Lista 7

As classes a seguir foram criadas para fazer um mostrador de relógio digital. Analise as duas classes, veja como elas trabalham juntas usando a Composição.

Faça uma classe no mesmo projeto para testar a criação de um mostrador de relógio digital. Crie um mostrador digital, chame métodos e entenda como ela funciona. Procure uma forma de simular o funcionamento do relógio para que ele funcione, isso deverá ser feito chamando o método “timeTick” a cada minuto, tente usar a repetição e uma forma de que o programa faça chamadas depois de um determinado tempo.

```
/**
 * A classe MostradorNumero representa um mostrador de número digital
 * que pode armazenar valores de zero a um determinado limite.
 * O limite pode ser especificado durante a criação do mostrador.
 * Os valores variam de zero (inclusive) a limite - 1.
 *
 * Se usado, por exemplo, para os segundos em um relógio digital, o
 * limite seria 60, resultando na exibição de valores de 0 a 59.
 *
 * Quando incrementado, o mostrador muda automativamente para zero
 * quando alcançar o limite.
 *
 * Versão em português do livro Programação orientada a objetos
 * com Java de David J. Barnes and Michael Kolling
 *
 * @author Simone
 * @version 18/02/2021
 */
public class MostradorNumero {
      private int limite;
      private int valor;
      /**
       * Construtor cria um objeto que define o valor e o limite
       * do mostrador digital.
       * @param limite limite do valor do mostrador digital
       */
      public MostradorNumero(int limite) {
            this.limite = limite;
            this.valor = 0;
      }
      /**
       * Retorna o valor atual.
       * @return o valor atual no mostrador
       */
      public int getValor() {
            return valor;
      }
```

```
	/**
	 * Configura o valor do mostrador com o novo valor especificado.
	 * Se o novo valor for menor que zero ou excedor o limite,
	 * não faz nada.
	 * @param valor o valor a ser mostrado.
	 */
	public void setValor(int valor) {
		if ((valor >= 0) && (valor < limite)){
			this.valor = valor;
		}
	}

	/**
	 * Retorna o valor do mostrador (String de dois dígitos, se o valor
	 * for menor do que dez, ele será preenchido com um zero à
esquerda.)
	 */
	public String getMostradorValor() {
		if (valor < 10) {
			return "0" + valor;
		} else {
			return "" + valor;
		}
	}

	/**
	 * Incrementa o valor do mostrador em 1 unidade,
	 * mudando para zero se o limite for alcançado.
	 */
	public void incrementar() {
		valor = (valor + 1) % limite;
	}

}

--------------------------------########--------------------------------
/**
 * A classe MostradorRelogio implementa o mostrador de um relógio
 * digital de 24 horas. O relógio mostra horas e minutos.
 * O intervalo do relógio é de 00:00 (meia-noite) a 23:59 (um
 * minuto antes da meia-noite).
 *
 * O mostrador do relógio recebe "tiques" (pelo método timeTick)
 * a cada minuto e reage incrementando o mostrador. Isso é feito no
 * estilo de relógio usual: a hora incrementa quando os minutos
 * retornam a zero.
 *
 * Versão em português do livro Programação orientada a objetos
 * com Java de David J. Barnes and Michael Kolling
 *
```

```
 * @author Simone
 * @version 18/02/2021
 */
public class MostradorRelogio {
	private MostradorNumero horas;
	private MostradorNumero minutos;
	private String mostradorString;
	/**
	 * Construtor cria um novo relógio ajustado em 00:00.
	 */
	public MostradorRelogio() {
		this.horas = new MostradorNumero(24);
		this.minutos = new MostradorNumero(60);
		atualizaMostrador();
	}
	/**
	 * Construtor cria um novo relógio ajustado com a hora
	 * especificada pelos parâmetros.
	 */
	public MostradorRelogio(int hora, int minuto) {
		this.horas = new MostradorNumero(24);
		this.minutos = new MostradorNumero(60);
		setTime(hora,minuto);
	}

	/**
	 * Esse método deve ser chamado uma vez por minuto - ele faz o
	 * mostrador do relógio avançar um minuto.
	 */
	public void timeTick() {
		minutos.incrementar();
		if (minutos.getValor() == 0) {
			horas.incrementar();
		}
		atualizaMostrador();
	}
	/**
	 * Configura o relógio como a hora e o minuto especificado.
	 */
	public void setTime(int hora, int minuto) {
		horas.setValor(hora);
		minutos.setValor(minuto);
	}
	/**
	 * Retorna a hora atual desse msotrador no formado HH:MM
	 */
	public String getTime() {
		return mostradorString;
	}
	/**
```

```
         * Atualiza a string interna que representa o mostrador.
         */
        public void atualizaMostrador() {
              mostradorString = horas.getMostradorValor() +
                          ":" + minutos.getMostradorValor();
        }
}
```