INE 5401 – Programação Orientada a Objetos
Professor: Isaías Camilo Boratti
Aluno: Lucas Pereira da Silva

Matéria

Trabalho: Escreva uma classe de forma que toda instância dessa classe represente um relógio. Escreva conforme a seguir para listar sua classe. Entrega 19/04 com apenas a classe Relógio.

```
public static void main(String[] p)
{
Interface i = new Interface();
Relogio r = new Relogio();
//Aqui r representa um relógio que marca 0:0:0
Relogio d = new Relogio(11, 59, 59);
String horario = d.fornecaHorario();
//Horário: 11:59:59
i.mostreMensagem(horario);
d.tictac(); //Incrementa horário em segundos
i.mostreMensagem(d.fornecaHorario());
//Será mostrado o string 12:0:0
d.tictac();
//Agora d marca o horario 12:0:1
Relogio rel = new Relogio(23, 59, 59);
rel.tictac();
//Agora rel marca 0:0:0
}
```

Trabalho: Considerando que uma conta bancária é caracterizado pelo seu número, saldo e limite de saque negativo, escreva um programa que para uma conta bancária apresente ao usuário um menu de opções conforme a seguir:

1: Efetuar depósito
2: Efetuar saque
3: Consultar saldo
4: Sair

Entregar em 21/04 todas as classes do programa.

```
public static void main(String[] p)
{
Interface i = newInterface();
int op = 1;
while(op == 1 || op == 2 || op == 3)
{
op = i.mostreMenu();
if(op == 1)
{
double valor = i.pecaValorDepositar();
boolean depositou = c.deposite(valor);
}
```

```
else
if(op == 2)
{
double valor = i.pecaValorSacar();
double saque = i.facaSaque(valor);
}
else //Erro: valores maiors que três serão aceitos
{
i.mostreMensagem("Saldo: "+c.informeSaldo());
}
}
```

Execício: Seja n um número inteiro positivo, escreva uma classe que permita manupular esse n como um objeto. Coloque na classe métodos para:

a) Construtor.
b) Informar se n é ou não um valor par.
c) Fornecer a soma de todos os valores pares maiores que n.
d) Fornecer o fatorial de n.
e) Sortear um valor inteiro >=0 e <n.
f) Fornecer o valor de s dado por: s=1/n+2/n-1+3/n-2+...+n/1.
g) Fornecer a soma dos dígitos que formam n, exemplo 129 = 12.
h) Informar se o valor representado pelo objeto executor é igual, maior ou menor que o valor representado por outro objeto.
i) Fornecer o objeto que representa a soma do valor representado pelo objeto executor com o valor representado por outro objeto.
j) Fornecer o valor de s dado por: s=n-n^2/2!+n^3/3!-n^4/4!+...+
k) Retornar o último digito de n por extenso.
l) Fornecer o valor correspondente a n expresso na base 16.
m) Informar se o valor representado pelo objeto ou é ou não um valor primo.

```
public class IP
{
private int n;
public Ip(int vN)
{
if(vN > 0)
n = vN;
else
n = 1;
}

public boolean ePar()
{
return n % 2  == 0;
}

public int informeN()
{
return n;
}
```

```
public boolean recebaValorN(int vN)
{
if(vN > 0)
{
n = vN;
return true;
}
return false;
}
}

public static void main(String[] p)
Interface i = new Inteface();
Ip a = new Ip(4);
if(a.epar())
{
i,mostreMensagem(“O valor”+a.informeN()+”é par”);
}
else
i.mostreMensagem(“O valor”+a.informeN()+”é impar”);
}
```

Encapsulamento

Encapsular → Consultar funções.

Ao projetar uma classe o projetista deve ocultar aqueles membros da classe (atributos, métodos) que não devem ficar visíveis (acessíveis) a que for usar objetos dessa classe. A classe deve ser escrita de forma que os atributos dos objetos fiquem sempre escondidos, isto é, não visíveis fora da classe.

```
public class Automovel
{
private String cor;
privete String marca;
private int quantPassageiros;
public Automovel()
{
//...
}

public void acelere()
{
//...
}
}
```

Classe

• Componentes declarados com modificador private: Visíveis apenas dentro da classe.

• Componentes declarados com modificador protect: Visíveis dentro da classe, subclasse e demais classes do mesmo pacote.

• Componentes declarados sem modificador: Visíveis dentro da classe e pacote.
• Componentes declarados com modificador public: Visíveis dentro e fora da classe.

Podem ser encapsulados: Atributos e Métodos (componentes da classe).
"Devem" ser encapsulados:
• Todos os atributos
• Aqueles métodos que o projetista julga que não deva disponibilizar

OBS: Métodos em principio devem ser públicos.

Exercício: Escrever uma classe de forma que toda instância dessa classe represente uma pessoa.

```
public class Pessoa
{
protected String nome;
protected int idade;
protected char sexo;

public Pessoa(String vNome, char vSexo, int vIdade)
{
nome = vNome;
vSexo = Character.toUpperCase(vSexo);
if(vSexo == 'M' || vSexo == 'F')
{
sexo = vSexo;
}
else
{
sexo = '*';
}
idade = vIdade;
if(idade < 0)
{
idade = 0;
}
if(idade > 200)
{
idade = 200;
}
}
}
```

Ordem recomendada para os membros de uma classe:
1) Atributos;
2) Construtores;
3) Demais métodos.

Trabalho: Escreva um programa que para um grupo de pessoas determine:
a) Quantidade de mulheres com idade abaixo de 40 anos;
b) Idade média dos adolescentes (12 e 18);
c) Qual a pessoa mais jovem;

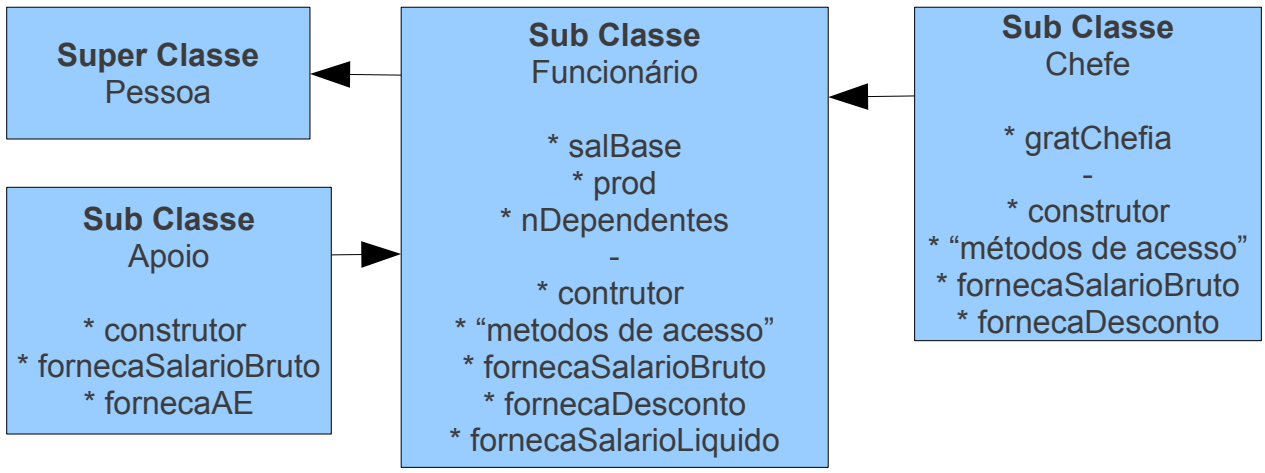
d) Qual a pessoa mais idosa.

Escreva um programa que processe a folha de pagamento dos funcionários de uma empresa. Considere que todo funcionário recebe um salário base e gratificação de produtividade. Para os funcionários que exercem cargos de chefia, a empresa paga uma gratificação adicional. Considere que sobre o salário bruto é feito um desconto para fins de imposto de renda conforme a tabela a seguir:

Estado Bruto	**Aliquota (%)**	**Parcela a deduzir**
Até R$ 1000,00	Isento	-
Acima de R$ 1000,00 até R$ 1800,00	10	R$ 100,00
Acima de R$ 1800,00	25	R$ 370,00

Para os funcionários que apresentam nível primário de escolaridade (funcionários de apoio) a empresa paga auxilio de R$ 60,00 reais por dependente, limitado ao máximo de 5 dependentes. Sobre o auxilio educação não incide nenhum desconto. O programa deve para cada funcionário apresentar o seu demonstrativo salarial, isto é apresentar o nome, salário base, gratificação de produtividade, salário bruto, desconto e salário liquido.

Prova: 10/5



Sobreposição: Mesma assinatura.

```
Public Funcionário pegueFuncionario()
{
–solicitar nome
–solicitar salario base
–solicitar produtividade
–solicitar número de dependentes

char tipo;
–solicitar tipo (F: Funcionário Comum, C: Chefe, A: Apoio)

Se tipo == 'F'
–instanciar e retornar o objeto da classe Funcionário
```

```
Se tipo == 'C'
–solicitar gratificacão de chefia
–instanciar e retornar objeto da classe chefia
...
}
```

Ao derivarmos uma classe, exceto construtores que não são herdados, todos os demais componentes da superclasse são herdados pela subclasse.

Apesar de serem herdados, aqueles componentes que na superclasse são declarados com o modificador private, não estarão visíveis na subclasse. Podem entratento serem acessador por algum método público herdado da superclasse. Veja o exemplo abaixo:

Classe Pessoa

```
private int idade;
public int informeIdade()
{
return idade;
}
```

Classe Funcionário

```
.
.
.
int x = this.informeIdade();
.
.
.
```

Ao derivarmos uma subclasse se algum método herdado da superclasse não servir para a subclasse pois, na subclasse a forma de realização da tarefa implementada pelo método herdado é diferente, este método deve ser reescrito. A isto denomina-se sobreposição.

Para existir a sobreposição é necessário que o método reescrito tenha a mesma assinatura que o respectivo método da superclasse.

Sobreposição: Método herdado é reescrito na subclasse e possui a mesma assinatura.

Sobrecarga: Classe contendo dois ou mais métodos, herdados ou não, com o mesmo nome, mas com assinaturas diferentes,

O programa Empresa deve também: Determinar e apresenta o demonstrativo do funcionário comum de maior salario bruto (caso exista).

Todo construtor de uma subclasse deve chamar o construtor da superclasse. Isto tem por objetivo inicializar os atributos herdados da superclasse e é feito por: `super(lista de argumentos);`

A chamada ao construtor da superclasse deve ser a primeira instrução do construtor da subclasse.

Caso não seja colocada a chamada ao construtor da superclasse, Java coloca automaticamente uma chamada ao construtor padrão: `super();` e se este não existir dará erro.

Polimorfismo

Polimorfismo → várias formas.
Contextos:
• Polimorfismo de método;
• Polimorfismo quanto a forma de instância.
Polimorfismo de Método: É feita a seleção de qual método deve ser executado em funcão da classe da instância executora.

Polimorfismo quanto a forma da instância: Em um hierarquia de classe, um identificador de objeto pode referir objetos com formatos diferentes.

Arranjos

```
public static void main(String[] p)
{
int[] a; //Declarra que a é um arranjo de inteiros.
a =  new int[10];
a[1] = 25;
int k =2;
a[k] = -4;
a[k+1] = 100;
for(int i = 0; i < 10, i++)
{
a[i] = i*i;
}
char[] s = new char[5];
s[1] = 'X';
s[0] = 'a';
s[2] = '* ';
int tam = s.lenght; // Aqui s vale tam = 5.
double[] y = {1.7, -3.1, 9.4, 7.3, 8.0};
tam = y.lenght; //Tam = 5.
Pessoa[] turma = new Pessoa[3];
turma[1] = new Pessoa(“Maria”, 'F', 20);
y[10] = 20.7; //Dará erro.
}
```

Um arranjo é um conjunto de posição onde cada posição armazena um valor (ou um endereço). Todos os valores são do mesmo tipo.
Arranjos são objetos.
OBS: int[]a é diferente de int a e int[] é diferente de int.

Exercício: Escreva um programa que através do método main, faca com que o objeto que representa a interface leia 10 valores inteiros e estes valores devem ficar todos armazenados na memória. Após isto deve ser calculado a média desses valores e também quantos dos valores são maiores que a média.

Trabalho: Escreva um programa que analise o desempenho de um grupo de alunos em determinada disciplina. Considere que na disciplina são realizados “n” testes e uma prova final. Para efeito de determinação da média final, o teste de menor nota é abandonado caso “n” seja maior que 1. Considerando média dos teste com peso de 60%

e prova final com peso de 40% mostre o nome do aluno, média dos testes, nota prova final, se foi aprovado ou não etc.

```
public class Aluno extends Pessoa
{
private double pf;
private double[] int;
private int n;

public Aluno(String vNome, char vSexo, int vI)
{
super(vNome, vSexo, vI);
n = 0;
pf = 0.0;
nt = new double[20];
}

public int informeN()
{
return n;
}

public boolean recebaValorPF(double vPF)
{
if(vPF >= 0 && vPF <= 10)
{
pf = vPF;
return true;
}
return false;
}

public double informePF()
{
return pf;
}

public boolean recebaNotaNovoTeste(double nota)
{
if(nota >= 0 && nota <= 10 && n < 20)
{
nt[n] = nota;
n++;
return true;
}
}

public double informeNotaTeste(int qual)
{
if(qual >= 1 && qual <= n)
return nt[qual-1];
return -1.0;
```

```
}

public double fornecaMenorNotaTeste()
{
double menor = nt[0];
for(int i = 1; i < n; i++)
if(nt[i] < menor)
menor = nt[i];
return menor;
}

public double fornecaMediaTestes()
{
double soma = 0;
for(int i = 0; i < n; i++)
soma += nt[i];
if(n == 1)
return soma;
return (soma-this.fornecaMenorNotaTeste())/(n-1);
}

public double fornecaMediaFinal()
{
return this.fornecaMediaTestes()*0.6+pf*0.4;
}

public boolean foiAprovado(double criterio)
{
return this.fornecaMediaFinal() >= criterio;
}
}

public Aluno pegueAluno()
{
–solicita nome; → String nome = this.pegueNome();
–solicita sexo; → char sexo = this.pegueSexo();
–solicita idade;
Aluno a = new Aluno(nome, sexo, idade)
–solicitar notaPF;
a.recebaValorNotaPF(notaPF);
–solicitar quantidade de testes (qt);
for(int i =1; i <= gt; i++)
{
–solicitar nota teste i (nota);
a.recebaValorNotaNovoTeste (nota);
}
return a;
}
```

O programa também deve:
a) Mostrar o melhor aluno.
b) Determinar quantos alunos tiveram média final maior que a média da turma.

Exercício: Escreva um método main que:

a) Sorteie 20 valores inteiros, todos na faixa 50 a 100, inclusive extremos e armazene-os todos na memória.

b) Após resolvido o item a, mostre todos os valores sorteados.

c) Crie na memória um arranjo contendo aqueles valores sorteados, que são pares e após mostre esses valores.

```
public class Principal
{
public static void main(String[] parametro)
{
Interface comunicaUsuario = new Interface();
int[] sorteados = new int[20];
for(int cont = 0; cont < 20; cont++)
{
sorteados[cont] = (int)(Math.random()*51)+50;
}
comunicaUsuario.mostreArranjo(random);
Principal pares = new Principal();
int[] sorteadosPares = pares.fornecaPares(sorteados);
comunicaUsuario.mostreSorteados(sorteadosPares);
}

public int[] fornecaPares(int[] sort)
{
int k = 0;
int tam = sort.lenght;
int contPar = 0;
for(int i = 0; i < tam; i++)
if(sort[i]%2 == 0)
contPar++;
int[] pares = new int[contPar];
for(int i = 0; i < tam; i++)
if(sort[i]%2 == 0)
{
pares[k++] = sort[i];
}
}
return pares;
}
```

Trabalho: Escreva um programa que analise os apostadores da mega sena. O programa deve efetuar o sorteio da mega sena (6 números inteiros, todos diferentes e todos no intervalo de 1 a 60, incluindo os extremos). Os números sorteados devem ser mostrados ao usuário. Cada apostador pode apostar de 6 até 10 números, todos diferentes e todos no intervalo de 1 a 60. Considere que se apostar 6 números o apostador pagará R$ 1 , se apostar 7 números pagará R$ 7, se apostar 8 números pagará R$ 28, se apostar 9 números pagará R$ 168 e se apostar 10 números pagará R$ 1260. O programa deve mostrar o nome e o total de pontos de cada apostador. Deve também mostrar o total pago pelos apostadores. OBS: trate o apostador como um objeto. Lembrando que todo apostador é uma pessoa.

Exercício: Escrever um classe de forma que toda instância dessa classe represente uma coleção de valores inteiros.

```
public class Ctém um ColecaoInteiros
{
private int n;
private int[] x;
pulibc ColecaoInteiros(int tam)
{
/*Contém um objeto que representa uma colecão vazia mas com capacidade para armazenar tam valores*/
n = 0;
x = new int[tam];
}

public ColecaoInteiros(int[] valores)
{
n = valores.lenght;
x = new int[n];
for(int i = 0; n < tam; i++)
{
x[i] = valores[i];
}
}

public int informeN()
{
return n;
}

public String toString()
{
String s = “Valores da Colecão \n”;
for(int i = 0; i < n; i++)
{
s = s+x[i]+” ”;
}
return s;
}
}
```

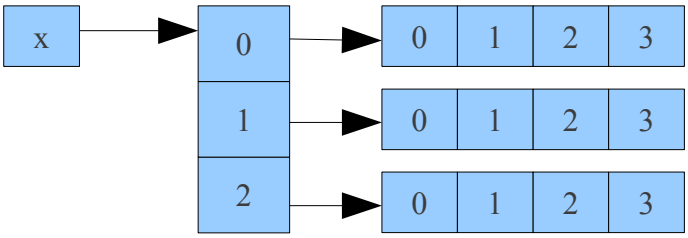
- Método que retorna com a posição da primeira ocorrências de determinado valor.
- Método que retorna com o valor de determinada posição.
- Método que retorna com um arranjo contendo os valores pares da seleção.
- Método que elimina da coleção a primeira ocorrência de um valor.
- Método que retorna um objeto da classe ColecaoInteiros que represente a união da coleção representada pelo objeto executor com a coleção representada por outro objeto.
- Método que inverta a seleção.

Prova 31/05

Arranjos Multidimensionais

Podemos imaginar arranjos multidimensionais como uma matriz.

```
int[][] x = new int[3][4];
```



Métodos Estáticos

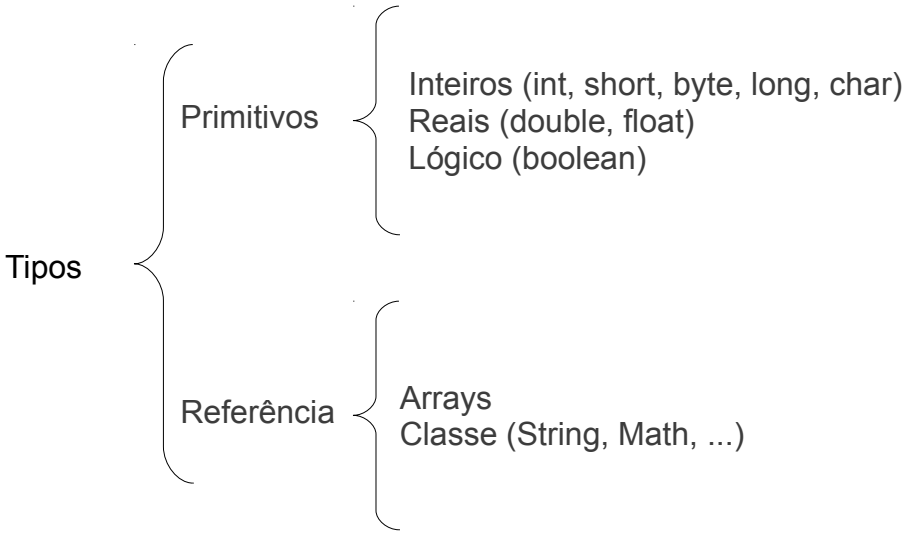
```
public class Principal
{
public static void main(String[] parametro)
{
int[][] m = Interface.pegueMatriz();
Interface.mostreMatriz(m);
int soma = Matriz.fornecaSoma(m);
int somaColuna3 = Matriz.fornecaSomaColuna(m,3);
int max = Matriz.fornecaMaiorValor(m);
}
}

Public class Matriz
{
public static int fornecaSoma(int[][] mat)
{
int soma = 0;
int nL = mat.length;
int nC = mat[0].length;
for(int lin = 0; lin < nL; lin++)
{
for(int col = 0; col < nC; col++)
{
soma += mat[lin][col];
}
}
return soma;
}
}
```

Inicializar a matriz com determinados valores:

```
int[] a = { 4, 7, -3, 2};
int[][] b = {{1, 4, 7}, {1, 9, 10}, {-2, 4, 7}};
```

- Criar um método que, para uma matriz de elementos inteiros retorne com a sua transposta.
- Criar um método estático que para uma matriz, retorne com a soma dos elementos da diagonal principal.

Exercício: Escreva um programa que sorteie n números inteiros, todos na faixa 0 a 20. O programa deve determinar a frequência de cada um dos valores de 0 a 20.



Manipulação de Strings

- String é uma sequência de zero ou mais caracteres.
- Strings são objetos da classe string.
- Linguagem oferece facilidade de forma a manipular um string com um tipo primitivo.
- Strings são imutáveis.

Classe String

Método int length()

```
String s = “UFSC”;
int tam = s.length;
//tam = 4
```

Método char charAt(int)

```
String s = “Universidade”;
char letra  = s.charAt(3);
//tam = v
```

Método boolean equals(String)

```
String s1 = “Curso de POO”;
String s2 = “Curso de Java”;
boolean b = s1.equals(s2);
//b = false;
```

Método int compareTo(String)

```
int x = s1.compareTo(s2);
//x = 0 se s1 = s2;
//x < 0 se s1 < s2;
//x > 0 se s1 > s2;
```

/*O compareTo se baseia na ordem alfabética, ou seja o maior string é aquele que vem depois na ordem alfabética.*/

Método String substring(int inicio, int fim)

```
//Copia e retorna um substring do objeto executor do método. A cópia começa na posição inicio e termina na posição fim-1.
String s1 = “Universidade Federal”;
String s2 = s1.substring(7, 12);
//s2 = idade;
```

Método String toUpperCase()

```
String s1 = “Exemplo 1”;
String s2 = s1.toUpperCase();
//s2 = “EXEMPLO 1”;
```

Método String trim()

```
//Retorna um string sem os espacos em branco no inicio e no fim
String s1 = “       UFS C     “;
String s2 = “s1.trim();
//s2 = “UFS C”;
```

Exercício: Escreva uma classe contendo métodos que manipulam strings:

1) Método que informe se um string contém ou não determinado caractere.

2) Método que retorna com a posição da última ocorrência de um determinado caractere em um string.

3) Método que retorna a quantidade de letras de um string.

4) Método que informa se um caractere é ou não um vogal.

5) Método que para um arranjo de strings retorne com o maior string (ordem alfabética).

6) Método que retorna com um string de tamanho n formado pela concatenação de letras minúsculas sorteadas.

7) Método que informa se um string representa ou não um valor inteiro.

8) Método que informa se um string é ou não um palíndromo.

9) Método que informa a quantidade de palavras de uma frase. Palavra é uma sequência continua de letras.

10) Método que coloque em ordem alfabética um arranjo de strings.

Exercício: Uma revendedora de automóveis tem para cado que está a venda o seu ano, marca e preço de venda. Escreva um programa que:

a) Armazene na memória todos os dados.

b) Determine e mostre o caro mais caro da revendedora.

c) Mostre a quantidade de carro que cada marca tem.

Prova Final: 30/06 – Sala EEL008

Exercício: Escreva um programa que controle o processo de atendimento das pessoas que chegam a uma determinada repartição. O programa deve colocar ao usuário

o seguinte menu de opções:

1. Colocar pessoa na fila.
2. Atender pessoa.
3. Verificar se pessoa está na fila.
4. Listar pessoas da fila.
5. Finalizar programa.

Colocar as seguintes opções:

a) Mostrar a idade média das pessoas da fila.

b) Mostrar nome de pessoa mais idosa da fila.