



CAJERAS NORMAL, THREAD Y RUNNABLE

MARIO JIMÉNEZ MARSET

ÍNDICE

1. ENUNCIADO - OBJETIVOS.....	3
2. DESARROLLO	3

1. ENUNCIADO - OBJETIVOS

En esta práctica se pedía realizar, por una parte, tres tipos de programas relacionados con cajeras, en los que se tenía que simular cómo una cajera pasaba los productos (el tiempo que tardaba en hacerlo, el número de productos y clientes...).

Todo esto con la filosofía de los hilos:

- De forma normal (donde se crea un proceso con un solo hilo de ejecución).
- Heredando de Thread (se ejecutan varios hilos a la vez).
- Implementando la interfaz Runnable (también se ejecutan varios hilos a la vez).

Finalmente, se pedía crear otros dos programas con más clientes y productos con Thread y Runnable, con el objetivo de comprobar su funcionamiento con más objetos.

2. DESARROLLO

En primer lugar, se muestra el código y comentarios de las clases 'Cajera' y 'Cliente', donde se establecen sus características y métodos.

Explicación Clase '**Cajera**':

```
package cajeras;
```

```
public class Cajera {

    private String nombre;

    public Cajera(String nombre) {
        this.nombre=nombre;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre=nombre;
    }
    public void procesarCompra(Cliente cliente, long timeStamp) {
        System.out.println("La cajera " + this.nombre +
            "\nCOMIENZA A PROCESAR LA COMPRA DEL
CLIENTE " + cliente.getNombre() +
            " EN EL TIEMPO: " + (System.currentTimeMillis() -
timeStamp) / 1000      +
            "seg");
        for (int i = 0; i < cliente.getcarroCompra().length; i++) {
            this.esperarXsegundos(cliente.getcarroCompra()[i]);
            System.out.println("Procesado el producto " + (i + 1) +
            "->Tiempo: " + (System.currentTimeMillis() -
timeStamp) / 1000 +
            "seg");
        }
        System.out.println("La cajera " + this.nombre + " HA TERMINADO DE
PROCESAR " +
            cliente.getNombre() + " EN EL TIEMPO: " +
            (System.currentTimeMillis() - timeStamp) / 1000 +
            "seg");
    }

}
```

```
        private void esperarXsegundos(int segundos) {
            try {
                Thread.sleep(segundos * 1000);
            } catch (InterruptedException ex) {
                Thread.currentThread().interrupt();
            }
        }
    }
}
```

Se crea una clase Cajera donde se pasa por el constructor el nombre de la cajera (construyendo su getter y setter).

Posteriormente, se crea un método void que procesa la compra del cliente. Se pasa por parámetro un objeto de la clase Cliente y un tiempo de tipo long.

Dentro se imprime un mensaje donde se muestra el nombre del cliente y el tiempo exacto de cuándo se va a atender a ese cliente (al primer cliente será en el segundo 0 obviamente). Esto se consigue con el método currentTimeMillis y operaciones con el tiempo de tipo long pasado por parámetro.

Se crea un bucle for, donde el índice sea menor que el volumen del carro de compra del cliente: dentro se procesa el producto que esté en la posición específica del for y se calcula el tiempo de procesamiento igual que en el apartado anterior.

Finalmente, en otro método void se calculan esos segundos de espera que se muestran en el for.

Explicación Clase '**Cliente**':

```
package cajeras;
```

```
public class Cliente {

    private String nombre;
    private int[] carroCompra;

    public Cliente(String nombre, int[] carroCompra) {
        this.nombre=nombre;
        this.carroCompra=carroCompra;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre=nombre;
    }

    public int[] getcarroCompra() {
        return carroCompra;
    }

    public void setcarroCompra(int[] carroCompra) {
        this.carroCompra=carroCompra;
    }
}
```

```
    }  
}
```

En esta clase se crean las variables de cliente nombre y carroCompra (array), las cuales se introducen en el constructor y de las que se crean sus getters y setters.

Explicadas estas clases clave para la realización de los procesos, se muestran los códigos y explicaciones de los tres tipos de cajeras, además de los resultados.

Explicación Clase '**CajeraNormal**':

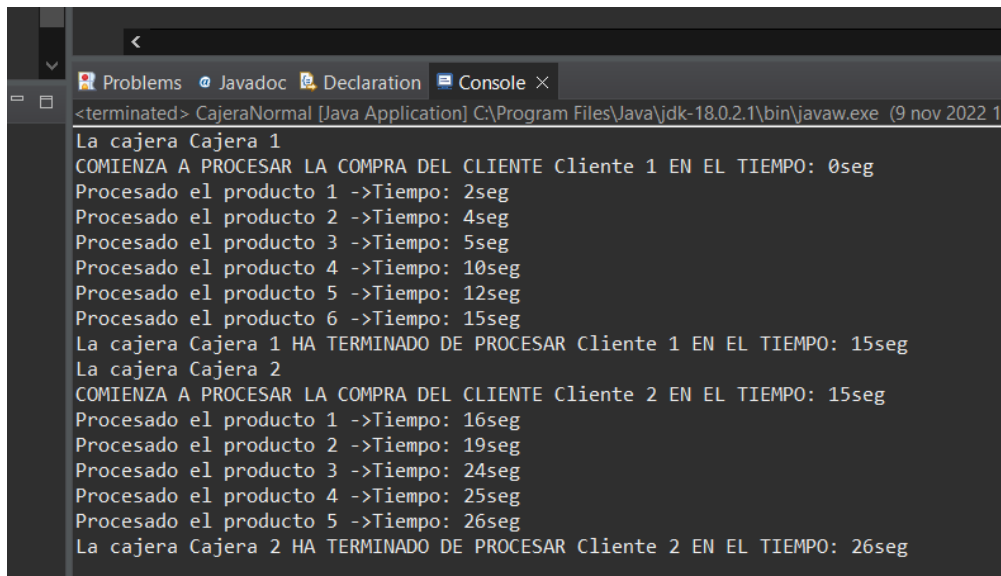
```
package cajeras;  
  
public class CajeraNormal {  
    public static void main(String[] args) {  
        //Proceso con un solo hilo de ejecucion  
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });  
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });  
  
        Cajera cajera1 = new Cajera("Cajera 1");  
        Cajera cajera2 = new Cajera("Cajera 2");  
  
        // Tiempo inicial de referencia  
        long initialTime = System.currentTimeMillis();  
        cajera1.procesarCompra(cliente1, initialTime);  
        // initialTime = System.currentTimeMillis();  
        cajera2.procesarCompra(cliente2, initialTime);  
    }  
}
```

En esta clase, dentro del main, se crean dos objetos cliente, donde se establecen el nombre y el array del carrito de la compra: dentro del mismo se establecen números cualquiera; esos números establecen el tiempo que se tarda en procesarlos y pasar al siguiente producto.

Se crean dos objetos cajera, estableciendo en cada uno el nombre de la cajera.

Se crea un tiempo de tipo long, el cual se establece en milisegundos con el método currentTimeMillis. Se llama a los objetos cajera con el método que procesa la compra, pasándoles por argumentos un cliente a cada una y el tiempo de tipo long.

Resultado:



```
<terminated> CajeraNormal [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (9 nov 2022 1
La cajera Cajera 1
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg
La cajera Cajera 2
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIEMPO: 15seg
Procesado el producto 1 ->Tiempo: 16seg
Procesado el producto 2 ->Tiempo: 19seg
Procesado el producto 3 ->Tiempo: 24seg
Procesado el producto 4 ->Tiempo: 25seg
Procesado el producto 5 ->Tiempo: 26seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 26seg
```

Se muestra cómo se ejecuta un proceso con un solo hilo en ejecución, y cuando este ha terminado, se ejecuta el otro proceso con su hilo de ejecución.

Explicación Clase '**CajeraThread**':

```
package cajeras;
```

```
class CajeraThread extends Thread {
    private String nombre;
    private Cliente cliente;
    private long initialTime;

    // Constructor, getter & setter

    public CajeraThread(String nombre, Cliente cliente, long initialTime) {
        this.nombre=nombre;
        this.cliente=cliente;
        this.initialTime=initialTime;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre=nombre;
    }

    public Cliente getCliente() {
        return cliente;
    }

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

```
        public long getInitialTime() {
            return initialTime;
        }

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

        @Override
        public void run() {
            Cajera cajera=new Cajera(this.nombre);
            cajera.procesarCompra(this.cliente, this.initialTime);
        }
    }

    public class MainThread {
        public static void main(String[] args) {
            //Se ejecutan varios hilos a la vez (extendiendo Thread)
            Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });
            Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });

            // Tiempo inicial de referencia
            long initialTime = System.currentTimeMillis();
            Thread cajera1 = new CajeraThread("Cajera 1", cliente1, initialTime);
            Thread cajera2 = new CajeraThread("Cajera 2", cliente2, initialTime);

            cajera1.start();
            cajera2.start();
        }
    }
```

Se crea la primera clase “CajeraThread”, la cual extiende de la clase Thread, implementando su método run y sobrescribiéndolo.

Primero, se crean el nombre de la cajera, el tiempo de tipo long y un objeto de la clase Cliente. Se pasan al constructor, además de crearse sus respectivos getters y setters.

En el método run, entonces, se sobrescribe creando un objeto cajera, al cual se pasa el nombre de cajera de esta clase. Se llama al método que procesa la compra, pasándole por parámetros el objeto cliente y el tiempo de tipo long.

Se crea la clase “MainThread”, en cuyo main se crean dos objetos clientes, escribiendo sus respectivos nombres y contenido de sus arrays de carrito de compra.

Se crea el tiempo de tipo long, además de los objetos cajera, escribiendo su nombre, el cliente en específico y el tiempo de tipo long. Finalmente, se llama al método start para empezar los procesos con los objetos cajera.

Resultado:

```

52 Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });
<terminated> MainThread [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (9 nov 2022 12:52:56 - 12:53:11) [pid: 7
La cajera Cajera 1
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg
La cajera Cajera 2
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIEMPO: 0seg
Procesado el producto 1 ->Tiempo: 1seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 3 ->Tiempo: 9seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 5 ->Tiempo: 11seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 11seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg

```

Este caso es diferente a la anterior ejecución, ya que se ejecutan varios hilos a la vez, lo cual hace que un producto de un cliente se procese y, después, el de la misma posición del otro cliente: así, hasta terminar un cliente; cuando este lo hace, se termina el proceso del cliente que queda.

Cada Thread crea un objeto único y, este, se asocia con ese mismo objeto.

Explicación Clase '**Cajera Runnable**':

package cajeras;

```

public class CajeraRunnable implements Runnable{
    private Cliente cliente;
    private Cajera cajera;
    private long initialTime;

    public CajeraRunnable (Cliente cliente, Cajera cajera, long initialTime){
        this.cajera = cajera;
        this.cliente = cliente;
        this.initialTime = initialTime;
    }

    public static void main(String[] args) {
        //Se ejecutan varios hilos a la vez (implementando Runnable)
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });

        Cajera cajera1 = new Cajera("Cajera 1");
        Cajera cajera2 = new Cajera("Cajera 2");

        // Tiempo inicial de referencia
        long initialTime = System.currentTimeMillis();

        Runnable proceso1 = new CajeraRunnable(cliente1, cajera1,
initialTime);

```



```

        Runnable proceso2 = new CajeraRunnable(cliente2, cajera2,
initialTime);

        new Thread(proceso1).start();
        new Thread(proceso2).start();

    }

    @Override
    public void run() {
        this.cajera.procesarCompra(this.cliente, this.initialTime);
    }
}

```

En esta clase se implementa la interfaz Runnable, lo cual permite que cada subprocesso comparta la misma instancia ejecutable. Además, aún se guarda la opción de extender a cualquier otra clase.

Se realiza el mismo procedimiento que en la cajera Thread, cambiando que el método run implementado solamente llama al proceso de compra con el objeto cajera creado y pasando por parámetro el objeto cliente y el tiempo de tipo long.

También, en el main cambia que se crean los objetos de tipo Runnable, pasando por parámetros el objeto cliente y cajera, además del tiempo de tipo long.

Resultado:

```

<terminated> CajeraRunnable [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (9 nov 2022 14:03:37 -
La cajera Cajera 1
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg
La cajera Cajera 2
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIEMPO: 0seg
Procesado el producto 1 ->Tiempo: 1seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 3 ->Tiempo: 9seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 5 ->Tiempo: 11seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 11seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg

```

El resultado es igual al de la clase Thread. Sin embargo, entre Thread y Runnable hay diferencias. Al extender la clase Thread, se crea un objeto único para él y se asocia con ese objeto; sin embargo, cada subprocesso creado al implementar una interfaz ejecutable comparte la misma instancia ejecutable.

Al pasar esto, con Thread se requiere más memoria y, con Runnable, menos.

Explicación Clase **'MainThreadDos'**:

```
package cajerasdeluxe;
import cajeras.Cajera;
import cajeras.Cliente;

class CajeraThread extends Thread {
    private String nombre;
    private Cliente cliente;
    private long initialTime;

    // Constructor, getter & setter

    public CajeraThread(String nombre, Cliente cliente, long initialTime) {
        this.nombre=nombre;
        this.cliente=cliente;
        this.initialTime=initialTime;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre=nombre;
    }

    public Cliente getCliente() {
        return cliente;
    }

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

    public long getInitialTime() {
        return initialTime;
    }

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

    @Override
    public void run() {
        Cajera cajera=new Cajera(this.nombre);
        cajera.procesarCompra(this.cliente, this.initialTime);
    }
}

public class MainThreadDos {
    public static void main(String[] args) {
        //Se ejecutan varios hilos a la vez (extendiendo Thread)
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });
        Cliente cliente3 = new Cliente("Cliente 3", new int[] { 4, 1, 3, 6, 2, 7, 8 });
        Cliente cliente4 = new Cliente("Cliente 4", new int[] { 1, 1, 2, 6 });
    }
}
```

```

        Cliente cliente5 = new Cliente("Cliente 5", new int[] { 7, 8, 1, 1, 1 });

        // Tiempo inicial de referencia
        long initialTime = System.currentTimeMillis();
        Thread cajera1 = new CajeraThread("Cajera 1", cliente1, initialTime);
        Thread cajera2 = new CajeraThread("Cajera 2", cliente2, initialTime);
        Thread cajera3 = new CajeraThread("Cajera 3", cliente3, initialTime);
        Thread cajera4 = new CajeraThread("Cajera 4", cliente4, initialTime);
        Thread cajera5 = new CajeraThread("Cajera 5", cliente5, initialTime);
        cajera1.start();
        cajera2.start();
        cajera3.start();
        cajera4.start();
        cajera5.start();
    }
}

```

Simplemente, esta clase realiza exactamente la misma función que la clase “MainThread”. El único cambio visible es que, en vez de ser solamente dos cajeras y dos clientes, son cinco cajeras y cinco clientes. El objetivo de volver a realizar el mismo programa es ver cómo con más objetos sigue funcionando el programa.

Resultado:

```

<terminated> MainThreadDos [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (9 nov 2022 14:13:54 - 14:14:25) [pid
18]
La cajera Cajera 2
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIEMPO: 0seg
La cajera Cajera 1
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg
La cajera Cajera 5
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 5 EN EL TIEMPO: 0seg
Procesado el producto 1 ->Tiempo: 1seg
Procesado el producto 1 ->Tiempo: 1seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 2 ->Tiempo: 2seg
Procesado el producto 1 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 3 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 5seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 1 ->Tiempo: 7seg
Procesado el producto 3 ->Tiempo: 8seg
Procesado el producto 3 ->Tiempo: 9seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 4 ->Tiempo: 10seg
La cajera Cajera 4 HA TERMINADO DE PROCESAR Cliente 4 EN EL TIEMPO: 10seg
Procesado el producto 5 ->Tiempo: 11seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 11seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 4 ->Tiempo: 14seg
Procesado el producto 2 ->Tiempo: 15seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg
Procesado el producto 3 ->Tiempo: 16seg
Procesado el producto 5 ->Tiempo: 16seg
Procesado el producto 4 ->Tiempo: 17seg
Procesado el producto 5 ->Tiempo: 18seg
La cajera Cajera 5 HA TERMINADO DE PROCESAR Cliente 5 EN EL TIEMPO: 18seg
Procesado el producto 6 ->Tiempo: 23seg
Procesado el producto 7 ->Tiempo: 31seg
La cajera Cajera 3 HA TERMINADO DE PROCESAR Cliente 3 EN EL TIEMPO: 31seg

```

El mismo resultado que en la otra clase se muestra: solamente cambia la cantidad de cajeras y clientes a procesar.

Explicación Clase **'MainRunnableDos'**:

```
package cajerasdeluxe;
import cajeras.Cajera;
import cajeras.Cliente;

public class MainRunnableDos implements Runnable{
    private Cliente cliente;
    private Cajera cajera;
    private long initialTime;

    public MainRunnableDos (Cliente cliente, Cajera cajera, long initialTime){
        this.cajera = cajera;
        this.cliente = cliente;
        this.initialTime = initialTime;
    }

    public static void main(String[] args) {
        //Se ejecutan varios hilos a la vez (implementando Runnable)
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });
        Cliente cliente3 = new Cliente("Cliente 3", new int[] { 3, 1, 2, 3, 7, 2 });
        Cliente cliente4 = new Cliente("Cliente 4", new int[] { 4, 5, 3, 6, 1 });
        Cliente cliente5 = new Cliente("Cliente 5", new int[] { 5, 2, 3, 1, 4, 2 });

        Cajera cajera1 = new Cajera("Cajera 1");
        Cajera cajera2 = new Cajera("Cajera 2");
        Cajera cajera3 = new Cajera("Cajera 3");
        Cajera cajera4 = new Cajera("Cajera 4");
        Cajera cajera5 = new Cajera("Cajera 5");

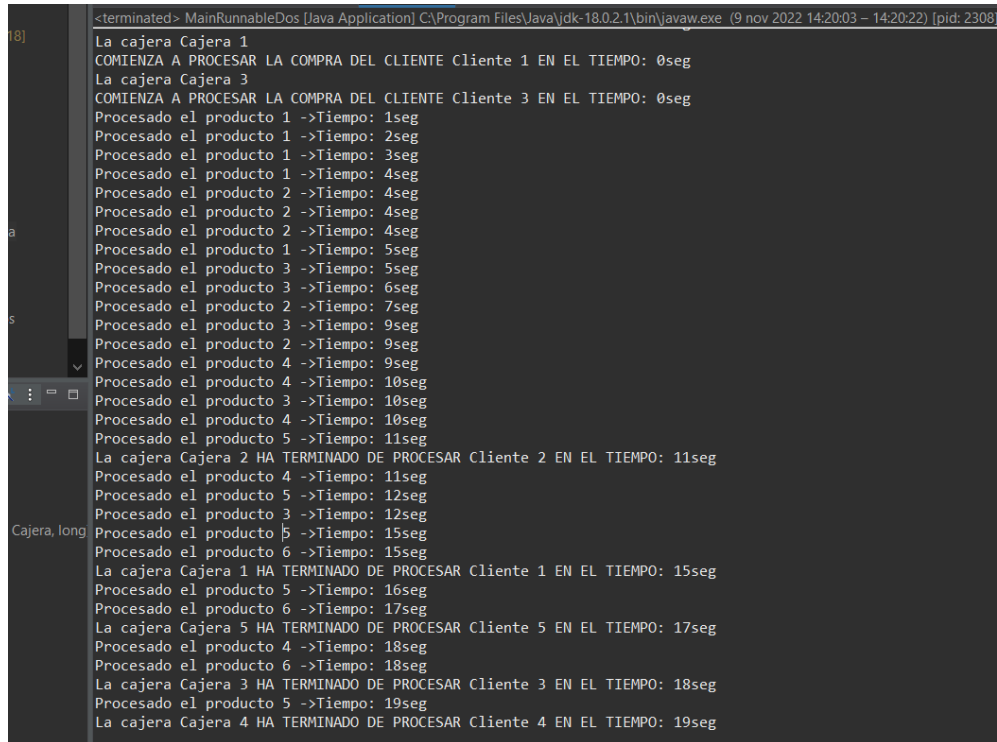
        // Tiempo inicial de referencia
        long initialTime = System.currentTimeMillis();

        Runnable proceso1 = new MainRunnableDos(cliente1, cajera1,
initialTime);
        Runnable proceso2 = new MainRunnableDos(cliente2, cajera2,
initialTime);
        Runnable proceso3 = new MainRunnableDos(cliente3, cajera3,
initialTime);
        Runnable proceso4 = new MainRunnableDos(cliente4, cajera4,
initialTime);
        Runnable proceso5 = new MainRunnableDos(cliente5, cajera5,
initialTime);

        new Thread(proceso1).start();
        new Thread(proceso2).start();
        new Thread(proceso3).start();
        new Thread(proceso4).start();
        new Thread(proceso5).start();
    }
    @Override
    public void run() {
        this.cajera.procesarCompra(this.cliente, this.initialTime);
    }
}
```

Al igual que la clase “MainRunnable”, esta clase realiza la misma función, con el único cambio de que el número de clientes y cajeras asciende de dos a cinco.

Resultado:



```
<terminated> MainRunnableDos [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (9 nov 2022 14:20:03 – 14:20:22) [pid: 2308]
La cajera Cajera 1
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg
La cajera Cajera 3
COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 3 EN EL TIEMPO: 0seg
Procesado el producto 1 ->Tiempo: 1seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 1 ->Tiempo: 3seg
Procesado el producto 1 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 1 ->Tiempo: 5seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 3 ->Tiempo: 6seg
Procesado el producto 2 ->Tiempo: 7seg
Procesado el producto 3 ->Tiempo: 9seg
Procesado el producto 2 ->Tiempo: 9seg
Procesado el producto 4 ->Tiempo: 9seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 3 ->Tiempo: 10seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 5 ->Tiempo: 11seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 11seg
Procesado el producto 4 ->Tiempo: 11seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 3 ->Tiempo: 12seg
Procesado el producto 5 ->Tiempo: 15seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg
Procesado el producto 5 ->Tiempo: 16seg
Procesado el producto 6 ->Tiempo: 17seg
La cajera Cajera 5 HA TERMINADO DE PROCESAR Cliente 5 EN EL TIEMPO: 17seg
Procesado el producto 4 ->Tiempo: 18seg
Procesado el producto 6 ->Tiempo: 18seg
La cajera Cajera 3 HA TERMINADO DE PROCESAR Cliente 3 EN EL TIEMPO: 18seg
Procesado el producto 5 ->Tiempo: 19seg
La cajera Cajera 4 HA TERMINADO DE PROCESAR Cliente 4 EN EL TIEMPO: 19seg
```

Finalmente, se visualiza de la misma forma, pero con más clientes y cajeras que la clase “MainRunnable”.