UCS UNIVERSIDADE DE CAXIAS DO SUL	**UNIVERSIDADE DE CAXIAS DO SUL** **Área do Conhecimento de Ciências Exatas e Engenharias** **Programação Orientada a Objetos** **Prof. Alexandre Krohn**

Lista de Exercícios 5b – Sobrecarga de métodos

1) Abra o Eclipse e crie um novo projeto chamado *PraticaSobrecarga*. Sugere-se agora a criação de três construtores diferentes para a classe *Relogio* apresentada na listagem abaixo. Desta maneira, o relógio pode ser inicializado de três maneiras diferentes:

a) Informando-se hora, minuto e segundo (como já feito);

b) Informando-se somente a hora e o minuto, e inicializando o segundo com 1;

c) Informando-se somente a hora, e inicializando o minuto e o segundo com 1.

Obs.: Considere a utilização do método *acertaHora*.

```
public class Relogio {

	// Atributos

	private int hora, minuto, segundo;

	/**
	 *
	 * Construtor para objetos da classe relógio
	 *
	 * @param int h (hora), int m (minuto), int s (segundo)
	 *
	 */
	public Relogio(int h, int m, int s) {

		acertaHora(h, m, s);

	}

	/**
	 *
	 * Método para incrementar um segundo
	 *
	 */
	public void incrementa() {

		if (++segundo > 59) {
			segundo = 0;
			if (++minuto > 59) {
```

```
                        minuto = 0;
                        if (++hora > 23)
                                hora = 0;
                }
        }
    }

    /**
     *
     * Método para decrementar um segundo
     *
     */
    public void decrementa() {
        if (--segundo < 0) {
                segundo = 59;
                if (--minuto < 0) {
                        minuto = 59;
                        if (--hora < 0)
                                hora = 23;
                }
        }
    }

    /**
     *
     * Método para atualizar a hora atual
     *
     * @param int h (hora), int m (minuto), int s (segundo)
     *
     */
    public void acertaHora(int h, int m, int s) {

        if (h >= 0 && h <= 23)
                hora = h;
        else
                hora = 0;
        if (m >= 0 && m <= 59)
                minuto = m;
        else
                minuto = 0;
        if (s >= 0 && s <= 59)
                segundo = s;
        else
                segundo = 0;
    }

    /**
     *
     * Método para informar a hora atual
     *
     * @return String hora
     *
     */
    public String toString() {
        String str = "Hora atual: " + hora + ":" + minuto + ":" + segundo;
        return str;
    }
}
```

Agora, crie uma classe *TestaRelogio* que implementa o método *main*. Dentro deste método, crie novos objetos da classe *Relogio* com valores diferentes utilizando todos os construtores implementados. Depois, apresente no console a hora armazenada em cada relógio criado. Verifique que aqueles não informados são inicializados corretamente com 1.

Exercício :

Pode ser desejável que a hora sempre apareça com duas casas (por exemplo: 12:05:00). Neste caso, podemos aproveitar os métodos fornecidos pela classe *DecimalFormat*. Consulte a API para verificar o seu funcionamento, depois importe a classe na implementação da classe *Relogio* (*import java.text.DecimalFormat;*) e altere o método *toString* da seguinte maneira:

DecimalFormat numInt = new DecimalFormat("00");
String str = "Hora atual: "+numInt.format(hora)+":"+numInt.format(minuto) +":"+numInt.format(segundo);

Agora, experimente alterar e exibir a hora das instâncias da classe *Relogio* já criadas (utilize os métodos *acertaHora* e *toString()*).

2) Considere agora a implementação de construtores para a classe *Ponto* apresentada logo abaixo. Inclua três construtores diferentes descritos abaixo:

a) Construtor sem parâmetros, que cria um ponto nas coordenadas (1,1);

b) Construtor que recebe dois parâmetros de coordenadas X e Y;

c) Construtor que inicializa o ponto através das coordenadas de um outro *Ponto* recebido como argumento.

```
public class Ponto {

    // Atributos
    private double x;
    private double y;

    // Métodos
    /**
     * Método para especificar um novo valor para
     * o atributo x.
     *
     * @param double xVal
     * @return -
     */
    public void setX(double xVal) {
       x = xVal;
    }

    /**
     * Método para especificar um novo valor para
     * o atributo y.
     *
     * @param double yVal
     * @return -
     */
    public void setY(double yVal) {
       y = yVal;
    }

    /**
     * Método para alterar os valores dos atributos x e y,
```

```
     * somando a eles os valores passados por parâmetro.
     *
     * @param double dx, double dy
     * @return -
     */
    public void desloca(double dx, double dy) {
       x = x + dx;
       y = y + dy;
    }

    /**
     * Método que retorna o conteúdo do atributo x.
     *
     * @param -
     * @return double x
     */
    public double getX() {
       return (x);
    }

    /**
     * Método que retorna o conteúdo do atributo y.
     *
     * @param -
     * @return double y
     */
    public double getY() {
       return (y);
    }

    /**
     *
     * Método que exibe uma mensagem que mostra o
     * conteúdo dos atributos x e y.
     *
     * @param -
     * @return -
     */
    public String toString() {
       String str = "(" + x + "," + y + ")";
       return str;
    }
}
```

Para que seja possível testar a criação de instâncias da classe *Ponto* utilizando os seus diferentes construtores, acrescente a classe abaixo no seu projeto Eclipse. Depois, complete o código conforme especificado.

```
public class TestaPonto {

    /**
     * Método main
     */
    public static void main(String args[]) {

       Ponto p1, p2, p3;

       // Instancie os objetos utilizando cada um dos diferentes
       // construtores implementados

       // Apresente os valores dos atributos de cada um dos objetos
       // através da chamada do método toString()
```

```
    }
}
```

Exercício :

a) Faça a sobrecarga dos métodos *setX*, *setY* e *desloca* considerando que eles podem receber por parâmetro *Strings* ao invés de *doubles*. Dentro do método estas *Strings* devem ser convertidas para *double*.

b) Na classe TestaPonto, inclua chamada para os métodos *setX*, *setY* e *desloca* que recebem *String* por parâmetro. Após a chamada destes métodos exiba na tela o conteúdo dos objetos chamando o método *toString*.