        next = new Sim_event(); // to avoid that again deflected
        sim_get_next(next);
      }
      sim_process(dist.sample());
      sim_completed(next);
    }
  }
}

class Queue {
  public static void main(String args[]) {
    Sim_system.initialise();
    if(args != null && args.length >0){  // support commandline parameter for seed
      int seednumber = Integer.parseInt(args[0]);
      while ((seednumber--) != 0) {Sim_system.next_seed();}
    }

    Source customers = new Source("Customers", 60.0);
    Server service = new Server("Service", 45.0, 10.0, 180.0);
    Sim_system.link_ports("Customers", "enqueue", "Service", "arrival");
    Sim_system.set_termination_condition(Sim_system.EVENTS_COMPLETED, "Service", 0, 50000, true);
    Sim_system.run();
  }
}

```

B Souce code to task 3

B.1 Part a)

Here my code. All changes are commented either through `//added` or through `//Error X`.

3a/ECommerceSite.java

```
/*
 * The E-Commerce Site (animated) -- with errors
```

```
 *
 */

/* LOG
  - add variance to WS distribution (Error 1)
  - modify connection from Disk to DBS (Error 2)
  -  add overload sim_trace message with loglevel to each entity for testing purpose
  - change the mean time of the distribution of the application server from 1500 to 150 (Error 3)
  - add the second processing of the data at the database server (Error 4)
  - getting the stop message out of the queue at WSQ (Error 5)
  -  insert inputbox for random seed and necessary variables
  - insert additional case in WSQ for flushing (Error 6)

*/
import java.applet.*;
import java.awt.*;
import eduni.simjava.*;
import eduni.simanim.*;
import eduni.simjava.distributions.*;

class Source extends Sim_entity {
  private Sim_port enqueue, exit;
  private Sim_negexp_obj source_hold;
  private double iatime;
// Added
  private int loglevel;


//  public Source(String name, int x, int y, double iatime) {
  public Source(String name, int x, int y, double iatime, int loglevel) {
    super(name, "source1", x, y);
    this.iatime = iatime;
    this.loglevel = loglevel;//added

    enqueue = new Sim_port("enqueue", "port", Anim_port.RIGHT, 10);
    add_port(enqueue);
    exit = new Sim_port("exit", "port", Anim_port.RIGHT, 70);
    add_port(exit);
    source_hold = new Sim_negexp_obj("iatime", iatime);
    add_generator(source_hold);
  }

  //Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("Src  (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }


  public void body() {
    while (Sim_system.running()) {
        sim_pause(source_hold.sample());
        sim_schedule(enqueue, 0.0, 1);
        sim_trace(1,"S enqueue");
    }
  }
}

class WSQ extends Sim_entity {
  private Sim_port request_queue, access, ws_control;
  private Sim_stat stat;
  private int num_requests, capacity;
  private boolean WS_FULL;
//added
  private int loglevel;
```

```
//  public WSQ(String name, int x, int y, int WSQcap) {
  public WSQ(String name, int x, int y, int WSQcap, int loglevel) {

    super(name, "wsq", x, y);
    this.capacity = WSQcap;
    this.loglevel = loglevel;//added

    request_queue = new Sim_port("request_queue", "port", Anim_port.LEFT, 10);
    add_port(request_queue);
    access = new Sim_port("access", "port", Anim_port.RIGHT, 10);
    add_port(access);
    ws_control = new Sim_port("ws_control", "port", Anim_port.RIGHT, 30);
    add_port(ws_control);
    add_param(new Anim_param("num_requests", Anim_param.VALUE, "0", 1, 10));
    stat = new Sim_stat();
    stat.add_measure("Loss rate", Sim_stat.RATE_BASED);
    stat.set_efficient("Loss rate");
    set_stat(stat);
  }

//Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("WSQ  (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

  public void body() {
    Sim_type_p new_request = new Sim_type_p(1);
    Sim_type_p stop_msg = new Sim_type_p(10);
    Sim_type_p start_msg = new Sim_type_p(11);
    int j;

    WS_FULL = false;
    num_requests = 0;

    Sim_event new_event = new Sim_event();
    sim_get_next(new_event);
    sim_schedule(access, 0.0, 2);
    sim_trace(1, "S access");

    while (Sim_system.running()) {
      if ((sim_waiting(stop_msg) > 0) && !(WS_FULL)) {
//added
        sim_select(stop_msg, new_event); // Error 5 (missed line)
        WS_FULL = true;
        continue;
      } else if ((sim_waiting(start_msg) > 0) && (WS_FULL)) {
        sim_select(start_msg, new_event);

        WS_FULL = false;
        j = WebServer.webThreads - WebServer.num_threads;
        if (j > num_requests) j = num_requests;
        while ((j > 0) && Sim_system.running()) {
          sim_schedule(access, 0.0, 2);
          sim_trace(1, "S access");
          num_requests--;
          sim_trace(1, "P "+num_requests);
          j--;
        }
      } else // added for flushing queue if full and Webserver have capacities
              if(num_requests != 0 && (WebServer.webThreads - WebServer.num_threads)>=2 && !(WS_FULL)){
          sim_schedule(access, 0.0, 2);
          sim_trace(1, "S access");
          num_requests--;
```

```
          sim_trace(1, "P "+num_requests);
      } else  if (sim_waiting(new_request) > 0) {
         sim_select(new_request, new_event);
        if (!(WS_FULL)){
          sim_schedule(access, 0.0, 2);
          sim_trace(1, "S access");
        } else if (num_requests < capacity) {
          num_requests++;
          sim_trace(1, "P "+num_requests);
        } else {
          stat.update("Loss rate", Sim_system.sim_clock());
        }
      } else {
        sim_get_next(new_event);
        sim_putback(new_event);
      }
    }
  }
}

class WebServer extends Sim_entity {
  private Sim_port grant_thread, bf_req_out, bf_reply_in, exit, q_control, as_control;
  private Sim_normal_obj WS_hold;
  private final double WSmean = 120;
  private final double WSvar = 15;
  public static boolean FULL;
  public static boolean AS_FULL;
  public static int webThreads, num_threads;
// Added
  private int loglevel;

//  public WebServer(String name, int x, int y, int WSThreads) { // add loglevel
  public WebServer(String name, int x, int y, int WSThreads, int loglevel) {
    super(name, "webserver", x, y);
    this.webThreads = WSThreads;
    this.loglevel = loglevel;//added

    grant_thread = new Sim_port("grant_thread", "port", Anim_port.LEFT, 10);
    add_port(grant_thread);
    bf_req_out = new Sim_port("bf_req_out", "port", Anim_port.RIGHT, 10);
    add_port(bf_req_out);
    as_control = new Sim_port("as_control", "port", Anim_port.RIGHT, 30);
    add_port(as_control);
    bf_reply_in = new Sim_port("bf_reply_in", "port", Anim_port.RIGHT, 70);
    add_port(bf_reply_in);
    exit = new Sim_port("exit", "port", Anim_port.LEFT, 70);
    add_port(exit);
    q_control = new Sim_port("q_control", "port", Anim_port.LEFT, 30);
    add_port(q_control);

    add_param(new Anim_param("num_threads", Anim_param.VALUE, "0", 1,10));
    //  WS_hold = new Sim_normal_obj("WS_hold", WSmean);  error 1
    WS_hold = new Sim_normal_obj("WS_hold", WSmean, WSvar);
    add_generator(WS_hold);
  }
//Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("WS   (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

  public void body() {
    Sim_type_p request = new Sim_type_p(2);
    Sim_type_p reply = new Sim_type_p(6);
    Sim_type_p stop_msg = new Sim_type_p(12);
```

```
    Sim_type_p start_msg = new Sim_type_p(13);

    FULL = false;
    AS_FULL = false;
    num_threads = 0;

    Sim_event next_event = new Sim_event();
    sim_get_next(request, next_event);
    num_threads++;
    sim_trace(1,"P "+num_threads);
    if (num_threads == webThreads) {
        FULL=true;
        sim_schedule(q_control, 0.0, 10);
        sim_trace(1, "S q_control");
    }
    sim_process(WS_hold.sample());
    sim_completed(next_event);
    sim_schedule(bf_req_out, 0.0, 3);
    sim_trace(1, "S bf_req_out");

    while (Sim_system.running()) {
      if ((sim_waiting(stop_msg) > 0) && !(AS_FULL)) {
        sim_select(stop_msg, next_event);
        AS_FULL = true;
        sim_completed(next_event);
      continue;
         } else if ((sim_waiting(start_msg) > 0) && (AS_FULL)) {
        sim_select(start_msg, next_event);
        AS_FULL = false;
        sim_completed(next_event);
      continue;
        } else if (sim_waiting(request) > 0 && (!FULL)) {
        sim_select(request, next_event);
        num_threads++;
        sim_trace(1,"P "+num_threads);
        if (num_threads == webThreads) {
            FULL= true;
            sim_schedule(q_control, 0.0, 10);
            sim_trace(1, "S q_control");
        }
        sim_process(WS_hold.sample());
        sim_completed(next_event);
            if (AS_FULL == true) {
            sim_get_next(start_msg, next_event);
            AS_FULL = false;
            sim_completed(next_event);
        }
        sim_schedule(bf_req_out, 0.0, 3);
        sim_trace(1, "S bf_req_out");
        } else if (sim_waiting(reply) > 0) {
            sim_select(reply, next_event);
        sim_process(WS_hold.sample());
        sim_completed(next_event);
        sim_schedule(exit, 0.0, 7);
        sim_trace(1, "S exit");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        if (FULL == true) {
            FULL = false;
            sim_schedule(q_control, 0.0, 11);
            sim_trace(1, "S q_control");
        }
        } else if (FULL) {
        sim_get_next(reply, next_event);
        sim_process(WS_hold.sample());
        sim_completed(next_event);
        sim_schedule(exit, 0.0, 7);
```

```
        sim_trace(1, "S exit");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        FULL = false;
        sim_schedule(q_control, 0.0, 11);
        sim_trace(1, "S q_control");
        } else { //?
        sim_get_next(next_event);
        sim_putback(next_event);
        }
    }
  }
}

class ApplicationServer extends Sim_entity {
    private Sim_port bf_req_in, bf_reply_out, trans_req_out, trans_reply_in, ws_control, dbs_control;
    private Sim_normal_obj AS_hold;
//    private final double ASmean = 1500; // Error 3
    private final double ASmean = 150;
    private final double ASvar = 50;
    private int appThreads;
    private int num_threads;
    private boolean FULL;
    private boolean DBS_FULL;
  // Added
  private int loglevel;

//    public ApplicationServer(String name, int x, int y, int ASThreads) {
    public ApplicationServer(String name, int x, int y, int ASThreads, int loglevel) {

    super(name, "applicationserver", x, y);
    this.appThreads = ASThreads;
    this.loglevel = loglevel;//added

    bf_req_in = new Sim_port("bf_req_in", "port", Anim_port.LEFT, 10);
    add_port(bf_req_in);
    trans_req_out = new Sim_port("trans_req_out", "port", Anim_port.RIGHT, 10);
    add_port(trans_req_out);
    dbs_control = new Sim_port("dbs_control", "port", Anim_port.RIGHT, 30);
    add_port(dbs_control);
    trans_reply_in = new Sim_port("trans_reply_in", "port", Anim_port.RIGHT, 70);
    add_port(trans_reply_in);
    bf_reply_out = new Sim_port("bf_reply_out", "port", Anim_port.LEFT, 70);
    add_port(bf_reply_out);
    ws_control = new Sim_port("ws_control", "port", Anim_port.LEFT, 30);
    add_port(ws_control);

    add_param(new Anim_param("num_threads", Anim_param.VALUE, "0", 1,10));
    AS_hold = new Sim_normal_obj("AS_hold", ASmean, ASvar);
    add_generator(AS_hold);
    }

//Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("AS   (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

  public void body() {
    Sim_type_p request = new Sim_type_p(3);
    Sim_type_p reply = new Sim_type_p(5);
    Sim_type_p stop_msg = new Sim_type_p(14);
    Sim_type_p start_msg = new Sim_type_p(15);

    FULL = false;
```

```
    DBS_FULL = false;
    num_threads = 0;

    Sim_event next_event = new Sim_event();
    sim_get_next(request, next_event);
    num_threads++;
    sim_trace(1,"P "+num_threads);
    if (num_threads == appThreads) {
      FULL= true;
      sim_schedule(ws_control, 0.0, 12);
      sim_trace(1, "S ws_control");
    }
    sim_process(AS_hold.sample());
    sim_completed(next_event);
    sim_schedule(trans_req_out, 0.0, 4);
    sim_trace(1, "S trans_req_out");

    while (Sim_system.running()) {
      if ((sim_waiting(stop_msg) > 0) && !(DBS_FULL)) {
        sim_select(stop_msg, next_event);
        DBS_FULL = true;
        sim_completed(next_event);
        continue;
      } else if ((sim_waiting(start_msg) > 0) && (DBS_FULL)) {
        sim_select(start_msg, next_event);
        DBS_FULL = false;
        continue;
      } else if ((sim_waiting(request) > 0) && (!FULL)) { //? ()
        sim_select(request, next_event);
        num_threads++;
        sim_trace(1,"P "+num_threads);
        if (num_threads == appThreads) {
          FULL= true;
          sim_schedule(ws_control, 0.0, 12);
          sim_trace(1, "S ws_control");
        }
        sim_process(AS_hold.sample());
        sim_completed(next_event);
        if (DBS_FULL == true) {
          sim_get_next(start_msg, next_event);
          DBS_FULL = false;
          sim_completed(next_event);
        }
        sim_schedule(trans_req_out, 0.0, 4);
        sim_trace(1, "S trans_req_out");
      } else if (sim_waiting(reply) > 0) {
        sim_select(reply, next_event);
        sim_process(AS_hold.sample());
        sim_completed(next_event);
        sim_schedule(bf_reply_out, 0.0, 6);
        sim_trace(1, "S bf_reply_out");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        if (FULL == true) {
          FULL = false;
          sim_schedule(ws_control, 0.0, 13);
          sim_trace(1, "S ws_control");
        }
      } else if (FULL) {
        sim_get_next(reply, next_event);
        sim_process(AS_hold.sample());
        sim_completed(next_event);
        sim_schedule(bf_reply_out, 0.0, 6);
        sim_trace(1, "S bf_reply_out");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        FULL = false;
```

```
        sim_schedule(ws_control, 0.0, 13);
        sim_trace(1, "S ws_control");
      } else {
        sim_get_next(next_event);
        sim_putback(next_event);
      }
    }
    }
}

class DBServer extends Sim_entity {
  private Sim_port trans_req_in, trans_reply_out, io_request, io_reply, as_control;
  private Sim_normal_obj DBS_hold;
  private final double DBSmean = 125.0;
  private final double DBSvar = 25.0;
  private int dbThreads;
  private int num_threads=0;
  private boolean FULL = false;
  // Added
  private int loglevel;

  //    public DBServer(String name, int x, int y, int DBSThreads) {
  public DBServer(String name, int x, int y, int DBSThreads, int loglevel) {
    super(name, "databaseserver", x, y);
    this.dbThreads = DBSThreads;
    this.loglevel = loglevel;//added

    trans_req_in = new Sim_port("trans_req_in", "port", Anim_port.LEFT, 10);
    add_port(trans_req_in);
    as_control = new Sim_port("as_control", "port", Anim_port.LEFT, 30);
    add_port(as_control);
    trans_reply_out = new Sim_port("trans_reply_out", "port", Anim_port.LEFT, 70);
    add_port(trans_reply_out);
    io_request = new Sim_port("io_request", "port", Anim_port.BOTTOM, 70);
    add_port(io_request);
    io_reply = new Sim_port("io_reply", "port", Anim_port.BOTTOM, 10);
    add_port(io_reply);

    add_param(new Anim_param("num_threads", Anim_param.VALUE, "0", 1,10));
    DBS_hold = new Sim_normal_obj("DBS_hold", DBSmean, DBSvar);
    add_generator(DBS_hold);
  }

  //Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("DBS  (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

  public void body() {
    Sim_type_p reply = new Sim_type_p(21);
    Sim_type_p request = new Sim_type_p(4);

    FULL = false;
    num_threads = 0;

    Sim_event next_event = new Sim_event();
    sim_get_next(request, next_event);
    num_threads++;
    sim_trace(1,"P "+num_threads);
    if (num_threads == dbThreads) {
      FULL=true;
      sim_schedule(as_control, 0.0, 14);
      sim_trace(1, "S(init) as_control");
    }
```

```
    sim_process(DBS_hold.sample());
    sim_completed(next_event);
    sim_schedule(io_request, 0.0, 20);
    sim_trace(1, "S(init) io_request");

    while (Sim_system.running()) {
      if (sim_waiting(request) > 0 && (!FULL)) {
        sim_select(request, next_event);
        num_threads++;
        sim_trace(1,"P "+num_threads);
        if (num_threads == dbThreads) {
          FULL= true;
          sim_schedule(as_control, 0.0, 14);
          sim_trace(1, "S as_control");
          sim_process(DBS_hold.sample());
          sim_completed(next_event);
          sim_schedule(io_request, 0.0, 20);
          sim_trace(1, "S io_request");
          sim_get_next(reply, next_event);
          sim_process(DBS_hold.sample());
          sim_completed(next_event);
          sim_schedule(trans_reply_out, 0.0, 5);
          sim_trace(1, "S trans_reply_out");
          num_threads--;
          sim_trace(1, "P "+num_threads);
          FULL = false;
          sim_schedule(as_control, 0.0, 15);
          sim_trace(1, "S as_control");
        } else {
          sim_process(DBS_hold.sample());
          sim_completed(next_event);
          sim_schedule(io_request, 0.0, 20);
          sim_trace(1, "S io_request");
        }
      } else if (sim_waiting(reply) > 0) {
        sim_select(reply, next_event);
        // Error 4 missing next line (processing second time)
        sim_process(DBS_hold.sample());
        sim_completed(next_event);
        sim_schedule(trans_reply_out, 0.0, 5);
        sim_trace(1, "S trans_reply_out");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        if (FULL == true) {
          FULL = false;
          sim_schedule(as_control, 0.0, 15);
          sim_trace(1, "S as_control");
        }
      } else if (FULL) {
        sim_get_next(reply, next_event);
        sim_process(DBS_hold.sample());
        sim_completed(next_event);
        sim_schedule(trans_reply_out, 0.0, 5);
        sim_trace(1, "S trans_reply_out");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        FULL = false;
        sim_schedule(as_control, 0.0, 15);
        sim_trace(1, "S as_control");
      } else {
        sim_get_next(next_event);
        sim_putback(next_event);
      }
    }
  }
}
```

```
class Disk extends Sim_entity {
    private Sim_port io_req_in, io_reply_out;
    private Sim_normal_obj IO_hold;
    private final double IOmean = 250.0;
    private final double IOvar = 50.0;
// Added
  private int loglevel;

//    public Disk(String name, int x, int y) {
  public Disk(String name, int x, int y, int loglevel) {
    super(name, "disk", x, y);
    this.loglevel = loglevel;//added

    io_req_in = new Sim_port("io_req_in", "port", Anim_port.TOP, 40);
    add_port(io_req_in);
    io_reply_out = new Sim_port("io_reply_out", "port", Anim_port.TOP, 40);
    add_port(io_reply_out);

    IO_hold = new Sim_normal_obj("IO_hold", IOmean, IOvar);
    add_generator(IO_hold);
    }

//Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("Disk (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

    public void body() {
    Sim_type_p request = new Sim_type_p(20);
    while (Sim_system.running()) {
      Sim_event granting = new Sim_event();
      sim_get_next(request, granting);
      sim_process(IO_hold.sample());
      sim_completed(granting);
      sim_schedule(io_reply_out,0.0,21);
      sim_trace(1, "S io_reply_out");
    }
    }
}

// Main applet
public class ECommerceSite extends Anim_applet {
  public static Panel inputs;
  public static Label WSQsizeLabel, intArrTimeLabel, WSThreadsLabel,
      ASThreadsLabel, DBSThreadsLabel, warmupLabel, runLabel,
      seedLabel; // added for random seed
  public static Choice WSQcapChoice;
  public static TextField iatimeField, WSThreadsField, ASThreadsField,
                        DBSThreadsField, warmupField, runField,
                        seedField; // added for random seed
  private static double warmuptime, runtime;
  private static int seednum; // added for random seed

  // Initialise the GUI
  public void anim_init() {
    int j;
    inputs = new Panel();
    inputs.setLayout(new GridLayout(0,2));

    WSQsizeLabel = new Label("Queueing capacity at Web Server:", Label.RIGHT);
    inputs.add(WSQsizeLabel);
    WSQcapChoice = new Choice();
    for (j = 2; j <= 16; j +=2)
      WSQcapChoice.addItem(" "+j+" ");
```

```
    WSQcapChoice.select(1);
    inputs.add(WSQcapChoice);

    intArrTimeLabel = new Label("Mean Inter-arrival Time (exp. distribution):", Label.RIGHT);
    inputs.add(intArrTimeLabel);
    iatimeField = new TextField("400", 5);
    inputs.add(iatimeField);

    WSThreadsLabel = new Label("No. of Concurrent Threads at Web Server:", Label.RIGHT);
    inputs.add(WSThreadsLabel);
    WSThreadsField = new TextField("8", 2);
    inputs.add(WSThreadsField);

    ASThreadsLabel = new Label("No. of Concurrent Threads at Application Server:", Label.RIGHT);
    inputs.add(ASThreadsLabel);
    ASThreadsField = new TextField("5", 2);
    inputs.add(ASThreadsField);

    DBSThreadsLabel = new Label("No. of Concurrent Threads at Database Server:", Label.RIGHT);
    inputs.add(DBSThreadsLabel);
    DBSThreadsField = new TextField("3", 2);
    inputs.add(DBSThreadsField);

    warmupLabel = new Label("Transient Period (to be discarded):", Label.RIGHT);
    inputs.add(warmupLabel);
    warmupField = new TextField("0", 6);
    inputs.add(warmupField);

    runLabel = new Label("Total Running Time (including warmup):", Label.RIGHT);
    inputs.add(runLabel);
    runField = new TextField("0", 8);
    inputs.add(runField);

// start added
    seedLabel = new Label("Random Seed:", Label.RIGHT);
    inputs.add(seedLabel);
    seedField = new TextField("1", 8);
    inputs.add(seedField);
// end added

    this.add("North", inputs);
  }

  // Setup the animation
  public void anim_layout() {
    int WSQcap, WSThreads, ASThreads, DBSThreads;
    double iatime;

    WSQcap = (((WSQcapChoice.getSelectedIndex())+1)*2);
    iatime = Double.valueOf(iatimeField.getText()).doubleValue();
    WSThreads = Integer.parseInt(WSThreadsField.getText());
    ASThreads = Integer.parseInt(ASThreadsField.getText());
    DBSThreads = Integer.parseInt(DBSThreadsField.getText());
    warmuptime = Integer.parseInt(warmupField.getText());
    runtime = Integer.parseInt(runField.getText());
    seednum = Integer.parseInt(seedField.getText()); // added

// added for apply the seed
    for(int i = 0; i< seednum; i++ ){
      System.out.println("Next Random Seed");
      Sim_system.next_seed();
    }

    // Add entities
    Source source = new Source("Source", 10, 10, iatime,0); // add loglevel
    WSQ wsq = new WSQ("WSQ", 100, 10, WSQcap,0); // add loglevel
    WebServer webserver = new WebServer("WebServer", 220, 10, WSThreads, 0); // add loglevel
```

```
    ApplicationServer appserver = new ApplicationServer("ApplicationServer", 380, 10, ASThreads, 0); // add loglevel
    DBServer dbserver = new DBServer("DatabaseServer", 530, 10, DBSThreads, 0); // add loglevel
    Disk disk = new Disk("DatabaseDisk", 530, 150, 0); // add loglevel

    // Link entities
    Sim_system.link_ports("Source", "enqueue", "WSQ", "request_queue");
    Sim_system.link_ports("WSQ", "access", "WebServer", "grant_thread");
    Sim_system.link_ports("WebServer", "bf_req_out", "ApplicationServer", "bf_req_in");
    Sim_system.link_ports("WebServer", "q_control", "WSQ", "ws_control");
    Sim_system.link_ports("ApplicationServer", "trans_req_out", "DatabaseServer", "trans_req_in");
    Sim_system.link_ports("DatabaseServer", "io_request", "DatabaseDisk", "io_req_in");
    //  Sim_system.link_ports("DatabaseDisk", "io_reply_out", "ApplicationServer", "trans_reply_in"); Error 2
    Sim_system.link_ports("DatabaseDisk", "io_reply_out", "DatabaseServer", "io_reply");
    Sim_system.link_ports("DatabaseServer", "trans_reply_out", "ApplicationServer", "trans_reply_in");
    Sim_system.link_ports("DatabaseServer", "as_control", "ApplicationServer", "dbs_control");
    Sim_system.link_ports("ApplicationServer", "bf_reply_out", "WebServer", "bf_reply_in");
    Sim_system.link_ports("ApplicationServer", "ws_control", "WebServer", "as_control");
    Sim_system.link_ports("WebServer", "exit", "Source", "exit");
  }

  // Setup the simulation's parameters
  public void sim_setup() {
    if (warmuptime > 0.0) {
      Sim_system.set_transient_condition(Sim_system.TIME_ELAPSED, warmuptime);
    }
    Sim_system.set_termination_condition(Sim_system.TIME_ELAPSED, runtime, true);

  }

  // Setup the applet's output
  public void anim_output() {
    generate_messages(true);
    generate_report(true);
  }
}
```

B.2 Part d)

Here the part of the source code what is relevant for task 3d:

part of 3d/ECommerceSite.java

```
/*
 * The E-Commerce Site (animated) -- with errors
 *
 */

/* LOG
  - add variance to WS distribution (Error 1)
  - modify connection from Disk to DBS (Error 2)
  - add overload sim_trace message with loglevel to each entity for testing purpose
  - change the mean time of the distribution of the application server from 1500 to 150 (Error 3)
  - add the second processing of the data at the database server (Error 4)
  - getting the stop message out of the queue at WSQ (Error 5)
  - insert inputbox for random seed and necessary variables
  - insert additional case in WSQ for flushing (Error 6)
  - insert State_Based continuous measure for queue length in WSQ.
  - insert a new Disk and modify the DBServer. (a new out port and a random distr.
      for distribution of the iorequests to disk 1 or 2)
*/
import java.applet.*;
import java.awt.*;
import eduni.simjava.*;
import eduni.simanim.*;
import eduni.simjava.distributions.*;

```

```
class Source extends Sim_entity {
  private Sim_port enqueue, exit;
  private Sim_negexp_obj source_hold;
  private double iatime;
// Added
  private int loglevel;


//  public Source(String name, int x, int y, double iatime) {
  public Source(String name, int x, int y, double iatime, int loglevel) {
    super(name, "source1", x, y);
    this.iatime = iatime;
    this.loglevel = loglevel;//added

    enqueue = new Sim_port("enqueue", "port", Anim_port.RIGHT, 10);
    add_port(enqueue);
    exit = new Sim_port("exit", "port", Anim_port.RIGHT, 70);
    add_port(exit);
    source_hold = new Sim_negexp_obj("iatime", iatime);
    add_generator(source_hold);
  }

  //Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("Src  (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }


  public void body() {
    while (Sim_system.running()) {
        sim_pause(source_hold.sample());
        sim_schedule(enqueue, 0.0, 1);
        sim_trace(1,"S enqueue");
    }
  }
}

class WSQ extends Sim_entity {
  private Sim_port request_queue, access, ws_control;
  private Sim_stat stat;
  private int num_requests, capacity;
  private boolean WS_FULL;
//added
  private int loglevel;

//  public WSQ(String name, int x, int y, int WSQcap) {
  public WSQ(String name, int x, int y, int WSQcap, int loglevel) {

    super(name, "wsq", x, y);
    this.capacity = WSQcap;
    this.loglevel = loglevel;//added

    request_queue = new Sim_port("request_queue", "port", Anim_port.LEFT, 10);
    add_port(request_queue);
    access = new Sim_port("access", "port", Anim_port.RIGHT, 10);
    add_port(access);
    ws_control = new Sim_port("ws_control", "port", Anim_port.RIGHT, 30);
    add_port(ws_control);
    add_param(new Anim_param("num_requests", Anim_param.VALUE, "0", 1, 10));
    stat = new Sim_stat();
    stat.add_measure("Loss rate", Sim_stat.RATE_BASED);
    stat.set_efficient("Loss rate");
    stat.add_measure("WSQueue length", Sim_stat.STATE_BASED, 0.0); // added (continuous state measure)
    set_stat(stat);
```

```
  }

//Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("WSQ  (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

  public void body() {
    Sim_type_p new_request = new Sim_type_p(1);
    Sim_type_p stop_msg = new Sim_type_p(10);
    Sim_type_p start_msg = new Sim_type_p(11);
    int j;

    WS_FULL = false;
    num_requests = 0;

    Sim_event new_event = new Sim_event();
    sim_get_next(new_event);
    sim_schedule(access, 0.0, 2);
    sim_trace(1, "S access");

    while (Sim_system.running()) {
      if ((sim_waiting(stop_msg) > 0) && !(WS_FULL)) {
//added
        sim_select(stop_msg, new_event); // Error 5 (missed line)
        WS_FULL = true;
        continue;
      } else if ((sim_waiting(start_msg) > 0) && (WS_FULL)) {
        sim_select(start_msg, new_event);

        WS_FULL = false;
        j = WebServer.webThreads - WebServer.num_threads;
        if (j > num_requests) j = num_requests;
        while ((j > 0) && Sim_system.running()) {
          sim_schedule(access, 0.0, 2);
          sim_trace(1, "S access");
          num_requests--;
          stat.update("WSQueue length", (double)num_requests, Sim_system.sim_clock());// added
          sim_trace(1, "P "+num_requests);
          j--;
        }
      } else // added for flushing queue if full and Webserver have capacities
            if(num_requests != 0 && (WebServer.webThreads - WebServer.num_threads)>=2 && !(WS_FULL)){
          sim_schedule(access, 0.0, 2);
          sim_trace(1, "S access");
          num_requests--;
          stat.update("WSQueue length", (double)num_requests, Sim_system.sim_clock());// added
          sim_trace(1, "P "+num_requests);
      } else  if (sim_waiting(new_request) > 0) {
         sim_select(new_request, new_event);
        if (!(WS_FULL)){
          sim_schedule(access, 0.0, 2);
          sim_trace(1, "S access");
        } else if (num_requests < capacity) {
          num_requests++;
          stat.update("WSQueue length", (double)num_requests, Sim_system.sim_clock());// added
          sim_trace(1, "P "+num_requests);
        } else {
          stat.update("Loss rate", Sim_system.sim_clock());
        }
      } else {
        sim_get_next(new_event);
        sim_putback(new_event);
      }
```

```
    }
  }
}

class WebServer extends Sim_entity {
  private Sim_port grant_thread, bf_req_out, bf_reply_in, exit, q_control, as_control;
  private Sim_normal_obj WS_hold;
  private final double WSmean = 120;
  private final double WSvar = 15;
  public static boolean FULL;
  public static boolean AS_FULL;
  public static int webThreads, num_threads;
// Added
  private int loglevel;

//  public WebServer(String name, int x, int y, int WSThreads) { // add loglevel
  public WebServer(String name, int x, int y, int WSThreads, int loglevel) {
    super(name, "webserver", x, y);
    this.webThreads = WSThreads;
    this.loglevel = loglevel;//added

    grant_thread = new Sim_port("grant_thread", "port", Anim_port.LEFT, 10);
    add_port(grant_thread);
    bf_req_out = new Sim_port("bf_req_out", "port", Anim_port.RIGHT, 10);
    add_port(bf_req_out);
    as_control = new Sim_port("as_control", "port", Anim_port.RIGHT, 30);
    add_port(as_control);
    bf_reply_in = new Sim_port("bf_reply_in", "port", Anim_port.RIGHT, 70);
    add_port(bf_reply_in);
    exit = new Sim_port("exit", "port", Anim_port.LEFT, 70);
    add_port(exit);
    q_control = new Sim_port("q_control", "port", Anim_port.LEFT, 30);
    add_port(q_control);

    add_param(new Anim_param("num_threads", Anim_param.VALUE, "0", 1,10));
    //  WS_hold = new Sim_normal_obj("WS_hold", WSmean);  error 1
    WS_hold = new Sim_normal_obj("WS_hold", WSmean, WSvar);
    add_generator(WS_hold);
  }
//Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("WS   (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

  public void body() {
    Sim_type_p request = new Sim_type_p(2);
    Sim_type_p reply = new Sim_type_p(6);
    Sim_type_p stop_msg = new Sim_type_p(12);
    Sim_type_p start_msg = new Sim_type_p(13);

    FULL = false;
    AS_FULL = false;
    num_threads = 0;

    Sim_event next_event = new Sim_event();
    sim_get_next(request, next_event);
    num_threads++;
    sim_trace(1,"P "+num_threads);
    if (num_threads == webThreads) {
        FULL=true;
        sim_schedule(q_control, 0.0, 10);
        sim_trace(1, "S q_control");
    }
    sim_process(WS_hold.sample());
```

```
    sim_completed(next_event);
    sim_schedule(bf_req_out, 0.0, 3);
    sim_trace(1, "S bf_req_out");

    while (Sim_system.running()) {
      if ((sim_waiting(stop_msg) > 0) && !(AS_FULL)) {
        sim_select(stop_msg, next_event);
        AS_FULL = true;
        sim_completed(next_event);
      continue;
         } else if ((sim_waiting(start_msg) > 0) && (AS_FULL)) {
        sim_select(start_msg, next_event);
        AS_FULL = false;
        sim_completed(next_event);
      continue;
        } else if (sim_waiting(request) > 0 && (!FULL)) {
        sim_select(request, next_event);
        num_threads++;
        sim_trace(1,"P "+num_threads);
        if (num_threads == webThreads) {
            FULL= true;
            sim_schedule(q_control, 0.0, 10);
            sim_trace(1, "S q_control");
        }
        sim_process(WS_hold.sample());
        sim_completed(next_event);
            if (AS_FULL == true) {
            sim_get_next(start_msg, next_event);
            AS_FULL = false;
            sim_completed(next_event);
        }
        sim_schedule(bf_req_out, 0.0, 3);
        sim_trace(1, "S bf_req_out");
        } else if (sim_waiting(reply) > 0) {
            sim_select(reply, next_event);
        sim_process(WS_hold.sample());
        sim_completed(next_event);
        sim_schedule(exit, 0.0, 7);
        sim_trace(1, "S exit");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        if (FULL == true) {
            FULL = false;
            sim_schedule(q_control, 0.0, 11);
            sim_trace(1, "S q_control");
        }
        } else if (FULL) {
        sim_get_next(reply, next_event);
        sim_process(WS_hold.sample());
        sim_completed(next_event);
        sim_schedule(exit, 0.0, 7);
        sim_trace(1, "S exit");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        FULL = false;
        sim_schedule(q_control, 0.0, 11);
        sim_trace(1, "S q_control");
        } else { //?
        sim_get_next(next_event);
        sim_putback(next_event);
        }
    }
  }
}

class ApplicationServer extends Sim_entity {
    private Sim_port bf_req_in, bf_reply_out, trans_req_out, trans_reply_in, ws_control, dbs_control;
```

```
    private Sim_normal_obj AS_hold;
//    private final double ASmean = 1500; // Error 3
    private final double ASmean = 100;
    private final double ASvar = 50;
    private int appThreads;
    private int num_threads;
    private boolean FULL;
    private boolean DBS_FULL;
  // Added
  private int loglevel;

//    public ApplicationServer(String name, int x, int y, int ASThreads) {
    public ApplicationServer(String name, int x, int y, int ASThreads, int loglevel) {

    super(name, "applicationserver", x, y);
    this.appThreads = ASThreads;
    this.loglevel = loglevel;//added

    bf_req_in = new Sim_port("bf_req_in", "port", Anim_port.LEFT, 10);
    add_port(bf_req_in);
    trans_req_out = new Sim_port("trans_req_out", "port", Anim_port.RIGHT, 10);
    add_port(trans_req_out);
    dbs_control = new Sim_port("dbs_control", "port", Anim_port.RIGHT, 30);
    add_port(dbs_control);
    trans_reply_in = new Sim_port("trans_reply_in", "port", Anim_port.RIGHT, 70);
    add_port(trans_reply_in);
    bf_reply_out = new Sim_port("bf_reply_out", "port", Anim_port.LEFT, 70);
    add_port(bf_reply_out);
    ws_control = new Sim_port("ws_control", "port", Anim_port.LEFT, 30);
    add_port(ws_control);

    add_param(new Anim_param("num_threads", Anim_param.VALUE, "0", 1,10));
    AS_hold = new Sim_normal_obj("AS_hold", ASmean, ASvar);
    add_generator(AS_hold);
    }

//Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("AS   (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

  public void body() {
    Sim_type_p request = new Sim_type_p(3);
    Sim_type_p reply = new Sim_type_p(5);
    Sim_type_p stop_msg = new Sim_type_p(14);
    Sim_type_p start_msg = new Sim_type_p(15);

    FULL = false;
    DBS_FULL = false;
    num_threads = 0;

    Sim_event next_event = new Sim_event();
    sim_get_next(request, next_event);
    num_threads++;
    sim_trace(1,"P "+num_threads);
    if (num_threads == appThreads) {
      FULL= true;
      sim_schedule(ws_control, 0.0, 12);
      sim_trace(1, "S ws_control");
    }
    sim_process(AS_hold.sample());
    sim_completed(next_event);
    sim_schedule(trans_req_out, 0.0, 4);
    sim_trace(1, "S trans_req_out");
```

```
    while (Sim_system.running()) {
      if ((sim_waiting(stop_msg) > 0) && !(DBS_FULL)) {
        sim_select(stop_msg, next_event);
        DBS_FULL = true;
        sim_completed(next_event);
        continue;
      } else if ((sim_waiting(start_msg) > 0) && (DBS_FULL)) {
        sim_select(start_msg, next_event);
        DBS_FULL = false;
        continue;
      } else if ((sim_waiting(request) > 0) && (!FULL)) { //? ()
        sim_select(request, next_event);
        num_threads++;
        sim_trace(1,"P "+num_threads);
        if (num_threads == appThreads) {
          FULL= true;
          sim_schedule(ws_control, 0.0, 12);
          sim_trace(1, "S ws_control");
        }
        sim_process(AS_hold.sample());
        sim_completed(next_event);
        if (DBS_FULL == true) {
          sim_get_next(start_msg, next_event);
          DBS_FULL = false;
          sim_completed(next_event);
        }
        sim_schedule(trans_req_out, 0.0, 4);
        sim_trace(1, "S trans_req_out");
      } else if (sim_waiting(reply) > 0) {
        sim_select(reply, next_event);
        sim_process(AS_hold.sample());
        sim_completed(next_event);
        sim_schedule(bf_reply_out, 0.0, 6);
        sim_trace(1, "S bf_reply_out");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        if (FULL == true) {
          FULL = false;
          sim_schedule(ws_control, 0.0, 13);
          sim_trace(1, "S ws_control");
        }
      } else if (FULL) {
        sim_get_next(reply, next_event);
        sim_process(AS_hold.sample());
        sim_completed(next_event);
        sim_schedule(bf_reply_out, 0.0, 6);
        sim_trace(1, "S bf_reply_out");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        FULL = false;
        sim_schedule(ws_control, 0.0, 13);
        sim_trace(1, "S ws_control");
      } else {
        sim_get_next(next_event);
        sim_putback(next_event);
      }
    }
    }
}

class DBServer extends Sim_entity {
  private Sim_port trans_req_in, trans_reply_out, as_control,
          io_request1, io_request2, io_reply ; // added and modified
  private Sim_normal_obj DBS_hold;
  private Sim_random_obj DBS_disk_dist;
  private final double DBSmean = 125.0;
```

```
  private final double DBSvar = 25.0;
  private int dbThreads;
  private int num_threads=0;
  private boolean FULL = false;
  // Added
  private int loglevel;

  //    public DBServer(String name, int x, int y, int DBSThreads) {
  public DBServer(String name, int x, int y, int DBSThreads, int loglevel) {
    super(name, "databaseserver", x, y);
    this.dbThreads = DBSThreads;
    this.loglevel = loglevel;//added

    trans_req_in = new Sim_port("trans_req_in", "port", Anim_port.LEFT, 10);
    add_port(trans_req_in);
    as_control = new Sim_port("as_control", "port", Anim_port.LEFT, 30);
    add_port(as_control);
    trans_reply_out = new Sim_port("trans_reply_out", "port", Anim_port.LEFT, 70);
    add_port(trans_reply_out);
// modified
    io_request1 = new Sim_port("io_request1", "port", Anim_port.BOTTOM, 50);
    add_port(io_request1);
// added
    io_request2 = new Sim_port("io_request2", "port", Anim_port.BOTTOM, 90);
    add_port(io_request2);

    io_reply = new Sim_port("io_reply", "port", Anim_port.BOTTOM, 10);
    add_port(io_reply);

    add_param(new Anim_param("num_threads", Anim_param.VALUE, "0", 1,10));
    DBS_hold = new Sim_normal_obj("DBS_hold", DBSmean, DBSvar);
    add_generator(DBS_hold);
// added
    DBS_disk_dist = new Sim_random_obj("DBS_disk_dist");
    add_generator(DBS_disk_dist);
  }

  //Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("DBS  (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

  public void body() {
    Sim_type_p reply = new Sim_type_p(21);
    Sim_type_p request = new Sim_type_p(4);

    FULL = false;
    num_threads = 0;

    Sim_event next_event = new Sim_event();
    sim_get_next(request, next_event);
    num_threads++;
    sim_trace(1,"P "+num_threads);
    if (num_threads == dbThreads) {
      FULL=true;
      sim_schedule(as_control, 0.0, 14);
      sim_trace(1, "S(init) as_control");
    }
    sim_process(DBS_hold.sample());
    sim_completed(next_event);
    sim_schedule(io_request1, 0.0, 20); // modified
    sim_trace(1, "S(init) io_request1"); // modified

    while (Sim_system.running()) {
```

```
      if (sim_waiting(request) > 0 && (!FULL)) {
        sim_select(request, next_event);
        num_threads++;
        sim_trace(1,"P "+num_threads);
        if (num_threads == dbThreads) {
          FULL= true;
          sim_schedule(as_control, 0.0, 14);
          sim_trace(1, "S as_control");
          sim_process(DBS_hold.sample());
          sim_completed(next_event);
// added
          // decide which disk
          if(DBS_disk_dist.sample()<0.6){
            sim_schedule(io_request1, 0.0, 20);// modified
            sim_trace(1, "S io_request1"); // modified
          }else{
            sim_schedule(io_request2, 0.0, 20);// added
            sim_trace(1, "S io_request2"); // added
          }

          sim_get_next(reply, next_event);
          sim_process(DBS_hold.sample());
          sim_completed(next_event);
          sim_schedule(trans_reply_out, 0.0, 5);
          sim_trace(1, "S trans_reply_out");
          num_threads--;
          sim_trace(1, "P "+num_threads);
          FULL = false;
          sim_schedule(as_control, 0.0, 15);
          sim_trace(1, "S as_control");
        } else {
          sim_process(DBS_hold.sample());
          sim_completed(next_event);
          // added
          // decide which disk
          if(DBS_disk_dist.sample()<0.6){
            sim_schedule(io_request1, 0.0, 20);// modified
            sim_trace(1, "S io_request1"); // modified
          }else{
            sim_schedule(io_request2, 0.0, 20);// added
            sim_trace(1, "S io_request2"); // added
          }
        }
      } else if (sim_waiting(reply) > 0) {
        sim_select(reply, next_event);
        // Error 4 missing next line (processing second time)
        sim_process(DBS_hold.sample());
        sim_completed(next_event);
        sim_schedule(trans_reply_out, 0.0, 5);
        sim_trace(1, "S trans_reply_out");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        if (FULL == true) {
          FULL = false;
          sim_schedule(as_control, 0.0, 15);
          sim_trace(1, "S as_control");
        }
      } else if (FULL) {
        sim_get_next(reply, next_event);
        sim_process(DBS_hold.sample());
        sim_completed(next_event);
        sim_schedule(trans_reply_out, 0.0, 5);
        sim_trace(1, "S trans_reply_out");
        num_threads--;
        sim_trace(1, "P "+num_threads);
        FULL = false;
        sim_schedule(as_control, 0.0, 15);
```

```
        sim_trace(1, "S as_control");
      } else {
        sim_get_next(next_event);
        sim_putback(next_event);
      }
    }
  }
}

class Disk extends Sim_entity {
  private Sim_port io_req_in, io_reply_out;
  private Sim_normal_obj IO_hold;
  private final double IOmean = 250.0;
  private final double IOvar = 50.0;

// Added
  private int loglevel;
  private Sim_stat stat;

//    public Disk(String name, int x, int y) {
  public Disk(String name, int x, int y, int loglevel) {
    super(name, "disk", x, y);
    this.loglevel = loglevel;//added

    io_req_in = new Sim_port("io_req_in", "port", Anim_port.TOP, 40);
    add_port(io_req_in);
    io_reply_out = new Sim_port("io_reply_out", "port", Anim_port.TOP, 40);
    add_port(io_reply_out);

    IO_hold = new Sim_normal_obj("IO_hold", IOmean, IOvar);
    add_generator(IO_hold);

//added
    stat = new Sim_stat();
    stat.add_measure(Sim_stat.UTILISATION);
    set_stat(stat);

  }

//Added
  public void sim_trace(int level, java.lang.String msg){
    if(loglevel>level){
      System.out.println("Disk (" + (int)(Sim_system.sim_clock()) + "): " + msg );
    }
    super.sim_trace(level, msg);
  }

    public void body() {
    Sim_type_p request = new Sim_type_p(20);
    while (Sim_system.running()) {
      Sim_event granting = new Sim_event();
      sim_get_next(request, granting);
      sim_process(IO_hold.sample());
      sim_completed(granting);
      sim_schedule(io_reply_out,0.0,21);
      sim_trace(1, "S io_reply_out");
    }
    }
}

// Main applet
public class ECommerceSite extends Anim_applet {
  public static Panel inputs;
  public static Label WSQsizeLabel, intArrTimeLabel, WSThreadsLabel,
      ASThreadsLabel, DBSThreadsLabel, warmupLabel, runLabel,
      seedLabel; // added for random seed
  public static Choice WSQcapChoice;
```

```
  public static TextField iatimeField, WSThreadsField, ASThreadsField,
                      DBSThreadsField, warmupField, runField,
                      seedField; // added for random seed
  private static double warmuptime, runtime;
  private static int seednum; // added for random seed

  // Initialise the GUI
  public void anim_init() {
    int j;
    inputs = new Panel();
    inputs.setLayout(new GridLayout(0,2));

    WSQsizeLabel = new Label("Queueing capacity at Web Server:", Label.RIGHT);
    inputs.add(WSQsizeLabel);
    WSQcapChoice = new Choice();
    for (j = 2; j <= 16; j +=2)
      WSQcapChoice.addItem(" "+j+" ");
    WSQcapChoice.select(1);
    inputs.add(WSQcapChoice);

    intArrTimeLabel = new Label("Mean Inter-arrival Time (exp. distribution):", Label.RIGHT);
    inputs.add(intArrTimeLabel);
    iatimeField = new TextField("300", 5); // modified
    inputs.add(iatimeField);

    WSThreadsLabel = new Label("No. of Concurrent Threads at Web Server:", Label.RIGHT);
    inputs.add(WSThreadsLabel);
    WSThreadsField = new TextField("8", 2);
    inputs.add(WSThreadsField);

    ASThreadsLabel = new Label("No. of Concurrent Threads at Application Server:", Label.RIGHT);
    inputs.add(ASThreadsLabel);
    ASThreadsField = new TextField("5", 2);
    inputs.add(ASThreadsField);

    DBSThreadsLabel = new Label("No. of Concurrent Threads at Database Server:", Label.RIGHT);
    inputs.add(DBSThreadsLabel);
    DBSThreadsField = new TextField("3", 2);
    inputs.add(DBSThreadsField);

    warmupLabel = new Label("Transient Period (to be discarded):", Label.RIGHT);
    inputs.add(warmupLabel);
    warmupField = new TextField("4000", 6); // modified
    inputs.add(warmupField);

    runLabel = new Label("Total Running Time (including warmup):", Label.RIGHT);
    inputs.add(runLabel);
    runField = new TextField("500000", 8);  // modified
    inputs.add(runField);

// start added
    seedLabel = new Label("Random Seed:", Label.RIGHT);
    inputs.add(seedLabel);
    seedField = new TextField("1", 8);
    inputs.add(seedField);
// end added

    this.add("North", inputs);
  }

  // Setup the animation
  public void anim_layout() {
    int WSQcap, WSThreads, ASThreads, DBSThreads;
    double iatime;

    WSQcap = (((WSQcapChoice.getSelectedIndex())+1)*2);
    iatime = Double.valueOf(iatimeField.getText()).doubleValue();
```

```
    WSThreads = Integer.parseInt(WSThreadsField.getText());
    ASThreads = Integer.parseInt(ASThreadsField.getText());
    DBSThreads = Integer.parseInt(DBSThreadsField.getText());
    warmuptime = Integer.parseInt(warmupField.getText());
    runtime = Integer.parseInt(runField.getText());
    seednum = Integer.parseInt(seedField.getText()); // added

// added for apply the seed
    for(int i = 0; i< seednum; i++ ){
      System.out.println("Next Random Seed");
      Sim_system.next_seed();
    }

    // Add entities
    Source source = new Source("Source", 10, 10, iatime,0); // add loglevel
    WSQ wsq = new WSQ("WSQ", 100, 10, WSQcap,0); // add loglevel
    WebServer webserver = new WebServer("WebServer", 220, 10, WSThreads, 0); // add loglevel
    ApplicationServer appserver = new ApplicationServer("ApplicationServer", 380, 10, ASThreads, 0); // add loglevel
    DBServer dbserver = new DBServer("DatabaseServer", 530, 10, DBSThreads, 0); // add loglevel
    Disk disk1 = new Disk("DatabaseDisk1", 490, 150, 0); // add loglevel
// added
    Disk disk2 = new Disk("DatabaseDisk2", 590, 150, 0);

    // Link entities
    Sim_system.link_ports("Source", "enqueue", "WSQ", "request_queue");
    Sim_system.link_ports("WSQ", "access", "WebServer", "grant_thread");
    Sim_system.link_ports("WebServer", "bf_req_out", "ApplicationServer", "bf_req_in");
    Sim_system.link_ports("WebServer", "q_control", "WSQ", "ws_control");
    Sim_system.link_ports("ApplicationServer", "trans_req_out", "DatabaseServer", "trans_req_in");
    Sim_system.link_ports("DatabaseServer", "io_request1", "DatabaseDisk1", "io_req_in");
    //  Sim_system.link_ports("DatabaseDisk", "io_reply_out", "ApplicationServer", "trans_reply_in"); Error 2
    Sim_system.link_ports("DatabaseDisk1", "io_reply_out", "DatabaseServer", "io_reply");

// added
    Sim_system.link_ports("DatabaseServer", "io_request2", "DatabaseDisk2", "io_req_in");
    Sim_system.link_ports("DatabaseDisk2", "io_reply_out", "DatabaseServer", "io_reply");

    Sim_system.link_ports("DatabaseServer", "trans_reply_out", "ApplicationServer", "trans_reply_in");
    Sim_system.link_ports("DatabaseServer", "as_control", "ApplicationServer", "dbs_control");
    Sim_system.link_ports("ApplicationServer", "bf_reply_out", "WebServer", "bf_reply_in");
    Sim_system.link_ports("ApplicationServer", "ws_control", "WebServer", "as_control");
    Sim_system.link_ports("WebServer", "exit", "Source", "exit");
  }

  // Setup the simulation's parameters
  public void sim_setup() {
    if (warmuptime > 0.0) {
      Sim_system.set_transient_condition(Sim_system.TIME_ELAPSED, warmuptime);
    }
    Sim_system.set_termination_condition(Sim_system.TIME_ELAPSED, runtime, true);

  }

  // Setup the applet's output
  public void anim_output() {
    generate_messages(true);
    generate_report(true);
  }
}
```