## Exercício 1:

a) A Figura 1 mostra a Figura geométrica círculo e seus atributos. Um círculo pode ser representado por duas coordenadas centroX e centroY, que são o centro do círculo, mais um valor representando o raio. Na Figura 2, temos um programa simples escrito em Java que cria uma instância da classe Circulo e imprime seus valores. Implemente a classe Circulo de maneira que o programa da Figura 2 funcione.

b) Vimos em aula que é uma prática comum em Programação Orientada a Objetos (POO) manter os atributos de uma classe privados a essa classe, permitindo que estes sejam acessados fora da classe através de métodos. A capacidade de ocultar dados dentro de classes, permitindo somente que operações especializadas ou dedicadas manipulem os dados ocultos chama-se **encapsulamento**. O encapsulamento de dados é um dos maiores benefícios da POO pois permite que somente operações especializadas acessem os atributos de uma classe, assim podemos forçar que apenas valores corretos sejam colocados nesses atributos. Um exemplo clássico é o de uma classe representando uma ContaBancaria que possui um atributo saldo. Se permitirmos que esse atributo seja acessado fora da classe um usuário poderia retirar mais dinheiro do que realmente possui, deixando o saldo negativo. Por outro lado se o valor de saldo é somente acessado por um método, podemos garantir que o saldo nunca será negativo!!! Modifique a classe Circulo fazendo que os atributos da classe sejam privados (private). Adicione métodos para acessar esses atributos. Modifique o programa da Figura 2 para que este funcione depois das modificações feitas em Circulo.

**Fig. 2.** Programa que usa a classe Circulo

Fig. 1. Variáveis de classe

c) Adicione um construtor na classe Circulo que receba como argumento valores iniciais para seus a

d) Dois círculos são considerados iguais se seus raios e seus valores centroX e centroY são igual todo na classe Circulo que receba um Circulo como argumento e retorne true se o circulo é igual ao circulc lo e false caso contrário. Ex:

```
class Programa2{
public static void main (String [] arg)
{ Circulo c1 = new Circulo (1,2,100);
   Circulo c2 = new Circulo (20,22, 99);
   Circulo c3 = new Circulo (1,2,100);
   if (c1.comparaCirculo(c2))
   System.out.println("c1 igual a c2");
   else
     if (c1.comparaCirculo(c3))
        System.out.println("c1 igual a c3");
 }
}
```

**Esse programa deve imprimir na tela a String "c1 igual a c3".**

e) Escreva um método move() para a classe Circulo que permita modificar os valores do centro do circulo.

f) Escreva um método area() que calcule a área de um círculo. A área de um círculo com raio r é calculada como area = Pi * r * r. O valor de Pi pode ser obtido em Java com a constante Math.PI

g) Escreva um método perimetro() que calcule o perímetro de um círculo. O perímetro de um círculo com raio r é calculado como p = 2* Pi * r

h) Implemente um programa que use os novos métodos implementados em Circulo.

**Exercício 2:**

Crie uma classe em Java denominada **Elevador** que represente as informações e o comportamento de um elevador dentro de um prédio. A classe deve armazenar o andar atual (para o térreo, considere o valor zero), total de andares no prédio (excluindo o térreo), capacidade de pessoas no elevador e quantas pessoas estão presentes nele. A classe deve também disponibilizar os seguintes métodos:

- **Construtor**: que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (o elevador sempre começa no térreo e está vazio);
- **Entrar**: para acrescentar uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
- **Sair**: para remover uma pessoa do elevador (só deve remover se houver alguém dentro dele);
- **Subir**: para subir um andar (não deve subir se já estiver no último andar);
- **Descer**: para descer um andar (não deve descer se já estiver no

térreo); Obs: Encapsular todos os atributos da classe (criar os métodos

sets e gets).

**Exercício 3:**

Escreva uma classe em Java que simule uma calculadora bem simples. Essa classe deve ter como atributos duas variáveis double e um char. Deve possuir um construtor que recebe como parâmetro dois números e um caracter, correspondente a uma das operações básicas (+, -, *, /). Deve ter um método para calcular a operação e um para imprimir o resultado.

**Exercício 5:**

**Globo de Bingo.** Desenvolva uma classe para representar um globo utilizado nos jogos de bingo. A classe Globo deverá fornecer através de um método, números aleatórios não repetidos entre 1 e um valor máximo definido na sua instanciação. No projetoda classe, defina construtores, métodos gets, sets e demais métodos necessários.

**Exercício 6:**

Escreva uma classe chamada “MatrizDeInteiros” que tenha como atributo uma matriz de inteiros e um construtor que receba como parâmetro a ordem da matriz, a instancie e inicialize com zeros. Acrescente a classe os seguintes métodos:

a. um método que receba como parâmetro três números inteiros indicando respectivamente linha, coluna e o valor que deve ser armazenado na linha e coluna indicada. Obs: Caso a linha ou a coluna passadas como parâmetro estejam fora da ordem da matriz retorne true, caso contrário, retorne false.
b. um método “éQuadrada”, que retorna true se a matriz for quadrada (isto é, tem o mesmo número de linhas e colunas).
c. um método total que some todos os valores da matriz retornando o resultado.
d. um método que receba como parâmetro um determinado valor e retorne a primeira linha onde o elemento foi encontrado na matriz ou –1 caso contrário.

**Exercício 7:**

A linguagem Java dispõe de um suporte nativo a vetores, que exige a definição de seu tamanho no momento da instanciação. Depois de instanciado, o tamanho do vetor não pode ser modificado.

Escreva uma classe chamada MeuVetor cujos objetos simulem vetores de tamanho variável. A classe define os seguintes métodos:

**construtor**: recebe como parâmetro o tamanho inicial do vetor.

**insert:** recebe como parâmetro uma string e a coloca na próxima posição disponível do vetor; note que o vetor cresce automaticamente, portanto, se a inserção ultrapassar o tamanho inicial estabelecido na criação, por exemplo, o vetor deve aumentar seu tamanho automaticamente. Crie uma algoritmo para redimensionar um vetor de forma a não perder seus valores.

**get**: recebe como parâmetro uma posição do vetor e retorna a string que estiver naquela posição; se a posição não estiver ocupada ou ultrapassar o tamanho do vetor, este método retorna nulo.

**get:** não recebe parâmetros, retorna o último elemento inserido no vetor, retorna nulo caso o vetor esteja vazio.

**size:** retorna o número de elementos inseridos no vetor (independente do tamanho físico do mesmo)