Inspección de Código

Problema 1

Dado el siguiente problema que se describe y el código Java que lo resuelve, inspecciónelo y llene la tabla que consigna el resultado de la inspección.

La empresa LavaCar ha decidido modernizarse y cambiar su imagen. Pretende instalar un sistema de lavado de autos completamente automatizado, el cual será capaz de lavar 1 auto en un lapso de tiempo que va de 1 a 10 minutos.

A la empresa le interesa conocer cuál es la forma más eficiente este nuevo mecanismo de lavado de autos. Si se selecciona la opción de lavado rápido, el auto se puede lavar en 1 minuto, pero requiere del uso de agua y jabón a alta presión. Cualquier otra opción de lavado, supone un mayor tiempo para lavar el auto, pero requiere de menos agua y jabón.

Interesa conocer cuántos clientes serán atendidos y cuánto tiempo tendrán que esperar en la línea de espera cuanto el sistema está siendo empleado bajo la opción que requiere más tiempo.

Diseñe un programa que simule los automóviles que están en una línea de espera aguardando ser lavados. Esto permitirá conocer al administrador cuanto toma el lavado de un auto, cuántos autos hay esperando y otros factores que interactúan.

Especificación

La especificación está compuesta por los siguientes inputs/outputs:

Input: Son los siguientes:

- Tiempo necesario para lavar un auto (en segundos)
- La probabilidad de que un cliente llegue en cualquier momento (se asume que a lo más un cliente llegará en cualquier segundo).
- Tiempo total de simulación (en segundos)

Output: El programa entregará la siguiente la siguiente información:

- Cantidad de clientes atendidos durante el tiempo de simulación.
- Tiempo promedio de espera de un cliente en la fila

Solución Propuesta

Declare una cola compuesta por números enteros. Esta cola se usará para hacer un seguimiento de la hora de llegada de los clientes. Además, se han de declarar los siguientes objetos:

1. Un lavador: El constructor del lavador tiene un argumento que indica la cantidad de tiempo necesario para lavar el auto.

2. Un Indicador Booleano: El constructor tiene un argumento que indica que tan seguido el indicador booleano devuelve el valor true (lo que indica que tan a menudo llegan los clientes).

3. Un método que permita calcular promedios

Para (segundoActual = 0; segundoActual < tiempo_total; segundoActual++)
{
Cada iteración de este ciclo simula el avance del tiempo en un (1) segundo, del siguiente modo: se consulta al indicador booleano si ha llegado algún cliente durante este segundo y, si es así, ingrese segundoActual en la cola.

Si (el lavador no está ocupado y la cola no está vacía)
{
Extraiga el siguiente número entero de la cola y asigne a este valor el nombre **siguiente**. Este número representa la hora de llegada del cliente cuyo auto va a ser lavado. Se calcula cuanto tiempo tuvo que esperar el cliente (**segundoActual** – **siguiente**) y entregue este valor al método que calcula promedios. Indique al lavador que debe comenzar a lavar.
}

Indique al lavador que ya ha transcurrido otro segundo. Esto permitirá al lavador determinar si sigue o no ocupado.
}

En este punto, la simulación ha terminado. Se puede obtener e imprimir la información desde el método que calcula valores promedios, la cual es:

1. Cantidad de clientes atendidos
2. Tiempo promedio de espera (en segundos)

Código

```
public class Lavador
{
   private int segundosPorLavado; // Tiempo en segundos por un solo lavado
   private int tiempoRestanteLavado;   // Segundos que restan para que el lavador se desocupe


   public Lavador(int s)
   {
       segundosPorLavado = s;
       tiempoRestanteLavado = 0;
   }
```

```
   public boolean estaOcupado( )
   {
      return (tiempoRestanteLavado > 0);
   }


   public void reducirTiempoRestante( )
   {
      if (tiempoRestanteLavado > 0)
         tiempoRestanteLavado--;
   }


   public void iniciarLavado( )
   {
      if (tiempoRestanteLavado > 0)
         throw new IllegalStateException("Lavador esta ocupado.");
      tiempoRestanteLavado = segundosPorLavado;
   }

}


public class OrigenBooleano
{
   private double probabilidad; // La probabilidad de que un cliente llegue dentro del segundo de simulación.

   public OrigenBooleano(double p)
   {
       if ((p < 0) || (p > 1))
           throw new IllegalArgumentException("valor de probabilidad inaceptable: " + p);
       probabilidad = p;
   }


   public boolean consultaProbabilidad( )  // Para obtener un valor de probabilidad al azar
   {
      return (Math.random( ) < probabilidad);
   }

}

public class Promedio
{
   private int cont;  // cuantas veces ha recibido un valor
   private double suma; // suma de todos los valores recibidos


   public Promedio( )
   {
       cont =0;
       suma = 0;
```

```
   }


   public void sumaNumeros(double valor)
   {
      if (cont == Integer.MAX_VALUE)
         throw new IllegalStateException("numero demasiado alto.");
      cont++;
      suma += valor;
   }


   public double valorPromedio( )
   {
      if (cont == 0)
         return Double.NaN;
      else
         return suma/cont;
   }


   public int cantidadNumeros( )
   {
      return cont;
   }
}
```

Main

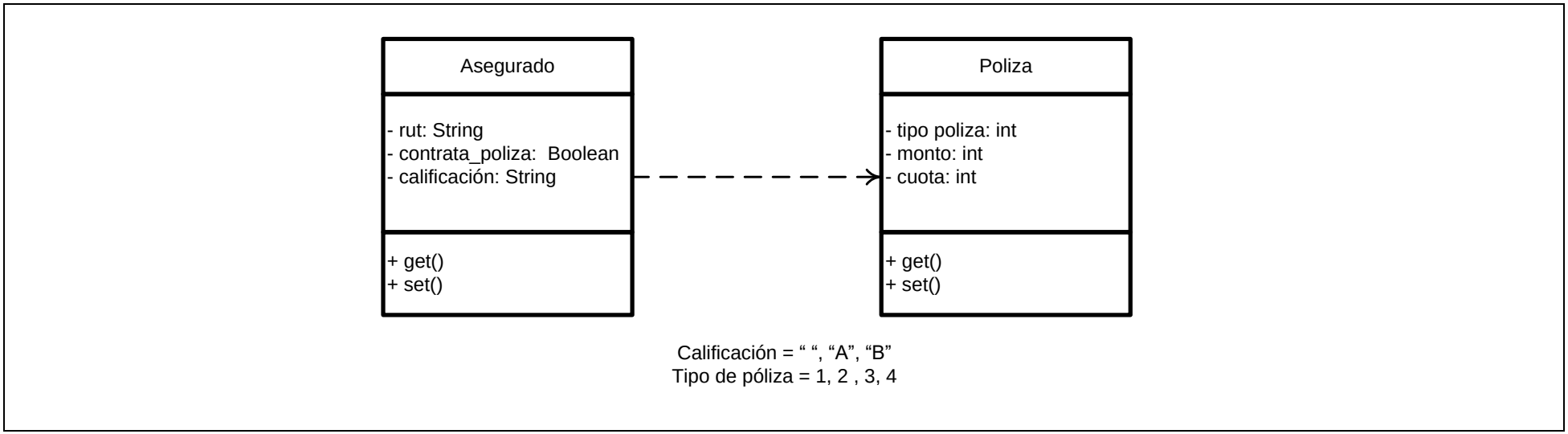
```
public static void main(String[] args) {
        // Ejecutar simulacion e imprimir resultados
        // tiempo de lavado = 240 segundos
        // probabilidad de llegada de un cliente = 0,0025
        // tiempo total de simulacion = 28800 segundos (8 horas)
        simularLavadoAuto(240, 0.0025, 28800);

    }

    public static void simularLavadoAuto(int tiempoDeLavado, double probabilidadLlegada, int tiempoTotal)
    {
        Queue<Integer> tiempoDeLlegada = new LinkedList<Integer>();
        int sigte;
        OrigenBooleano llegada = new OrigenBooleano(probabilidadLlegada);
        Lavador maquina = new Lavador(tiempoDeLavado);
        Promedio tiemposDeEspera = new Promedio();
        int segundoActual;

        // Mostrar los parametros
        System.out.println("Segundo para lavar un auto = " + tiempoDeLavado);
        System.out.println("Probabilidad de llegada de un nuevo cliente = " + probabilidadLlegada);
        System.out.println("Tiempo total de simulación = " + tiempoTotal + " segundos");
```

```
        // Verificación de las condiciones previas
        if (tiempoDeLavado <= 0 || probabilidadLlegada < 0 || probabilidadLlegada > 1 || tiempoTotal < 0)
            throw new IllegalArgumentException("Valores fuera de rango.");
        for (segundoActual = 0; segundoActual < tiempoTotal; segundoActual++)
        {
            // ciclo simula el avance de 1 segundo
            if (llegada.consultaProbabilidad())
                tiempoDeLlegada.add(segundoActual);
            // verificar si se puede comenzar a lavar otro auto
            if (!maquina.estaOcupado() && !tiempoDeLlegada.isEmpty())
            {
                sigte = tiempoDeLlegada.remove();
                tiemposDeEspera.sumaNumeros(segundoActual - sigte);
                maquina.iniciarLavado();
            }
            // restar 1 segundo al tiempo total de simulacionque va quedando
            maquina.reducirTiempoRestante();
        }
        // Entrega de resultados
        System.out.println("Clientes atendidos = " + tiemposDeEspera.cantidadNumeros());
        if (tiemposDeEspera.cantidadNumeros() > 0)
            System.out.println("Tiempo promedio de espera = " + tiemposDeEspera.valorPromedio() + " segundos");
    }
```

Problema 2

Dado el siguiente problema que se describe y el código Java que lo resuelve, inspecciónelo y llene la tabla que consigna el resultado de la inspección.

Toda persona que contrata un seguro, adquiere una póliza de seguro que detalla el tipo de póliza, el monto de seguro y la cuota mensual que debe pagar, para conservar los beneficios del seguro. Tal como lo muestra el siguiente diagrama, se cuenta con los datos del asegurado y la póliza.



La empresa aseguradora, necesita implementar las siguientes operaciones:

a. **recarga**: devolverá la recarga que se aplicará sobre el monto de la póliza, para aquellos clientes cuya calificación sea 'B' (Baja). Dicha recarga corresponderá a un x% (que se ingresa por teclado). En caso de que la calificación sea 'A' (Alta), no tendrá recarga.
b. **cuota**: informará sobre el valor propuesto de la cuota que debe pagar mensualmente el asegurado, la cual se calcula del siguiente modo:

monto + recarga (método) / 1000000

Además, dejará en true **contrata_poliza**.

Se pide:

1. Construir un proyecto en Java, utilizando NetBeans, donde implementará la clase que represente la situación descrita, implementando además constructores, accesadores y mutadores.
2. Agregar al proyecto una aplicación la cual permita mostrar y ejecutar el siguiente menú:

1. Crear un Asegurado
2. Crear Póliza
3. Mostrar información (debe mostrar la calificación, tipo de póliza y monto.
4. Si tiene calificación, debe habilitar que se pueda modificar el monto de la cuota. Si no tiene calificación, desplegar mensaje “No tiene calificación” (debe mostrar el monto de la cuota antes y después de la modificación).
5. Asegurar, debe mostrar el monto total que deberá pagar por un año (método)). Esto corresponde al valor unitario de la cuota multiplicado por doce.
6. Salir

Código

```
public class Asegurado {
    private String rut;
    private String calificacion;
    private Boolean contrata_poliza;


    public Asegurado(String rut, String calificacion, Boolean contrata_poliza, int tipoPoliza, int monto, int cuota)
{
        this.rut = rut;
        this.calificacion = calificacion;
        this.contrata_poliza = contrata_poliza;
    }

    @Override
    public String toString() {
        return "Asegurado{" + "rut=" + rut + ", calificacion=" + calificacion + ", contrata_poliza=" +
contrata_poliza + '}';
    }



    public String getRut() {
        return rut;
    }

    public String getCalificacion() {
        return calificacion;
    }

    public Boolean getContrata_poliza() {
        return contrata_poliza;
    }



    public void setRut(String rut) {
        this.rut = rut;
    }

    public void setCalificacion(String calificacion) {
        this.calificacion = calificacion;
    }

    public void setContrata_poliza(Boolean contrata_poliza) {
        this.contrata_poliza = contrata_poliza;
    }



    public int recarga(int x)
```

```
    {
        int recargar=0;
        if (calificacion=="B" || calificacion=="b")
            return recargar= recargar * x;
        else
            System.out.println("No tiene recarga...");
        return recargar;
    }




}

public class Poliza {
    private int tipoPoliza;
    private int monto;
    private int cuota;

    public Poliza(int tipoPoliza, int monto, int cuota) {
        this.tipoPoliza = tipoPoliza;
        this.monto = monto;
        this.cuota = cuota;
    }

    @Override
    public String toString() {
        return "Poliza{" + "tipoPoliza=" + tipoPoliza + ", monto=" + monto + ", cuota=" + cuota + '}';
    }

    public int getTipoPoliza() {
        return tipoPoliza;
    }

    public int getMonto() {
        return monto;
    }

    public int getCuota() {
        return cuota;
    }

    public void setTipoPoliza(int tipoPoliza) {
        this.tipoPoliza = tipoPoliza;
    }

    public void setMonto(int monto) {
        this.monto = monto;
    }

    public void setCuota(int cuota) {
        this.cuota = cuota;
```

```
    }


}

import java.io.*;
class Leer
{     public static String dato()
      {     String sdato="";
            try
            {     InputStreamReader isr=new InputStreamReader(System.in);
                  BufferedReader flujoe=new BufferedReader(isr);
                  sdato=flujoe.readLine();
            }
            catch(IOException e)
            {
                   System.err.println("Error: "+ e.getMessage());
            }
            return(sdato);
      }
      public static char datoChar()
      {
            try
            {
                    return dato().charAt(0);
            }
            catch(Exception error)
            {
                  return('z');
            }

      }
      public static int datoInt()
      {
            try
            {     return (Integer.parseInt(dato()));
            }
            catch(NumberFormatException error)
            {
                  return(Integer.MIN_VALUE);
            }
      }
      public static byte datoByte()
      {
            try
            {     return (Byte.parseByte(dato()));
            }
            catch(NumberFormatException error)
            {
                  return(Byte.MIN_VALUE);
            }
      }
```

```
	public static boolean datoBoolean()
	{
		try
		{	return (Boolean.parseBoolean(dato()));
		}
		catch(NumberFormatException error)
		{
			return(false);
		}
	}
	public static short datoShort()
	{
		try
		{
			return (Short.parseShort(dato()));
		}
		catch(NumberFormatException error)
		{
			return(Short.MIN_VALUE);
		}
	}
	public static long datoLong()
	{
		try
		{
			return (Long.parseLong(dato()));
		}
		catch(NumberFormatException error)
		{
			return(Long.MIN_VALUE);
		}
	}
	public static float datoFloat()
	{
		try
		{	Float f =new Float(dato());
			return (f.floatValue());
		}
		catch(NumberFormatException error)
		{
			return(Float.NaN);
		}
	}
	public static double datoDouble()
	{
		try
		{	Double f =new Double(dato());
			return (f.doubleValue());
		}
		catch(NumberFormatException error)
		{
			return(Double.NaN);
```

```
            }
        }
}

public class Seguro {


    public static void main(String[] args) {
        int op, run, op2,tipoP, montoS,Vcuota;
        Asegurado a1=null;
        Poliza s1=null;
        String tipo="";
        boolean contrata;
        do{

            System.out.println("--- Menu Principal ---");
            System.out.println("1. Crear un asegurado");
            System.out.println("2. Crear poliza");
            System.out.println("3. Mostrar información");
            System.out.println("4. Calificacion");
            System.out.println("5. Total de pago por año");
            System.out.println("6. Salir");

            op=Leer.datoInt();
            switch(op)
            {
                case 1:
                    System.out.println("Run: ");
                    run=Leer.datoInt();
                    System.out.println("Tipo calificacion(A-B): ");
                    tipo=Leer.dato();

                    System.out.println("Contrata poliza(1-si,2-no):");
                    op2=Leer.datoInt();
                    if (op2==1)
                        contrata=true;
                    else
                        contrata=false;

                    a1= new Asegurado(tipo, tipo, contrata, op2, op2, op2);

                    break;

                case 2:

                    System.out.println("Tipo poliza(1,2,3,4): ");
                    tipoP=Leer.datoInt();

                    System.out.println("Monto seguro: ");
                    montoS=Leer.datoInt();

                    System.out.println("Cuota: ");
```

```
                    Vcuota=Leer.datoInt();

                    s1=new Poliza(tipoP, montoS, Vcuota);
                    break;


                case 3:
                    if (s1==null && a1 != null)
                        System.out.println("Calificacion "+a1.getCalificacion());
                    if (a1==null && s1!=null)
                        System.out.println("Tipo de poliza: "+s1.getTipoPoliza());
                        System.out.println("Monto seguro: "+s1.getMonto());
                    if (a1!=null && s1!=null)
                        System.out.println("Tipo de poliza: "+s1.getTipoPoliza());
                        System.out.println("Monto seguro: "+s1.getMonto());
                        System.out.println("Calificacion "+a1.getCalificacion());
                    if (a1==null && s1==null)
                        System.out.println("Error... debe ingresar algun dato");
                    break;

                case 4:
                    System.out.println("Tipo de poliza: "+s1.getMonto());
                    if (tipo=="")
                        System.out.println("No tiene calificacion");
                    else
                        System.out.println("Modificar monto cuota: "+s1.getMonto());
                        Vcuota=Leer.datoInt();
                        s1.setMonto(Vcuota);
                    break;
                case 5:
                    System.out.println("Monto a pagar en el año: "+s1.getCuota()*12);
                    break;


                case 6:
                    System.out.println("Fin del Programa... ");
                    break;
                default:
                    System.out.println("Error...Ingrese opcion valida");
                break;
            }



        }while(op!=6);
    }

}
```