
	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

		<b>FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES</b>	
<b>CARRERA:</b> COMPUTACIÓN/INGENIERÍA DE SISTEMAS		<b>ASIGNATURA:</b> PROGRAMACIÓN APLICADA	
<b>NRO. PROYECTO:</b>	1.1	<b>TÍTULO PROYECTO:</b> Prueba Practica 2 Desarrollo e implementación de un sistema de simulación de acceso y atención bancaria	
<b>OBJETIVO:</b> Reforzar los conocimientos adquiridos en clase sobre la programación en Hilos en un contexto real.			
<b>INSTRUCCIONES:</b>		1. Revisar el contenido teórico y practico del tema	
		2. Profundizar los conocimientos revisando los libros guías, los enlaces contenidos en los objetos de aprendizaje Java y la documentación disponible en fuentes académicas en línea.	
		3. Deberá desarrollar un sistema informático para la simulación y una interfaz grafica.	
		4. Deberá generar un informe de la practica en formato PDF y en conjunto con el código se debe subir al GitHub personal y AVAC.	
		5. <b>Fecha de entrega:</b> El sistema debe ser subido al git hasta <b>17 de enero del 2021 – 23:55.</b>	
<b>ACTIVIDADES POR DESARROLLAR</b>			

## 1. Enunciado:

Realizar un sistema de simulación de acceso y atención a través de colas de un banco.

**Problema:** Un banco necesita controlar el acceso a cuentas bancarias y para ello desea hacer un programa de prueba en Java que permita lanzar procesos que ingresen y retiren dinero a la vez y comprobar así si el resultado final es el esperado.

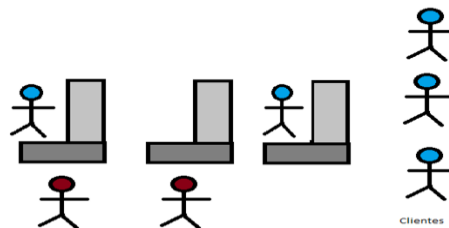
Se parte de una cuenta con 100 euros y se pueden tener procesos que ingresen 100 euros, 50 o 20. También se pueden tener procesos que retiren 100, 50 o 20 euros. Se desean tener los siguientes procesos:

- 40 procesos que ingresan 100
- 20 procesos que ingresan 50
- 60 que ingresen 20.

De la misma manera se desean lo siguientes procesos que retiran cantidades.

- 40 procesos que retiran 100
- 20 procesos que retiran 50
- 60 que retiran 20.


Ademas en el banco, existen 3 cajeros que pueden atender y hay un cola inicial de 10 clientes para ser atendidos, el proceso de atención es de 20 – 15 segundos y los clientes llegan constantemente cada 30 - 50 segundos. Ningún cajero puede atender simultáneamente, adicionalmente el tiempo de moverme de la cola al estante del cajero es de 2 - 5 segundos, esto deberán ser generados aleatoriamente entre los 100 clientes que disponen una cuenta, estos pueden volver a ingresar el numero de veces que sea necesario.



**Se desea comprobar que tras la ejecución la cuenta tiene exactamente 100 euros, que era la cantidad de la que se disponía al principio. Realizar el programa Java que demuestra dicho hecho.**

### Calificación:

- Diagrama de Clase 10%
- MVC: 10%
- Técnicas de Programación aplicadas (Java 8, Reflexión y Programación Genérica): 10%

	<b>Computación</b>	<b>Docente: Diego Quisi Peralta</b>
	Programación Aplicada	<b>Período Lectivo:</b> Septiembre 2020 – Febrero 2021

- Hilos 30%
- Sincronización 10%
- Interfaz Gráfica de simulación 20%
- Informe: 10%

## 2. Informe de Actividades:

- Planteamiento y descripción del problema.
- Diagramas de Clases.
- Patrón de diseño aplicado
- Descripción de la solución y pasos seguidos.
  - Comprobación de las cuentas bancarias e interfaz gráfica.
- Conclusiones y recomendaciones.
- Resultados.

## RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

## CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones gráficas en sistemas.
- Los estudiantes están en la capacidad de implementar hilos.

## RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la prueba.**

## BIBLIOGRAFIA:

[1]: <https://www.ups.edu.ec/evento?calendarBookingId=98892>

**Docente / Técnico Docente:** Ing. Diego Quisi Peralta Msc.

**Firma:** \_\_\_\_\_

**CARRERA:**

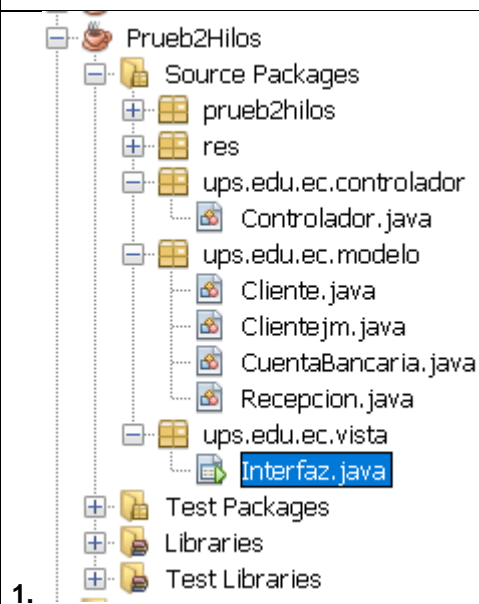
**ASIGNATURA:**

**NRO. PRÁCTICA:**

**TÍTULO PRÁCTICA:**

**OBJETIVO ALCANZADO:**

**ACTIVIDADES DESARROLLADAS**



```
14 public class Recepcion implements Runnable{
15     private Cliente cliente;
16     private CuentaBancaria cuentaBancaria;
17     private long initialTime;
18
19     public Recepcion(Cliente cliente, long initialTime, CuentaBancaria cuentaBancaria) {
20         this.cliente = cliente;
21         this.initialTime = initialTime;
22         this.cuentaBancaria = cuentaBancaria;
23     }
24
25     @Override
26     public void run() {
27         try {
28             tiempoFilacaja();
29         } catch (InterruptedException e) {
30             // TODO Auto-generated catch block
31             e.printStackTrace();
32         }
33
34         System.out.println("\nLa cajera " + Thread.currentThread().getName()
35             + "\n" + "COMIENZA recibe el dinero del " + this.cliente.getNombre() + " con valor de: " + cliente.cantidad);
36
37         for (int i = 0; i < this.cliente.getOperacional().length; i++) {
38             // Se procesa el pedido en X segundos
39             this.esperarXsegundos(cliente.getOperacional()[i]);
40             System.out.println("Procesando la accion" + (i + 1) + " del " + this.cliente.getNombre()
41                 + " en un tiempo de: " + cliente.getOperacional()[i] + " seg");
42         }
43
44         System.out.println("\nLa cajera " + Thread.currentThread().getName() + "\n" + "HA TERMINADO DE PROCESAR "
45             + this.cliente.getNombre());
46         // EN EL TIEMPO: "
47         // (System.currentTimeMillis() - this.initialTime) / 1000 + "seg" valor salido: "+cuentaBancaria.getSaldo() );
48
49         private void esperarXsegundos(int segundos) {
50             try {
51                 Thread.sleep(segundos * 1000);
52             } catch (InterruptedException ex) {
53                 Thread.currentThread().interrupt();
54             }
55         }
56
57         public void tiempoFilacaja() throws InterruptedException {
58             Random r = new Random();
59             int Low = 2;
60             int High = 5;
61             int Result = r.nextInt(High-Low) + Low;
62             System.out.println("Sale de la cola: " + this.cliente.getNombre());
63             Thread.sleep(Result*1000);
64             System.out.println("Llega al mostrador " + this.cliente.getNombre() + " Luego de " + Result + " segundos");
65
66             public long getInitialTime() {
67                 return initialTime;
68             }
69         }
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
8
```

```

public class Controlador {

    public void tColaAEstante() throws InterruptedException{
        Random r = new Random();
        int Low = 200;
        int High = 500;
        int Result = r.nextInt(High-Low) + Low;

        Thread.sleep(Result);

    }

    public void tatencion() throws InterruptedException{
        Random r = new Random();
        // int Low = 1500;
        //int High = 20000;
        int Low = 700;
        int High = 1000;
        int Result = r.nextInt(High-Low) + Low;
        System.out.println("Empieza la transaccion");
        Thread.sleep(Result);

    }

}

```

3.

```
public void tiempoFilacaja() throws InterruptedException {
    Random r = new Random();
    int Low = 2;
    int High = 5;
    int Result = r.nextInt(High-Low) + Low;
    System.out.println("Sale de la cola: "+this.cliente.getNombre());
    Thread.sleep(Result*100);
    System.out.println("Llega al mostrador "+this.cliente.getNombre()+" Luego de "+Result+" segundos");}

    public long getInitialTime() {
        return initialTime;
    }

    public void setInitialTime(long initialTime) {
        this.initialTime = initialTime;
    }

    public Cliente getCliente() {
        return cliente;
    }

    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
    }
}
```

```
import java.util.ArrayList;
import java.util.Random;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.logging.Level;
import java.util.logging.Logger;
import ups.edu.ec.controlador.Controlador;
import ups.edu.ec.modelo.Cliente;
import ups.edu.ec.modelo.CuentaBancaria;
import ups.edu.ec.modelo.Recepcion;

/**
 *
 * @author japch
 */
public class Interfaz extends javax.swing.JFrame {
```

```
private static final int numCajeras = 3;

/**
 * Creates new form Interfaz
 */
public Interfaz() throws InterruptedException {
    initComponents();

    CuentaBancaria cuenta = new CuentaBancaria(100);
    Controlador cn= new Controlador();
    Random r = new Random();

    final int Nopciones_100 = 41; // son 40
    final int Nopciones_50 = 21;//20
    final int Nopciones_20 = 61;//60

    int v1=100;
    int v11=-100;
    int v2=50;
    int v22=-50;
    int v3=20;
    int v33=-20;
    ArrayList<Cliente> clientes = new ArrayList<Cliente>();

    Thread[] hilosIngresan100 = new Thread[Nopciones_100];
    Thread[] hilosRetiran100 = new Thread[Nopciones_100];
    Thread[] hilosIngresan50 = new Thread[Nopciones_50];
    Thread[] hilosRetiran50 = new Thread[Nopciones_50];
    Thread[] hilosIngresan20 = new Thread[Nopciones_20];
    Thread[] hilosRetiran20 = new Thread[Nopciones_20];
```



/\* Arrancamos todos los hilos\*/

```
for (int i=1; i<Nopciones_100;i++){  
    int Low = 5;  
    int High = 10;  
    int Result = r.nextInt(High-Low) + Low;  
    Cliente ingresa = new Cliente(cuenta, v1, "Cliente "+i, new int[] { Result });  
    Cliente retira = new Cliente(cuenta, v11, "Cliente "+i, new int[] { Result });  
    clientes.add(new Cliente(cuenta, v1, "Cliente "+i, new int[] { Result }));  
  
    hilosIngresa100[i]= new Thread(ingresa);  
    hilosRetiran100[i] = new Thread(retira);  
    // System.out.println("Cuenta "+decliente: "+i+" "+cuenta.getSaldo());  
    hilosIngresa100[i].start();  
    hilosRetiran100[i].start();  
}
```

/\*

//////////\*/

System.out.println("----->Los que depositaran 50<-----");

/\* //////////

\*/

```
for (int i=1; i<Nopciones_50;i++){  
    int Low = 5;  
    int High = 10;  
    int Result = r.nextInt(High-Low) + Low;  
    Cliente ingresa = new Cliente(cuenta, v2, "Cliente "+i, new int[] { Result });
```

```

        Cliente retira = new Cliente(cuenta, v22, "Cliente "+i, new int[] { Result });
        clientes.add(new Cliente(cuenta, v2, "Cliente "+i, new int[] { Result }));

        hilosIngresa50[i]= new Thread(ingresa);
        hilosRetiran50[i] = new Thread(retira);

        hilosIngresa50[i].start();
        hilosRetiran50[i].start();

    }

    /*
    ////////////*/
    System.out.println("----->Los que depositaran 20<-----");
    /* ////////////
    */

    for (int i=1; i<Nopciones_20;i++){
        int Low = 5;
        int High = 10;
        int Result = r.nextInt(High-Low) + Low;
        Cliente ingresa = new Cliente(cuenta, v3, "Cliente "+i, new int[] { Result });
        Cliente retira = new Cliente(cuenta, v33, "Cliente "+i, new int[] { Result });
        clientes.add(new Cliente(cuenta, v3, "Cliente "+i, new int[] { Result }));

        hilosIngresa20[i]= new Thread(ingresa);
        hilosRetiran20[i] = new Thread(retira);

        hilosIngresa20[i].start();
        hilosRetiran20[i].start();
    }

```

```
long init = System.currentTimeMillis(); // Instante inicial del procesamiento
ExecutorService executor = Executors.newFixedThreadPool(numCajeras);
for (Cliente cliente: clientes) {
    Runnable cajera = new Recepcion(cliente, init, cuenta);
    executor.execute(cajera);
}
executor.shutdown(); // Cierro el Executor
while (!executor.isTerminated()) {
    // Los procesos deben terminar de ejecutarse
}
long fin = System.currentTimeMillis(); // Instante final del procesamiento
System.out.println("Tiempo total de procesamiento: " + (fin - init) / 1000 + " Segundos");
/* En este punto todos los hilos están arrancados,
ahora toca esperarlos */

for (int i=1; i<Nopciones_100;i++){
    // System.out.println("valor "+i);
    hilosIngresa100[i].join();
    hilosRetiran100[i].join();
}

for (int i=1; i<Nopciones_50;i++){
    hilosIngresa50[i].join();
    hilosRetiran50[i].join();
}

for (int i=1; i<Nopciones_20;i++){
    hilosIngresa20[i].join();
    hilosRetiran20[i].join();
}
```

```
}

if (cuenta.esSimulacionCorrecta()){
    System.out.println("La simulación fue correcta");
} else {
    System.out.println("La simulación falló ");
    System.out.println("La cuenta tiene:" +
        cuenta.getSaldo());
    System.out.println("Revise sus synchronized");
}
}
```

4.

Design Preview [Interfaz]



Cajera1



Cajera2



Cajera3

jBu...

jBu...

jBu...

jBu...

jBu...

jBu...

jBu...

jBu...

jBu...

jBu...

5.

6. Prueb2Hilos - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+H)

Project: Prueb2Hilos

Files: Cliente.java, CuentaBancaria.java, Interfaz.java, Recepcion.java, Controlador.java

Source Packages: productosconsumido, Consumidor.java, Productor.java, ProductorConsum

Test Packages: res, ups.edu.ec.controlac, Controlador.java, ups.edu.ec.modelo, Cliente.java, ClienteJm.java, CuentaBancaria.java, Recepcion.java, ups.edu.ec.vista, Interfaz.java, Lanzador.java

Libraries: UNLDiagramaGestorMatr, UNLDiagramas, UNLDiagramaGestorEscal, UNLDiagramaMovEc

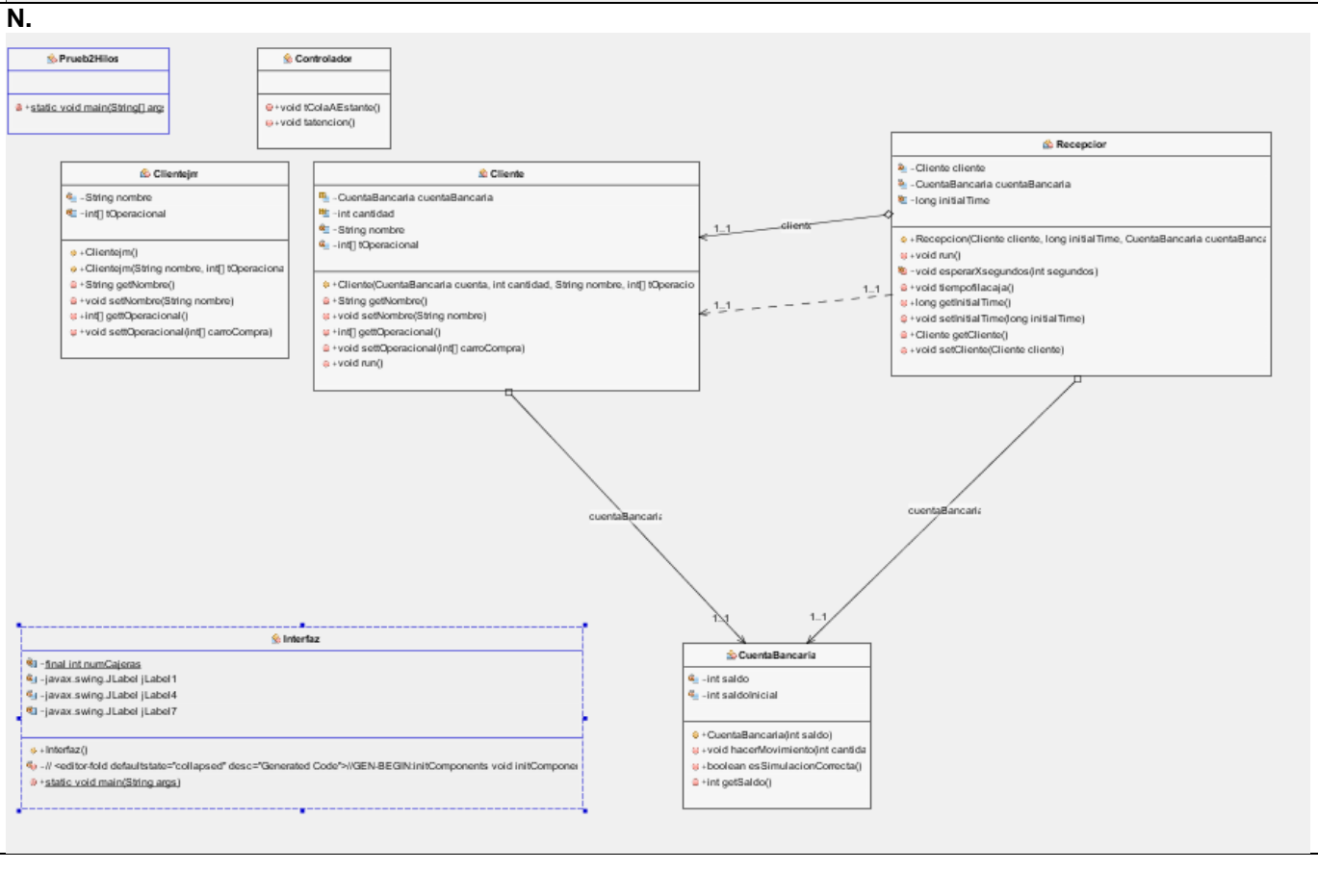
Members: Interface: Frame, Interface(), initComponents(), main(String[] args), JLabel1: JLabel, JLabel2: JLabel, JLabel3: JLabel, JLabel5: JLabel, JLabel6: JLabel, numCajas: int

Output: Prueb2Hilos (run)

```

run:
Sale de la cola: Cliente 1
Sale de la cola: Cliente 3
Sale de la cola: Cliente 2
Llega al mostrador Cliente 2 Luego de 2 segundos
Llega al mostrador Cliente 1 Luego de 2 segundos
"La cajera pool-1-thread-1" COMIENZA recibe el dinero del Cliente 1 de: 100
Llega al mostrador Cliente 3 Luego de 2 segundos
"La cajera pool-1-thread-3" COMIENZA recibe el dinero del Cliente 3 de: 100
"La cajera pool-1-thread-2" COMIENZA recibe el dinero del Cliente 2 de: 100
Procesando la accion del Cliente 2 en un tiempo de: 7seg
"La cajera pool-1-thread-2" HA TERMINADO DE PROCESAR Cliente 2
Sale de la cola: Cliente 4
Llega al mostrador Cliente 4 Luego de 3 segundos
"La cajera pool-1-thread-2" COMIENZA recibe el dinero del Cliente 4 de: 100
Procesando la accion del Cliente 1 en un tiempo de: 9seg
"La cajera pool-1-thread-1" HA TERMINADO DE PROCESAR Cliente 1
Procesando la accion del Cliente 3 en un tiempo de: 9seg
"La cajera pool-1-thread-3" HA TERMINADO DE PROCESAR Cliente 3
Sale de la cola: Cliente 5
Llega al mostrador Cliente 5 Luego de 3 segundos
"La cajera pool-1-thread-1" COMIENZA recibe el dinero del Cliente 5 de: 100
Llega al mostrador Cliente 6 Luego de 3 segundos
"La cajera pool-1-thread-3" COMIENZA recibe el dinero del Cliente 6 de: 100
Procesando la accion del Cliente 5 en un tiempo de: 5seg
"La cajera pool-1-thread-3" HA TERMINADO DE PROCESAR Cliente 6
Sale de la cola: Cliente 7
Llega al mostrador Cliente 7 Luego de 2 segundos
"La cajera pool-1-thread-3" COMIENZA recibe el dinero del Cliente 7 de: 100
Procesando la accion del Cliente 4 en un tiempo de: 8seg
"La cajera pool-1-thread-2" HA TERMINADO DE PROCESAR Cliente 4
Sale de la cola: Cliente 8
Llega al mostrador Cliente 8 Luego de 2 segundos
"La cajera pool-1-thread-2" COMIENZA recibe el dinero del Cliente 8 de: 100
Procesando la accion del Cliente 5 en un tiempo de: 9seg
"La cajera pool-1-thread-1" HA TERMINADO DE PROCESAR Cliente 5
Sale de la cola: Cliente 9
Llega al mostrador Cliente 9 Luego de 2 segundos
"La cajera pool-1-thread-1" COMIENZA recibe el dinero del Cliente 9 de: 100
Procesando la accion del Cliente 7 en un tiempo de: 8seg
"La cajera pool-1-thread-3" HA TERMINADO DE PROCESAR Cliente 7
Sale de la cola: Cliente 10
Llega al mostrador Cliente 10 Luego de 2 segundos
"La cajera pool-1-thread-3" COMIENZA recibe el dinero del Cliente 10 de: 100
Procesando la accion del Cliente 8 en un tiempo de: 8seg
  
```

65/87 PMS



**RESULTADO(S) OBTENIDO(S):**

No Satisfactorios por mi propia parte, siento yo que pude terminarlo con todo lo establecido en el enunciado

**CONCLUSIONES:** resuelto los puntos importantes a excepción de la interfaz visual del como un cliente se acerca a ventanilla en ese instante pasaría a un estado de ocupado, cuando apse el tiempo(Random) saldrá y llegara el siguiente a ocupar

**RECOMENDACIONES:**

Mas practicas directas en clases practicas

**Nombre de estudiante:** \_\_\_\_\_ Christian Japon \_\_\_\_\_

**Firma de estudiante** \_\_\_\_\_

