• U  C •

Projecto de Simulação e Computação Científica

Estudo do funcionamento de uma estação de serviço

PL2 - 30 de Abril de 2016

Ana Inês Mesquita Fidalgo 2013134819
Filipa Saraiva Lopes 2013144747
Ana Catarina Gonçalves 2013167088

Introdução

Este trabalho teve como objectivo o estudo do funcionamento de uma Estação de Serviço. A estação de serviço que implementamos é constituída por três bombas (dois para gasolina e uma para gasóleo) e uma loja onde se efetua o pagamento depois de os clientes abastecerem.

Funciona 24 horas por dia, na qual existem doze empregados que trabalham por turnos, estando em cada turno um na loja e três nas bombas. Os clientes chegam em média com um intervalo de tempo entre cada chegada que segue uma distribuição exponencial negativa de média 1.2 minutos. 20% dos clientes pretende abastecer gasóleo e os restantes gasolina.

Essa operação demoraria em ambos os casos, em média, 4 minutos com desvio padrão de 2.5 minutos, segundo uma distribuição normal. O pagamento demora em média 1 minuto com desvio de 0.5 minutos (distribuição normal).

Para realizar este estudo desenvolvemos um simulador em java com estas características e com as características dos diferentes cenários e assim podemos compará-los e escolher o que nós concluímos ser mais eficiente.

Guia de Utilizador

Fig. 1

1- Quando o utilizador corre o simulador é lhe apresentada a figura 1, onde poderá escolher entre três cenários:

Cenário 1:

- Duas máquina na Gasolina.
- Uma máquina no Gasóleo.
- Uma máquina na loja.

Cenário 2:

- Também com três estações mas contem a possibilidade de se poder parametrizar o número de máquinas por estação.

Cenário 3:

- Uma estação self-service, sem a necessidade de ter uma loja onde efetuar o pagamento, com quatro máquinas diferentes.

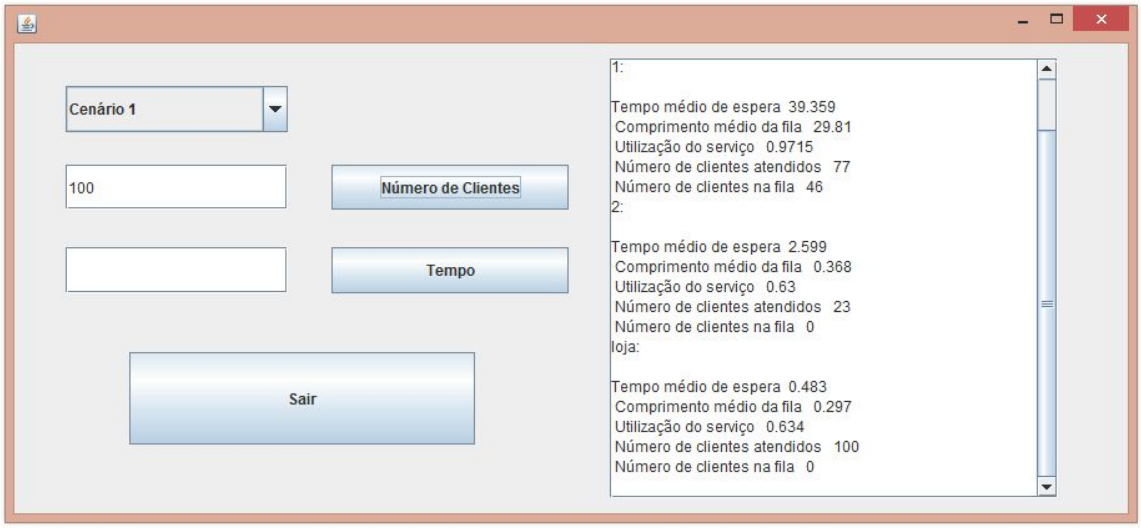


Fig. 2

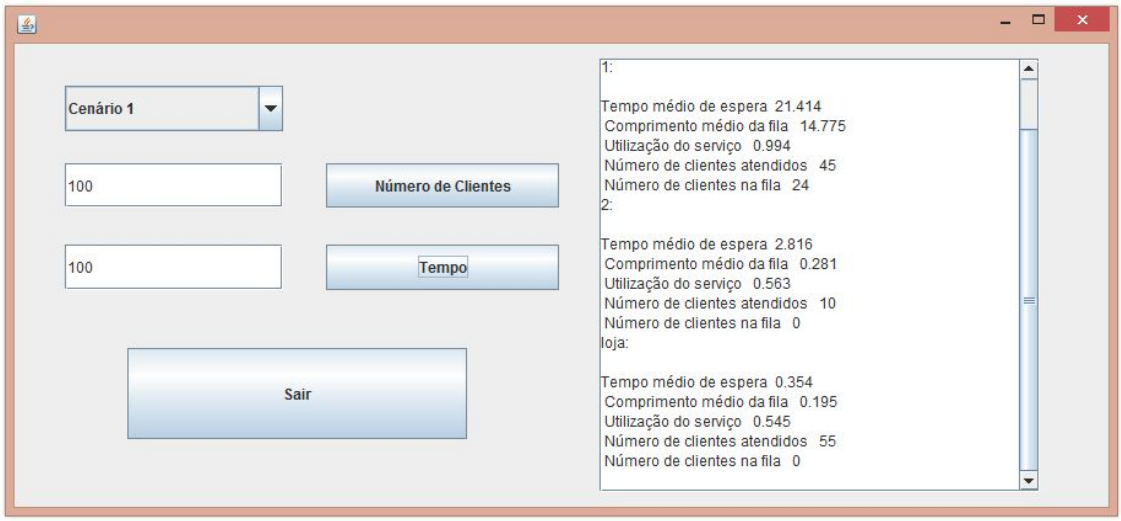


Fig. 3

2- O utilizador poderá escolher correr o simulador por número limite de clientes, como está apresentado na figura 2, ou por tempo limite de simulação, como está apresentado na figura 3. Sendo-lhe depois apresentados os resultados, como o tempo de Simulação, o tempo médio de espera, o comprimento médio da fila, a taxa de utilização de serviço e o número de clientes atendidos e os que estiveram na fila para cada estação.

Guia de Programador

Arquitetura do Simulador

Classe Aleatório

public static int tipoCliente() - Permite-nos determinar a partir da classe RandomGenerator para qual estação o novo cliente irá, se obtiver uma pecentagem superiror a 20% vai para a gasolina e se inferior para o gasóleo;

static double exponencial(double m) - Gera números segundo uma distribuição exponencial negativa;

static double[] normal(double media, double devpad, int seed) - Gera números segundo uma distribuição normal.

Classe Chegada

Esta classe é descendente da classe Evento, herdando as suas propriedades e representa a chegada de um cliente.

void executa () - Verifica qual é o tipo do cliente, se pretende gasolina ou gasóleo, se é tipo 1 ou 3, ou seja, se obtiver uma probabilidade superior a 20% é porque pretende gasolina, senão, pretende gasóleo. Depois da verificação desta probabilidade, o evento é inserido num determinado instante calculado através da distribuição exponencial.

Classe Cliente

Contem apenas uma variável tipo que é calculada através da função tipocliente() na classe Aleatório.

Classe Evento

Esta classe praticamente não sofreu nenhuma alteração. É a classe mãe das classes Chegada, Transição e Saída e contém ainda uma função chamada menor(), para comparação de instantes de eventos.

Classe Interface

Esta classe contém dois Text Fields onde se pode correr o simulador ou por número de clientes ou por tempo de simulação. Contém ainda uma ComboBox para se escolher o cenário pretendido. No cenário 1, o simulador é corrido com os parâmetros originais, ou seja, com 2 máquinas na gasolina, uma no gasóleo e uma na loja de pagamento. No cenário 2 decidimos criar um Spinner para podermos testar com mais casos para além dos pedidos, ou seja, é possível correr o nosso simulador com qualquer número de máquinas desejável para além da proposta de ter três máquinas na gasolina, uma no gasóleo e uma na loja. Temos ainda outro cenário, o 3, parametrizado com 4 máquinas self-service num único serviço.

Classe ListaEventos

Esta classe foi inalterada.

Classe RandomGenerator

Foi inalterada e permite-nos obter tabelas de valores aleatórios.

Classe SCC_PROJECT

Contém a main, na qual é criado o simulador e chamada a interface;

Classe Saida

Esta classe é chamada sempre que um cliente chega à loja para efectuar o pagamento, que irá chamar a função removeServico() localizada na classe

Servico, na qual é decrementar o estado do serviço e caso a fila não esteja vazia, a pessoa que está à frente é atendida, se está fazia não faz nada.

Classe Servico

Cada serviço tem uma média, desvio padrão, um id (1, 2 ou 3) e um tipo de simulação.

public void insereServico (Cliente c) - Esta função é chamada sempre que se quer inserir um serviço numa determinada estação, dependendo do tipo de simulação, ou seja, se é a original que contem 3 estações diferentes ou o cenário 3 na qual apenas existe uma. Esta função vai verificar se está alguma máquina livre, comparando o estado com o número de atendedores, caso este seja igual ao superior quer dizer que todas as máquinas estão ocupadas e adiciona este cliente à fila de espera. Caso esteja livre irá verificar se a flag está a verdadeiro ou falso, se esta estiver a verdadeiro vai calcular um instante, segundo uma distribuição normal, em que se vai realizar ou essa transição ou essa saída, dependendo da estação em que o cliente se encontrava.

public void removeServico - Semelhante à função descrita a cima em termos de implementação, mas permite a remoção de um cliente que já foi processado pela secção em vez de o inserir.

public void act_stats() - função que vai atualizar as estatísticas de um serviço.

public String relat() - função que irá permitir a impressão dos resultados na interface gráfica.

Classe Simulador

Esta classe contém o instante do tempo do sistema, as médias, os serviços, a lista de eventos onde vão ficar registados todos os eventos que vão ocorrer durante a simulação e o tipo de simulação, ou seja, o tipo de cenário.

A funções mais importantes, além do seu construtor e dos gets para aceder às respetivas variáveis, são as seguintes:

void insereEvento (Evento e1) - função de inserção de um novo evento na lista de eventos;

private void act_stats() - atualiza as estatísticas;

private String relat () - função que vai gerar o relatório e onde poderemos ver os resultados na interface gráfica;

public String executa(int maximo, int tipo, int simulationtype, int machineService1, int machineService2, int machineService3) - função que executa a simulação. Vai ter como parâmetros o limite da simulação, que pode ser o número de clientes ou o tempo e isto depende da variável tipo, a variável simulationtype vai definir o cenário em que estamos e no fim esta função vai retornar o relatório da simulação.

Classe Transição

Classe descendente da classe evento, da qual herda as suas propriedades. Esta classe permite a um cliente sair de uma determinada secção e ir ser inserido numa secção seguinte. Aplica-se na Gasolina ou Gasóleo quando o cliente quer efetuar o seu pagamento.

Resultados

- Cenário 1:

Este cenário é o original, onde a Gasolina tem duas máquinas, o gasóleo tem uma e a loja também tem uma. Pela análise dos resultados obtidos (ver anexos) podemos verificar a primeira secção, a Gasolina, obteve sempre tempos de espera elevados e um comprimento médio de fila extenso. Por exemplo para 1500 (em caso de o limite ser temporal) temos aproximadamente tempos de espera na Gasolina de 4.5 minutos, apresentando-se pouco eficiente. Obteve também nesta situação um comprimento médio de fila de 132 pessoas, reforçando assim a sua ineficiência nesta secção. As restantes apresentam tempos de espera e de serviço aceitáveis.

Para calcular os resultados obtidos, calculamos as medidas de desempenho usando os seguintes conhecimentos adquiridos:

1. Tempo médio de espera = Tempo total de espera / Nº total de clientes;
2. Taxa de utilização do serviço = Tempo total de serviço / Tempo de simulação;
3. Comprimento médio da fila = Tempo total de espera / Tempo de simulação;
4. Tempo médio entre chegadas = Soma dos intervalos entre chegadas/ (Nº de chegadas - 1);
5. Probabilidade de espera = Nº de clientes que esperaram / Nº total de clientes;
6. Tempo médio de serviço = Tempo total de serviço / Nº total de clientes;
7. Tempo médio no sistema = Tempo total no sistema / Nº total de clientes

Validação dos Resultados

Para validar os nossos resultados implementamos um código em GPSS com as mesmas características da estação de serviço e comparamos os valores obtidos:

```
************************************************************************
*                                                                      *
*              Super-Market Simulation                                 *
*                                                                      *
************************************************************************
Gasolina  STORAGE 2
Gasoleo   STORAGE 1
Pagamento STORAGE 1

          GENERATE  (Exponential(1,0,1.2));          ;Create next customer.
          TRANSFER  0.2,,DIESEL

GAS       QUEUE     Fila_Gasolina              ;Begin queue time.
          ENTER     Gasolina              ;Own or wait for barber.
          DEPART    Fila_Gasolina              ;End queue time.
          ADVANCE   (Abs(Normal(1,4,2.5)))               ;Haircut takes a few minutes.
          LEAVE     Gasolina              ;Haircut done. Give up the barber.
          TRANSFER  ,CAIXA

DIESEL    QUEUE     Fila_Gasoleo              ;Begin queue time.
          ENTER     Gasoleo              ;Own or wait for barber.
          DEPART    Fila_Gasoleo              ;End queue time.
          ADVANCE   (Abs(Normal(1,4,2.5)))             ;Haircut takes a few minutes.
          LEAVE     Gasoleo              ;Haircut done. Give up the barber.
          TRANSFER  ,CAIXA

CAIXA     QUEUE     Fila_Pagamento              ;Begin queue time.
          ENTER     Pagamento              ;Own or wait for barber.
          DEPART    Fila_Pagamento              ;End queue time.
          ADVANCE   (Abs(Normal(1,1,0.5)))            ;Haircut takes a few minutes.
          LEAVE     Pagamento              ;Haircut done. Give up the barber.
          TERMINATE

          GENERATE 60
          TERMINATE 1     ;Customer leaves.
```

Estação de Serviço - Cenário 1 em GPSS - no generate varia-se o tempo, não sendo o 60 fixo.

Resultados para 3000 minutos:

validation.141.1 - REPORT

```
QUEUE             MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 FILA_GASOLINA    584  583   2016      3   289.773    431.209    431.852   0
 FILA_PAGAMENTO     5    0   1932    884     0.380      0.591      1.089   0
 FILA_GASOLEO       7    1    503    161     0.937      5.590      8.221   0


STORAGE           CAP. REM. MIN. MAX.  ENTRIES AVL.  AVE.C. UTIL. RETRY DELAY
 GASOLINA           2    0   0     2     1433   1    2.000  1.000    0  583
 GASOLEO            1    0   0     1      502   1    0.704  0.704    0    1
 PAGAMENTO          1    0   0     1     1932   1    0.649  0.649    0    0


FEC XN   PRI        BDT      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
  1791    0       3000.089   1791     18     19
```

Fig. 3

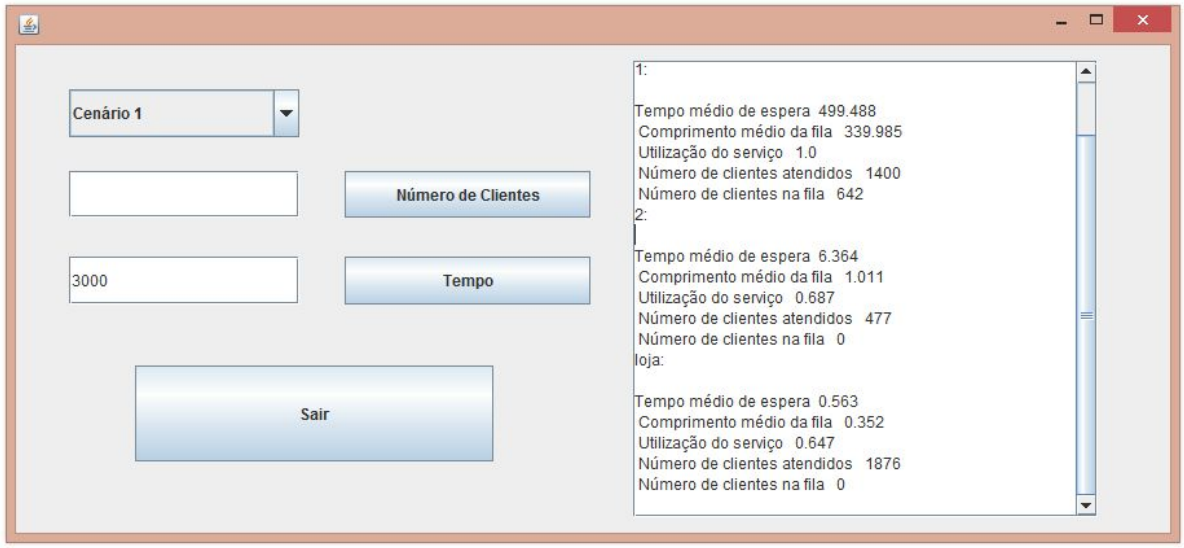


Fig. 4

Para 3000 minutos (180000 segundos) obtivemos resultados semelhantes tanto em GPSS como em Java:

- Na figura 3 obtivemos um total de 1932 clientes e na figura 4 obtivemos 1876.
- Na figura 4 obtivemos na Gasolina um fila de 642, sendo também semelhante na figura 3. O mesmo se verifica para as restantes secções.
- Os de taxa de utilização de serviço são também semelhantes em ambos os casos.

Com isto conseguimos validar a nossa implementação da estação de serviço, como os restantes cenários são diretamente baseados na nossa implementação do cenário original, sendo apenas só necessário alterar o valor de algumas variáveis e etc, obtivemos também resultados válidos ao comparar com o GPSS.

A maneira mais evidente de melhorar a performance deste cenário passa por introduzir um novo posto na Gasolina, com os mesmos tempos de processamento, a qual foi implementada e estudada no cenário dois.

Cenário 2

Neste cenário era pedido que implementássemos um simulador com três postos na primeira secção, a Gasolina, um no Gasóleo e um na loja.

Analisando os resultados e comparando com os valores do cenário 1, podemos ver uma grande melhoria no funcionamento e no atendimento dos clientes ao fornecer-lhes mais um posto de abastecimento para a Gasolina. Supondo também que o limite temporal é 1500, concluímos que no posto de Gasolina tivemos uma melhoria de 132 de comprimento médio de fila para aproximadamente 12.

Podemos ver esta melhoria em várias situações, principalmente no posto de Gasolina. No posto de Gasóleo há também uma descida destes valores, quer de tempo médio de espera, quer de comprimento médio da fila, porém a diferença não é tão significativa.

Cenário 3

No cenário 3 deixamos de diferenciar os três postos e passamos a ter uma estação com 4 máquinas Self-Service. Um cliente entra, vê se uma das máquinas está livre, senão entra na fila de espera.

Ao analisar os resultados obtidos, facilmente percebemos que este era o cenário mais eficiente em termos temporais.

Comparando a mesma situação em que o limite temporal é 1500, o comprimento médio de fila passou a ser aproximadamente 1, contudo verificamos que foram atendidos menos clientes em relação ao cenário 2, pois as máquinas são mais lentas comparativamente com os cenários restantes. Neste cenário temos ainda uma redução de custos com os salários, pois comparativamente com os outro cenários, há uma redução significativa de funcionários, contendo apenas dois funcionários e gastando apenas 1000€ mensalmente em ordenados.

Destes dois cenários escolheria a 2ª opção pois embora as máquinas sejam mais lentas e mais caras, pela análise de resultados concluímos que os tempos de espera são muito inferiores ao cenário 1 e 2 por causa das máquinas Self-Service, e conforme analisamos no ponto seguinte, tornasse mais rentável ao fim de dois anos. Uma nova solução passaria por ter 4 máquinas na Gasolina,

uma no Gasóleo e uma na loja, pois como a a probabilidade é de 4:1 (80% para Gasolina e 20% para o Gasóleo) eliminar-se-ia a discrepância e reduzir-se-iam as filas.

Resultados

Cenário 1

- Por número de clientes

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
100 clientes	Tempo de simulação	162.398	Gasolina	39,359	29,81	0,9715	77	46
			Gasóleo	2,599	0,368	0,63	23	0
			Loja	0,483	0,297	0,634	100	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
250 clientes	Tempo de simulação	392.049	Gasolina	72,859	47,947	0,9935	182	76
			Gasóleo	5,023	0,884	0,795	68	1
			Loja	0,624	0,398	0,677	250	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
500 clientes	Tempo de simulação	794.517	Gasolina	119,231	82,537	0,9925	351	199
			Gasóleo	5,169	0,988	0,766	149	3
			Loja	0,651	0,41	0,655	500	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
750 clientes	Tempo de simulação	1157.277	Gasolina	185,536	122,806	0,997	536	230
			Gasóleo	11,391	2,106	0,83	214	0
			Loja	0,568	0,368	0,67	750	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
1000 clientes	Tempo de simulação	1617.768	Gasolina	301,672	209,783	0,9995	749	376
			Gasóleo	4,105	0,636	0,654	251	0
			Loja	0,58	0,358	0,634	1000	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
1200 clientes	Tempo de simulação	1938.141	Gasolina	337,331	231,311	0,999	889	440
			Gasóleo	5,629	0,926	0,692	311	8
			Loja	0,525	0,325	0,642	1200	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
1500 clientes	Tempo de simulação	2415.658	Gasolina	365.136	252.275	0.9995	1137	532
			Gasóleo	4.311	0.645	0.652	363	1
			Loja	0.512	0.318	0.641	1500	0

- Por tempo de simulação

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 100	Clientes Atendidos:	69	Gasolina	19.283	12.534	0.9915	49	16
			Gasóleo	17.247	3.621	0.844	20	1
			Loja	0.653	0.45	0.715	69	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 250	Clientes Atendidos:	148	Gasolina	51,23	37,705	1,0045	110	73
			Gasóleo	2,924	0,444	0,655	38	0
			Loja	0,453	0,268	0,598	148	0

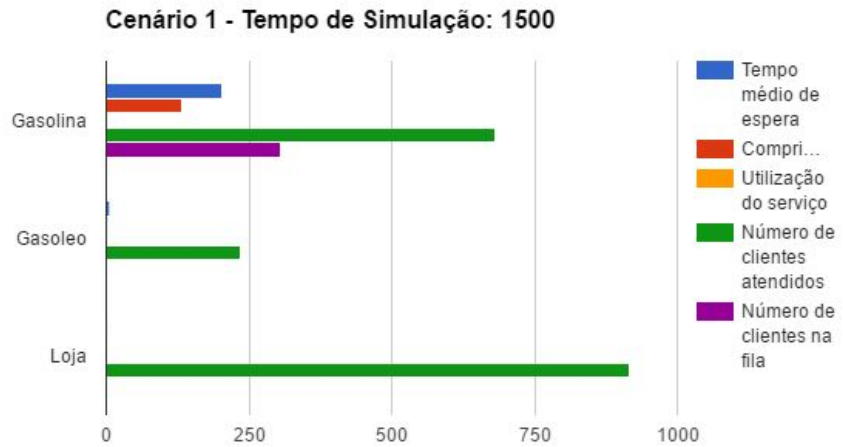
				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 500	Clientes Atendidos:	294	Gasolina	69,894	42,216	0,973	219	83
			Gasóleo	5,201	0,78	0,677	75	0
			Loja	0,479	0,282	0,595	294	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 750	Clientes Atendidos:	467	Gasolina	127,327	88,959	0,995	355	169
			Gasóleo	3,75	0,585	0,593	112	5
			Loja	0,438	0,273	0,625	467	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1000	Clientes Atendidos:	629	Gasolina	152,683	101,228	0,9855	460	203
			Gasóleo	5,702	0,963	0,724	169	0
			Loja	0,58	0,365	0,658	629	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1200	Clientes Atendidos:	730	Gasolina	196,386	131,251	0,999	549	253
			Gasoleo	6,308	0,956	0,645	182	0
			Loja	0,421	0,256	0,608	730	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1500	Clientes Atendidos:	915	Gasolina	201,711	132,725	0,9995	681	306
			Gasóleo	4,815	0,751	0,66	234	0
			Loja	0,522	0,318	0,627	915	0



Cenário 2

- Por número de clientes

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
100 clientes	Tempo de simulação	137.436	Gasolina	2.721	1,663	0,895	83	1
			Gasóleo	0.996	0,123	0,528	17	0
			Loja	0,668	0,486	0,706	100	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
250 clientes		328.161	Gasolina	8.718	5.579	0.878	199	11
			Gasóleo	4.795	0.774	0.744	51	2
			Loja	1,011	0.77	0.771	250	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
500 clientes	Tempo de simulação	612.582	Gasolina	7.313	4.859	0.90733	401	6
			Gasóleo	7.307	1.264	0.717	106	0
			Loja	1.927	1.592	0.857	500	6

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
750 clientes	Tempo de simulação	907.551	Gasolina	35.923	25.451	0.98	603	40
			Gasóleo	3.635	0.588	0.691	147	0
			Loja	1.521	1.257	0.856	750	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
1000 clientes	Tempo de simulação	1217.327	Gasolina	24.592	16.161	0.9566	797	3
			Gasóleo	6.833	1.162	0.734	203	4
			Loja	1.481	1.216	0.832	1000	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
1200 clientes	Tempo de simulação	1480.862	Gasolina	7.056	4.593	0.90133	952	12
			Gasóleo	6.637	1.111	0.701	248	0
			Loja	2.155	1.746	0.814	1200	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
1500 clientes	Tempo de simulação	1874,7	Gasolina	4.517	2.928	0.903	1209	6
			Gasóleo	4.747	0.744	0.654	291	3
			Loja	1.877	1.502	0.831	1500	0

-Por tempo de simulação

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 100	Clientes Atendidos:	75	Gasolina	10,354	7,144	0,8883	55	14
			Gasóleo	4,84	0,774	0,651	15	1
			Loja	1,304	0,913	0,719	70	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 250	Clientes Atendidos:	200	Gasolina	7,211	4,961	0,9306	159	13
			Gasóleo	2,977	0,488	0,657	41	0
			Loja	1,704	1,363	0,858	200	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 500	Clientes Atendidos:	395	Gasolina	9,475	5,988	0,9703	313	3
			Gasóleo	11,848	1,943	0,716	82	0
			Loja	2,021	1,597	0,815	395	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 750	Clientes Atendidos:	653	Gasolina	10,458	7,348	0,9863	519	8
			Gasóleo	8,152	1,467	0,749	134	1
			Loja	2,647	2,304	0,89	653	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1000	Clientes Atendidos:	818	Gasolina	12,053	8,22	0,9456	670	12
			Gasóleo	4,154	0,639	0,626	152	2
			Loja	1,507	1,237	0,831	818	3

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1200	Clientes Atendidos:	987	Gasolina	18,394	12,63	0,9456	789	35
			Gasóleo	4,131	0,681	0,687	198	0
			Loja	1,63	1,34	0,849	987	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1500	Clientes Atendidos:	1257	Gasolina	17,933	11,979	0,9593	1002	0
			Gasóleo	12,207	2,075	0,725	255	0
			Loja	2,758	2,311	0,866	1257	0

Cenário 3

-Por número de clientes

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
100 clientes	Tempo de simulação	151,631	Self-service	1.022	0.674	0.7505	100	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
250 clientes	Tempo de simulação	342.732	Self-service	2016/02/11	1.534	0.82325	250	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
500 clientes	Tempo de simulação	722.032	Self-service	2.021	2016/01/04	0.78325	500	1

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
750 clientes	Tempo de simulação	1113.387	Self-service	2.703	1.83	0.7745	750	3

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
1000 clientes	Tempo de simulação	1505.458	Self-service	1.412	0.941	0.75075	1000	3

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
1200 clientes	Tempo de simulação	1805.921	Self-service	1.405	0.936	0.74675	1200	4

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos
1500 clientes	Tempo de simulação	2325.233	Self-service	2.321	1.502	0.7375	1500

-Por tempo de simulação

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 100	Clientes Atendidos:	54	Self-service	1,145	0,618	0,68875	54	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 250	Clientes Atendidos:	145	Self-service	0,567	0,328	0,66275	145	0

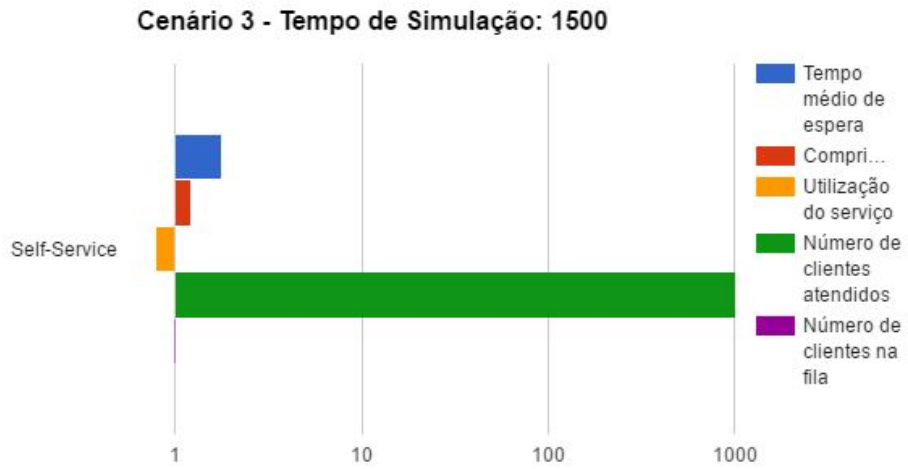
				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 500	Clientes Atendidos:	327	Self-service	1,687	1,103	0,784	327	0

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 750	Clientes Atendidos:	458	Self-service	0,858	0,527	0,702	458	3

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1000	Clientes Atendidos:	650	Self-service	1,302	0,851	0,74425	650	4

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1200	Clientes Atendidos:	857	Self-service	3,267	2,347	0,818	857	5

				Tempo médio de espera	Comprimento médio da fila	Utilização do serviço	Número de clientes atendidos	Número de clientes na fila
Tempo de simulação: 1500	Clientes Atendidos:	1010	Self-service	1,793	1,208	0,779	1010	1



Rendimentos

O cenário 2 apresenta um investimento de 3000€, sendo que o cenário 3 apresenta 3.5 vezes esse valor em investimento e que cada cliente atendido contribui em média com 1.5€ para amortização desse investimento numa primeira fase e numa fase posterior de lucro para a empresa. Relativamente a ordenados, no cenário 2, temos o mesmo número de funcionários que no cenário original, ou seja, 12 funcionários, o que faz um total de 500x12 € gastos

mensalmente, enquanto que no cenário 3, a estação de serviço com apenas uma estação e 4 máquinas, tem apenas dois funcionários a supervisionar as máquinas, o que representa um gasto de 1000€ em ordenados.

1. São necessárias 2000 vendas no cenário 2 para se tornar rentável não incluindo ordenados.
2. São necessárias 7000 vendas no cenário 3 para se tornar rentável não incluindo o dinheiro gasto em ordenados.

lucro pelos clientes despesas investimento

- Cenário 2 - ((nclientes/por mes * mês *1.5) - 500*12 *mês) - 3000
- Cenário 3 - ((nclientes/por mês* mês *1.5) - 500*2 *mês) - 1000

Depois de realizar uma simulação obtivemos os seguintes resultados:

2	clientes total	clientes por mes	Despesas com funcionarios
1 mes	35824	35824	6000
2 mes	72355	36531	6000
3 mes	108760	36405	6000
4 mes	143202	34442	6000
5 mes	180014	36812	6000
6 mes	215917	35903	
7 mes			salarios: 36000

3	clientes total	clientes por mes	Despesas com funcionarios
1 mes	28346	28346	1000
2 mes	57516	29170	1000
3 mes	86168	28652	1000
4 mes	115357	29189	1000
5 mes	143969	28612	1000
6 mes	173035	29066	
7 mes			salarios: 6000

PS: Consideramos que o cenário 2 tinha o mesmo número de funcionários que o cenário original.

Para além além das despesas com os funcionários é necessário calcular quando é que o investimento das máquinas é pago.

Sendo o cenário 3 mais lento comparativamente ao cenário 2 e tendo por isso normalmente menos clientes atendidos por mês, a grande desvantagem do cenário 2 é o facto de ter tantos funcionários e cada um retirar ao lucro mensal 500€, neste aspecto o cenário 3 é mais eficiente pois tem apenas dois funcionários a supervisionar as máquinas, o que lhe permite manter uma maior

quantidade de lucro mensal a partir do momento em que paga o seu investimento.

Concluímos que a partir do momento em que o cenário 3 consegue pagar o investimento realizado de 10000 €, fará mais lucro que o cenário 2, ainda que consiga um pouco menos de clientes mensalmente, pois o 2 tem de pagar a cada empregado 500 €, sendo 12 empregados, o que equivale a uma despesa de 6000 todos os meses em ordenados. Pelos nossos cálculos, através de duas funções do cálculo do lucro, cada uma correspondente a cada cenário, intersectamo-las e chegamos à conclusão que mais ou menos ao fim de 2 anos (23 meses) o Cenário 3 torna-se mais rentável que o Cenário 2, sendo por isso mais eficiente daí para a frente.

Etapas do Projecto e Distribuição de Tarefas

Etapas do Projecto	Elementos	Estado
Código Java	Nome	Terminado
	Inês Fidalgo	Sim
Relatório	Nome	
Testes e Resultados	Inês Fidalgo	Sim
Graficos	Catarina Gonçalves	Sim
Análise de Resultados	Catarina, Inês, Filipa	Sim
Alinea D)	Inês, Catarina, Filipa	Sim
Diagram de Gantt	Inês Fidalgo	Sim
Código GPSS	Nome	
	Inês, Catarina, Filipa	Sim

Conclusão

Com este trabalho podemos testar e comparar diversos cenários de organização de uma estação de serviço, aplicando e aprofundando os conhecimentos adquiridos ao longo das aulas.

Código

Classe Aleatorio

```
package scc_project;

// Classe para gera��o de n�meros aleat�rios segundos v�rias distribui��es

import java.util.Random;

// Apenas a distribui��o exponencial negativa est� definida

public class Aleatorio {


	public static int tipoCliente(){
	//isto está mal /ines corrigido

	//int random  = new Random().nextInt();
	if(RandomGenerator.rand64(10)>0.20){
	return 1;
	}
	else{
	return 3;
	}

	}

	// Gera um número segundo uma distribuição exponencial negativa de média m
	static Double[] myArray = new Double[3]; //armazena os valores se ainda houverem
	static Boolean if1 = false; //verifica se há numeros
	static Boolean if2 = false; //verifica se há numeros
	static Boolean if3 = false; //verifica se há numeros

	static double exponencial(double m) {

	return (-m * Math.log(RandomGenerator.rand64(0)));
	}

	static double[] normal(double media, double devpad, int seed)// o id verifica que seccao pede o
aleatorio
	{
	double v1, v2, w, y1, y2, temp, u1, u2;
	double [] x = new double [2];
	do{
```

```
        do {
        u1 = RandomGenerator.rand(seed);
        u2 = RandomGenerator.rand(seed);
        v1 = 2 * u1 - 1;
        v2 = 2 * u2 - 1;
        w = v1 * v1 + v2 * v2;
        } while (w > 1);

        temp = Math.pow(((-2) * Math.log(w) / w), 0.5);
        y1 = v1 * temp;
        y2 = v2 * temp;

        x[0] = media + y1 * devpad;
        x[1] = media + y2 * devpad;
        }while (x[0]<0 || x[1]<0);


        return x;

        }


}
```

Classe Chegada

```
package scc_project;


public class Chegada extends Evento {

        Chegada (double i, Simulador s, Servico serv, Servico fina, Cliente c){
         super (i, s, serv,fina,c);

   }


        void executa (){

        Cliente c = new Cliente();
        if (c.getTipo()==1) s.getServ1().insereServico(c);
        else s.getServ3().insereServico(c);
        s.insereEvento (new Chegada(s.getInstante()+Aleatorio.exponencial(s.getMedia_cheg()), s,
serv,target,c));
        }


        public String toString(){
        return "Chegada em " + instante;
        }
}
```

Classe Cliente

```
package scc_project;


public class Cliente {
	protected int tipo;

	public Cliente (){
	tipo = Aleatorio.tipoCliente();
	}


	public int getTipo(){
	return tipo;
	}
}
```

Classe Evento

```
package scc_project;


public abstract class Evento {

   protected double instante;
   protected Simulador s;
	protected Servico serv;
	protected Servico target;
	Cliente c;


	Evento (double i, Simulador s, Servico serv, Servico target, Cliente c){
	 instante = i;
	 this.s = s;
	this.serv = serv;
	this.target = target;
	this.c = c;
   }


	public boolean menor (Evento e1){
	 return (instante < e1.instante);
   }



	abstract void executa ();
```

```
	public Servico getTarget() {
	return target;
	}

	public double getInstante() {
	return instante;
	}
	public Cliente getC() {
	return c;
	}
}
```

Classe ListaEventos

```
package scc_project;

import java.util.*;



public class ListaEventos extends LinkedList<Evento> {

	private Simulador s;
	private static final long serialVersionUID = 1;


	ListaEventos (Simulador s){
	this.s = s;
	}


	public void insereEvento (Evento e1){
   int i = 0;

   while (i < size() && ((Evento)get(i)).menor(e1)) i++;

	add(i, e1);
	}


	public void print (){
	int i;
	System.out.println ("--- Lista de eventos em " + s.getInstante() + " ---");
	for (i = 0; i < size(); i++) System.out.println ("Evento " + (i+1) + " ♦ uma " + (Evento)(get(i)));
```

```
        }
}
```

Classe RandomGenerator

```
package scc_project;

/* Prime modulus multiplicative linear congruential pseudo random number generator

   Code copied from book "Simulation Modeling and Analysis, second edition,
   Averill M. Law and W. David Kelton, McGraw-Hill, 1991"

   Z[i] = (630360016 * Z[i-1]) (mod(pow(2,31) - 1)), based on Marse and
   Roberts' portable FORTRAN random-number generator UNIRAN. Multiple
   (100) streams are supported, with seeds spaced 100,000 apart.
   Throughout, input argument "stream" must be an int giving the
   desired stream number.

   Usage: (three functions)

   1. To obtain the next U(0,1) random number from stream "stream,"
         execute
         u = MarseRoberts.rand( stream )
         where rand is a double function. The double variable u will
         contain the next random number.

         Note: there is an equivalent function rand64() which generates the same
         value using Java's 64 bit arithmetic. This was added by me.

   2. To set the seed for stream "stream" to desired value zset,
         execute
         MarseRoberts.randst( zset, stream );
         where zset must be an integer to the desired seed,
         a number between 1 and 2147483646 (inclusive).
         Default seeds for all 100 streams are given in the code.

   3. To get the current (most recently used) integer in the sequence
         being generated for stream "stream" into the int variable zget,
         execute
         zget = MarseRoberts.randgt( stream );
         where randgt is a int function.

   Validation:

   To get an idea whether this port works correctly, rand() and
   rand64() were called 1000 times for each of the 100 streams.
   The two seed tables zrng and zrng64 were then compared using
   compareSeeds(). They contained identical values.

   Performance:
```

```
  For each of the 100 streams, 100 random numbers were generated. This
  was timed, once using rand64() and then using rand(). The results are as follows:

  Metrowerks Java (Metrowerks CodeWarrior Academic Pro 11 for Mac OS):

  rand64()        rand()          speed up by using rand64 (time(rand) / time(rand64))
  ---------       --------  ------------------------------------------------------
  1604.4 ms       210.4 ms        0.131

  Netscape Navigator 3.01 for Macintosh:

  rand64()        rand()
  --------        ------    ------------------------------------------------------
  289.1 ms        428.1           1.481

  (Average of 7 runs on a PowerMacintosh 6100, 66 MHZ and second level cache):
*/

public class RandomGenerator {

  // Default seeds for all 100 streams
  //
  private static int zrng[] = {
          1973272912,  281629770,   20006270, 1280689831, 2096730329, 1933576050,
          913566091,  246780520, 1363774876,  604901985, 1511192140, 1259851944,
          824064364,  150493284,  242708531,   75253171, 1964472944, 1202299975,
          233217322, 1911216000,  726370533,  403498145,  993232223, 1103205531,
          762430696, 1922803170, 1385516923,   76271663,  413682397,  726466604,
          336157058, 1432650381, 1120463904,  595778810,  877722890, 1046574445,
          68911991, 2088367019,  748545416,  622401386, 2122378830,  640690903,
          1774806513, 2132545692, 2079249579,   78130110,  852776735, 1187867272,
          1351423507, 1645973084, 1997049139,  922510944, 2045512870,  898585771,
          243649545, 1004818771,  773686062,  403188473,  372279877, 1901633463,
          498067494, 2087759558,  493157915,  597104727, 1530940798, 1814496276,
          536444882, 1663153658,  855503735,   67784357, 1432404475,  619691088,
          119025595,  880802310,  176192644, 1116780070,  277854671, 1366580350,
          1142483975, 2026948561, 1053920743,  786262391, 1792203830, 1494667770,
          1923011392, 1433700034, 1244184613, 1147297105,  539712780, 1545929719,
          190641742, 1645390429,  264907697,  620389253, 1502074852,  927711160,
          364849192, 2049576050,  638580085,  547070247
  };

  // This table was used to validate the generator
  //
  /*
  private static int zrng64[] = {
          1973272912,  281629770,   20006270, 1280689831, 2096730329, 1933576050,
          913566091,  246780520, 1363774876,  604901985, 1511192140, 1259851944,
          824064364,  150493284,  242708531,   75253171, 1964472944, 1202299975,
          233217322, 1911216000,  726370533,  403498145,  993232223, 1103205531,
          762430696, 1922803170, 1385516923,   76271663,  413682397,  726466604,
```

```
        336157058, 1432650381, 1120463904,  595778810,  877722890, 1046574445,
        68911991, 2088367019,  748545416,  622401386, 2122378830,  640690903,
        1774806513, 2132545692, 2079249579,   78130110,  852776735, 1187867272,
        1351423507, 1645973084, 1997049139,  922510944, 2045512870,  898585771,
        243649545, 1004818771,  773686062,  403188473,  372279877, 1901633463,
        498067494, 2087759558,  493157915,  597104727, 1530940798, 1814496276,
        536444882, 1663153658,  855503735,   67784357, 1432404475,  619691088,
        119025595,  880802310,  176192644, 1116780070,  277854671, 1366580350,
        1142483975, 2026948561, 1053920743,  786262391, 1792203830, 1494667770,
        1923011392, 1433700034, 1244184613, 1147297105,  539712780, 1545929719,
        190641742, 1645390429,  264907697,  620389253, 1502074852,  927711160,
        364849192, 2049576050,  638580085,  547070247
};
*/

// The magic constants
//
private static final int MODLUS  = 2147483647;
private static final int MULT1   = 24112;
private static final int MULT2   = 26143;


// Generate the next random number using 64 bit arithmetic.
// (A Java 'long' is 64 bits long.)
//
public static double rand64( int stream ) {

        // This version was used to validate the generator (using the zrng64 table)
        /*
        long zi = ((long)630360016 * (long)zrng64[stream]) % (long)MODLUS;
        zrng64[stream] = (int) zi;
        */

        long zi = ((long)630360016 * (long)zrng[stream]) % (long)MODLUS;
        zrng[stream] = (int) zi;

        // return (double)zi / (double)MODLUS;
        return (double)((zi >> 7 | 1) + 1) / 16777216.0;
}

// Compare the two seed tables; returns true if the tables are identical.
// This function was used to validate the generator.
//
/*
public static boolean compareSeeds() {
        for (int i=0; i<100; i++) {
        if (zrng[i] != zrng64[i]) return false;
        }
        return true;
}
*/
```

```
 // Generate the next random number.
 //
 public static double rand( int stream ) {
        int zi, lowprd, hi31;

        zi = zrng[stream];
        lowprd = (zi & 65535) * MULT1;
        hi31 = (zi >> 16) * MULT1 + (lowprd >> 16);
        zi = ((lowprd & 65535) - MODLUS) + ((hi31 & 32767) << 16) + (hi31 >> 15);
        if (zi < 0) zi += MODLUS;
        lowprd = (zi & 65535) * MULT2;
        hi31 = (zi >> 16) * MULT2 + (lowprd >> 16);
        zi = ((lowprd & 65535) - MODLUS) + ((hi31 & 32767) << 16) + (hi31 >> 15);
        if (zi < 0) zi += MODLUS;
        zrng[stream] = zi;

        return (double)((zi >> 7 | 1) + 1) / 16777216.0;
 }


 // Set the current zrng for stream "stream" to zset.
 //
 public static void randst( int zset, int stream ) {
        zrng[stream] = zset;
 }


 // Return the current zrng for stream "stream".
 //
 public static int randgt( int stream ) {
        return zrng[stream];
 }
}
```

Classe SCC_PROJECT

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package scc_project;

/**
 *
 * @author utilizador
 */
public class SCC_Project {

        /**
```

```
        * @param args the command line arguments
        */
        public static void main(String[] args) {
        Simulador simulator = new Simulador();
        //simulator.executa(1000,1,1,3,1,1);

        new Interface(simulator).setVisible(true);
        }

}
```

Classe Saida

```
package scc_project;



public class Saida extends Evento {


   Saida (double i, Simulador s, Servico serv, Servico fina, Cliente c){
            super(i, s, serv,fina,c );
   }


   void executa (){

        serv.removeServico();
        }


        public String toString(){
        return "Sa�da em " + instante;
        }


}
```

Classe Servico

```
package scc_project;

import java.util.*;
```

```
// Classe que representa um servi□o com uma fila de espera associada

public class Servico {
   private int estado; // Vari□vel que regista o estado do servi□o: 0 - livre; 1 - ocupado
        private int atendidos;
        private double temp_ult, soma_temp_esp, soma_temp_serv;
   private Vector<Cliente> fila;
   private Simulador s;
        private int atendedores;
        private int numero;
        private Servico next = null;
        private double media; //media
        private double desvio; //desvio
        private int simulationType;
        double [] valores = new double [2];
        boolean flag = true;

   // Construtor
        Servico (Simulador s, int numero, double media, double desvio, Servico next, int n, int
simulationtype){
         this.s = s;
        this.next = next;
        fila = new Vector <Cliente>();
        estado = 0; // Livre
        temp_ult = s.getInstante();
        atendidos = 0;
        soma_temp_esp = 0;
        soma_temp_serv = 0;
        this.atendedores=n;
        this.media = media;
        this.desvio = desvio;

        this.numero = numero;

        this.simulationType = simulationtype;

        }



        public void insereServico (Cliente c){
        if(simulationType==1){

        //System.out.println("numero1"+numero);

        if (estado <atendedores) {
        //System.out.println("entrou");
        estado++;
        //System.out.println("estado:"+estado);
        //System.out.println("numero"+numero);
        if (flag) {
```

```
            valores = Aleatorio.normal(media, desvio,1);
            flag = false;
}
else {
            valores [0] = valores [1];
            flag = true;
}

if (numero == 1) {
            //System.out.println("transitou1");
            s.insereEvento(new Transicao(s.getInstante() + valores[0], s, this, this.next, c));
} else if (numero == 3) {
            //System.out.println("transitou2");
            s.insereEvento(new Transicao(s.getInstante() + valores[0], s, this, this.next, c));
} else if (numero == 2) {

            //System.out.println("SAIU");
            s.insereEvento(new Saida(s.getInstante() + valores[0], s, this, this.next, c));
}

} else {
//System.out.println("adicionou a fila!!!!!!!!!!!");
fila.addElement(c);
}
}
else{
if (estado <atendedores) {
//System.out.println("entrou");
estado++;
//System.out.println("estado:"+estado);
//System.out.println("numero"+numero);
if (flag) {
            valores = Aleatorio.normal(media, desvio,1);
            flag = false;
}
else {
            valores [0] = valores [1];
            flag = true;
}


s.insereEvento(new Saida(s.getInstante() + valores[0], s, this, this.next, c));


} else {

fila.addElement(c);
}
}
}
```

```
	public void removeServico (){
if(simulationType==1){
	atendidos++;
	//System.out.println("ATENDIDOS"+atendidos);

	if (fila.size() == 0) {
	estado--;
	}
	else {
	Cliente c;
	c = fila.get(0);
	fila.removeElementAt(0);
	if (flag) {
		valores = Aleatorio.normal(media, desvio,1);
		flag = false;
	}
	else {
		valores [0] = valores [1];
		flag = true;
	}


	if (numero == 1) {


		s.insereEvento(new Transicao(s.getInstante() + valores[0], s, this, next, c));

	} else if (numero == 3) {

		s.insereEvento(new Transicao(s.getInstante() + valores[0], s, this, next, c));

	} else if (numero == 2) {

		s.insereEvento(new Saida(s.getInstante() + valores[0], s, this, next, c));
	}

	}
	}
	else{
	atendidos++;
	//System.out.println("ATENDIDOS"+atendidos);

	if (fila.size() == 0) {
	estado--;
	}
	else {
	Cliente c;
	c = fila.get(0);
	fila.removeElementAt(0);
	if (flag) {
```

```
            valores = Aleatorio.normal(media, desvio,1);
            flag = false;
            }
            else {
            valores [0] = valores [1];
            flag = true;
            }

            s.insereEvento(new Saida(s.getInstante() + valores[0], s, this, next, c));



            }
            }
}

            // M�todo que devolve o n�mero de clientes atendidos no servi�o at� ao momento
            public int getAtendidos() {
            return atendidos;
            }
            public void act_stats() { //actualiza estatisticas
            //System.out.println("TEMPO:"+s.getInstante());
            double temp_desde_ult = s.getInstante() - temp_ult;
            ////////////////////////////////////////////////////////////////TAMANHO DA FILA ESTA SEMPRE A ZERO - corrigir
/ines
            temp_ult = s.getInstante();
            //System.out.println("TEMPO2"+soma_temp_esp);
            soma_temp_esp += fila.size() * temp_desde_ult;

            soma_temp_serv += estado * temp_desde_ult;
            }

            public String relat() { //providencia o relatório deste serviço
            //System.out.println("Pessoas atendidas "+atendidos);
            String saida="";
            double temp_med_fila = soma_temp_esp / (atendidos + fila.size()); //tempo medio de espera na fila

            double comp_med_fila = soma_temp_esp / s.getInstante(); //comprimento medio da fila de espera

            double utilizacao_serv = soma_temp_serv / s.getInstante();//tempo medio de atendimento no serviço
            //saida+="tempo espera total"+((double)((int)(soma_temp_esp*1000))/1000)+"\ntempo serviço
total"+((double)((int)(soma_temp_serv*1000))/1000);

            saida+="\n\nTempo médio de espera  " + ((double)((int)(temp_med_fila*1000))/1000)+"\n
Comprimento médio da fila   " + ((double)((int)(comp_med_fila*1000))/1000)+"\n Utilização do serviço   " +
(((double)((int)(utilizacao_serv*1000))/1000)/atendedores)+"\n Número de clientes atendidos   " +
atendidos+"\n Número de clientes na fila   " + fila.size()+"\n"; // Valor actual
            return saida;

            }

}
```

Classe Simulador

```
package scc_project;

import java.util.Random;


public class Simulador {

	private double instante;
	private double media_cheg, media_serv1, media_serv2, media_serv3;
	//private int n_clientes;
	//private int tempo_simulacao;
	private Servico servico1;
	private Servico servico2;
	private Servico servico3;

	// Lista de eventos - onde ficam registados todos os eventos que v�o ocorrer na simula��o
	// Cada simulador s� tem uma
	private ListaEventos lista;
	private int simulationType;


	// Construtor
	public Simulador() {
	media_cheg = 1.2;
	//n_clientes = 100;
	//tempo_simulacao = 600;

	}


  void insereEvento (Evento e1){
	 lista.insereEvento (e1);
  }

  private void act_stats(){
	if(simulationType==1){
	servico1.act_stats();
	servico2.act_stats();
	servico3.act_stats();
	}
	else{
	servico1.act_stats();
	}


  }
```

```
    private String relat (){
            String saida="";

            saida+="Estatisticas:\n";
            saida+="Tempo de simulação  " + ((double)((int)(instante*1000))/1000);

            if(simulationType==1){
            saida+="\nCenário 1: original \n";
            saida+="1:";
            saida+=servico1.relat();
            saida+="2:";
            saida+=servico3.relat();
            saida+="loja:";
            saida+=servico2.relat();

            }
            else if(simulationType == 2){

            saida+="\nCenario 2: com self service\n";
            saida+="1:";
            saida+=servico1.relat();



            }
            return saida;

    }

            public String executa(int maximo, int tipo, int simulationtype, int machineService1, int
machineService2, int machineService3) { //caso o tipo seja 1, executa por numero de clientes, caso seja 2
executa por tempo de simulação
            this.simulationType=simulationtype;

            instante = 0;

            Evento e1;
            String saida="";

            if(simulationtype==1){
            servico3 = new Servico(this, 3, 4, 2.5, servico2, machineService3, simulationtype);
            servico2 = new Servico(this, 2, 1, 0.5, null, machineService2, simulationtype);
            servico1 = new Servico(this, 1, 4, 2.5, servico2, machineService1, simulationtype);

            lista = new ListaEventos(this);
            Cliente c =new Cliente();

            ////////////////////////////////////////////////////////////////////////////
            if(c.getTipo()==1){
                    insereEvento(new Chegada(instante, this, null, servico1,c ));
```

```
}
else{
	insereEvento(new Chegada(instante, this, null, servico3,c ));
}
////////////////////////////////////////////////////////////////////////////////
//////////CORRE EM RELAÇÃO AO N CLIENTE
if(tipo==1){
	while (servico2.getAtendidos() < maximo) {

	e1 = (Evento) (lista.removeFirst());
	instante = e1.getInstante();
	act_stats();

	e1.executa();

};
saida=relat();

}
//////////CORRE EM RELAÇÃO AO TEMPO
if(tipo==2){

	while (instante < maximo) {

	e1 = (Evento) (lista.removeFirst());
	instante = e1.getInstante();
	act_stats();

	e1.executa();
	};
	instante=maximo;
	saida=relat();
}
}
else if(simulationtype==2){

//um serviço com 4 maquinas self service
servico1 = new Servico(this, 1, 4.5, 2, null, machineService1, simulationtype);

lista = new ListaEventos(this);
Cliente c = new Cliente();

////////////////////////////////////////////////////////////////////////////

insereEvento(new Chegada(instante, this, null, servico1, c ));

////////////////////////////////////////////////////////////////////////////////
//////////CORRE EM RELAÇÃO AO N CLIENTE
if(tipo==1){
	while ((servico1.getAtendidos()) < maximo) {
```

```
            e1 = (Evento) (lista.removeFirst());
            instante = e1.getInstante();
            act_stats();

            e1.executa();

};
saida=relat();



}
//////////CORRE EM RELAÇÃO AO TEMPO
if(tipo==2){

            while (instante < maximo) {

            e1 = (Evento) (lista.removeFirst());
            instante = e1.getInstante();
            act_stats();

            e1.executa();
            };
            instante=maximo;
            saida=relat();
}
}
return saida;

}


public double getInstante() {
return instante;
}


public double getMedia_cheg() {
return media_cheg;
}

// M�todo que devolve a m�dia dos tempos de servi�o
public double getMedia_serv1() {
return media_serv1;
}

public double getMedia_serv2() {
return media_serv2;
}

public double getMedia_serv3(){
```

```
        return media_serv3;
        }

        public Servico getServ2 (){
        return servico2;
        }

        public Servico getServ1 (){
        return servico1;
        }

        public Servico getServ3(){
        return servico3;
        }



}
```

Classe Transicao

```
package scc_project;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author utilizador
 */
public class Transicao extends Evento{
        Cliente cliente;
        Transicao (double i, Simulador s, Servico serv,Servico fina, Cliente c){
         super(i, s, serv,fina,c);
        this.cliente =c;

   }
        void executa (){
        serv.removeServico();
        s.getServ2().insereServico(c);

        }
        public String toString(){
        return "Transição em " + instante;
        }


}
```