# Projeto e Construção de Sistemas

## Lista de Exercícios 3

**Exercício 1:**
Defina uma classe Empregado com os atributos nome, salário base, taxa percentual de desconto e altura (comprimento), contendo:

a) construtor default com taxa percentual de desconto = 10%.
b) construtor com nome, altura, salário e taxa percentual.
c) métodos get e set para os atributos.
d) método getPagamentoLiquido sendo o resultado do cálculo: salário descontado da taxa percentual.

Escreva um programa de teste que crie alguns empregados com diferentes taxas de desconto e imprima o nome e o respectivo pagamento liquído de cada empregado.

**Exercício 2:**
Desenvolva uma nova versão para o programa da lista 1 que permite que o usuário entre uma sequência de pares contendo a distância (em km) percorrida e combustível consumido (em litros) e imprima para cada um desses pares o consumo em km/l. Ao final, o programa deve imprimir o consumo médio considerando todos os abastecimentos.

O programa deve ser estruturado da seguinte forma:

Defina uma classe TrechoViagem com os seguintes atributos: distanciaPercorrida (em km) e volumeCombustivel (em listros). Esta classe deverá ter as seguintes operações:

- um construtor que recebe os valores para inicialização dos seus atributos.
- Operações get/set para os dois atributos (distanciaPercorrida e volumeCombustivel).
- Operação getConsumoMedio que retorna o consumo médio referente ao trecho.

Defina uma classe Viagem. Esta classe deverá ter:

- Atributo: um array criado dinamicamente capaz de armazenar n trechos de viagem.
- Atributo: o número de trechos que efetivamente fazem parte da viagem. (ex: o array pode ter capacidade máxima para 100 trechos e apenas 10 trechos podem ter sido adicionados à viagem até um determinado momento).
- Operação: Um construtor que recebe o número máximo de trechos (n) e inicializa o array criado dinamicamente capaz de armazenar (n) objetos TrechoViagem.
- Operação: adicionarTrecho que recebe dois parâmetros (distancia percorrida e volume de combustível consumido) e armazena no array de trechos um novo

objeto TrechoViagem criado dinamicamente a partir dos dois parâmetros recebidos.

- Operação: getConsumoMedio que calcula o consumo médio da viagem a partir dos trechos que tenham sido adicionados à viagem.
- Operação: getDistanciaTotal que retorna a soma da distância percorrida em cada trecho da viagem.
- Operação: getVolumeCombutivelTotal que retorna a soma dos volumes de combustível consumidos em cada trecho da viagem.
- Operação: getNumeroTrechos que retorna o valor do atributo número de trechos que efetivamente fazem parte da viagem.
- Operação: getTrechos que retorna um array de objetos TrechoViagem que façam parte da Viagem.

Implemente um programa (main) que:

a) Pergunte ao usuário quantos trechos fazem parte da viagem.
b) Construa o objeto da classe Viagem.
c) Faça um loop perguntando ao usuário os detalhes de cada trecho (repare que agora o número de iterações é determinado pela entrada do item a.)
d) Em cada passo do loop, chame a operação adicionarTrecho do objeto viagem construído no item b, que deverá criar um objeto TrechoViagem dinamicamente a partir da distancia e combustível informados.
e) Faça um outro loop que vai percorrer o array obtido a partir da operação getTrechos do objeto viagem e imprima a distância, a quantidade de combustível e o consumo médio de cada trecho.
f) Ao final, imprimir a distancia total, o volume de combustível total e o consumo médio total a partir do objeto viagem.

Exemplo de execução:
*Entrada*
Entre número de trechos: 2

Entre distância percorrida (em km): 1000
Entre quantidade de combustível (em litros): 100

Entre distância percorrida (em km): 1200
Entre quantidade de combustível (em litros): 100

*Saída*
Trecho 1: 1000 km e 100 litros – Consumo: 10 km/l
Trecho 2: 1100 km e 100 litros – Consumo: 11 km/l

Distância Total: 2100 km
Volume Combustível Total: 200 litros
Consumo Médio: 10,5 km/l

**Exercício 3:**
Implemente uma classe Data (em Java) para representar uma data de calendário.
Atributos da classe:

- dia, mês e ano (três números inteiros)

Operações da classe:

- Construtor recebendo dia, mês e ano. O construtor deve verificar se é uma data válida. Caso não seja, o objeto construído deverá ser inicializado com a data 1/1/1980.
- Operações get para os três atributos, ou seja, getDia, getMes e getAno, que devem retornar o valor do atributo correspondente.
- Operação diaDaSemana. Esta operação não recebe parâmetro e deverá retornar o dia da semana correspondente aos valores correntes dos atributos do objeto que receber a mensagem.

Como calcular o dia da semana:

```
if (month < 3)
{
        month = month + 12;
        year = year - 1;
}

dia da semana = (
        day
        + (2 * month)
        + (int) (6 * (month + 1) / 10)
        + year
        + (int) (year / 4)
        -  (int) (year / 100)
        + (int) (year / 400)
        + 1
        ) % 7;
```

Implemente um pequeno programa de teste que crie um objeto data a
partir do dia, mês e ano entrado pelo usuário e imprima a seguinte saída:
Data: dd/mm/aaaa => <dia da semana>

Exemplo:
Entrada:
        Dia => 14
        Mês => 8
        Ano => 2008
Saída:
        14/08/2008 => quinta-feira

**Exercício 4:**

**Parte 1:**
Implemente a classe Ponto com os atributos x e y e as seguintes operações:

e) Construtor recebendo os parâmetros x e y.
f) Operações get/set para os atributos x e y.

**Parte 2:**
Implemente a classe Retangulo com os atributos topLeft (objeto da classe Ponto) e bottomRight (objeto da classe Ponto) e as seguintes operações:

a) Construtor default que inicializa topLeft e bottomRight como (0,0) e (0,0)
b) Construtor recebendo os parâmetros topLeft e bottomRight.
c) Operações get/set para os atributos topLeft e bottomRight
d) Operação comprimento que retorna a diferença entre as coordenadas x dos pontos bottomRight e topLeft.
e) Operação largura que retorna a diferença entre as coordenadas y dos pontos bottomRight e topLeft.
f) Operação area que retorna a área do retângulo (largura x comprimento).
g) Operação perímetro que retorna o perímetro do retângulo (2 * largura + 2 * comprimento).

**Parte 3:**
Implemente um programa de teste (função main) que cria um objeto retângulo a partir dos pontos topLeft e bottomRight informados pelo usuário, e imprime o comprimento, a largura, a área e o perímetro do retângulo.

Exemplo:
*Entrada:*
*TopLeft.x => 5*
*TopLeft.y => 10*
*BottomRight.x => 15*
*BottomRight.y => 30*
*Saída:*
*Retangulo:*
*Comprimento = 10*
*Largura = 20*
*Área = 200*
*Perímetro = 60*

**Parte 4:**
Modifique o programa anterior de forma que o usuário entre um número N e o programa aloque dinamicamente (na heap) um array de N referências para objetos Retangulo. Depois, faça um loop que crie dinamicamente N objetos da classe Retangulo, onde o primeiro retângulo terá as coordenadas (0,0) e (1,1), o próximo retângulo (0,0) (2,2) e assim sucessivamente até (0,0) (N,N). Por fim, faça uma função que receba um array de

ponteiros para Retangulo e faça um loop imprimindo a área e o perímetro de cada retângulo, além da área e perímetro totais.

**Parte 5:**
Modifique a classe Ponto de forma que o construtor e as operações setX/setY verifiquem se o ponto (x, y) tem x e y >= 0 (não é permitido valor negativo).
Modifique a classe Retangulo de forma que o construtor e as operações setTopLeft e setBottomRight verifique se o ponto bottomRight está de fato à direita e abaixo do ponto topLeft.

Ex:
topLeft (5, 10) e bottomRight(2, 12) => 2 é menor que 5. => valor inválido para bottomRight.
topLeft (5, 10) e bottomRight(7, 8) => 8 é menor que 10 => valor inválido para bottomRight.

No caso de um bottomRight inválido, o programa deve fazer com que bottomRight seja definido como topLeft + 1 (para x e para y). No primeiro caso acima, bottomRight ficaria com os valores (6, 11) e no segundo (6, 11) também.