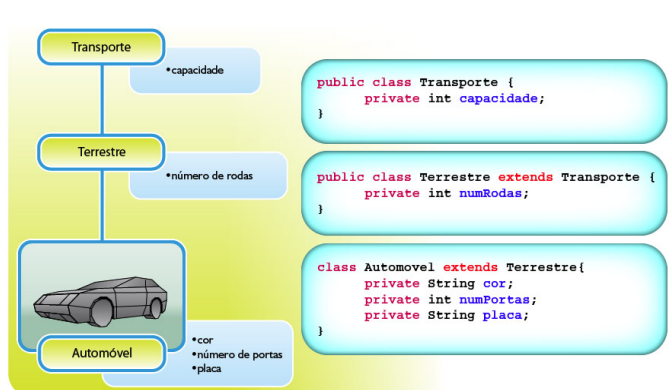
 UEPB	UNIVERSIDADE ESTADUAL DA PARAÍBA CENTRO DE CIÊNCIAS E TECNOLOGIA CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO	
ATIVIDADE DE ESTUDO X		
CURSO	Ciência da Computação	
DISCIPLINA	Linguagem de Programação II	
PROFESSOR		TITULAÇÃO
Jucelio Soares dos Santos		Mestrado
NOME	Lucas de Lucena Siqueira	
MATRÍCULA	201080354	CONCEITO
DATA		

### DESCRIÇÃO DA ATIVIDADE

1. O que é herança? E qual a diferença entre Herança simples e Herança múltipla?

R/ A herança é um mecanismo que permite o programador criar classes a partir de outras classes já existentes, aproveitando as características da classe que está sendo estendida. Quando uma classe herda características apenas de uma outra classe, trata-se de uma herança simples, mas quando uma classe está herdando duas ou mais classes, há uma herança múltipla.

2. Considerando o código das classes Transporte, Terrestre e Automóvel apresentados na Figura abaixo, finalize a implementação delas, adicionando os métodos get() e set() para cada um de seus atributos. Em seguida, crie uma classe Principal com um método main() que cria um objeto da classe Automóvel, e chama todos os métodos set() e get() criados, inclusive das classes Transporte e Terrestre. Observe no seu exemplo, que é possível chamar todos os métodos get() e set() herdados pela classe Automóvel.



## Classe Transporte

```
1 package Atividade10.entities;
2
3 public class Transporte {
4     private int capacidade;
5
6     public Transporte(int capacidade) {
7         this.capacidade = capacidade;
8     }
9
10    public Transporte() {
11
12    }
13
14    public int getCapacidade() {
15        return capacidade;
16    }
17
18    public void setCapacidade(int capacidade) {
19        this.capacidade = capacidade;
20    }
21 }
22
```

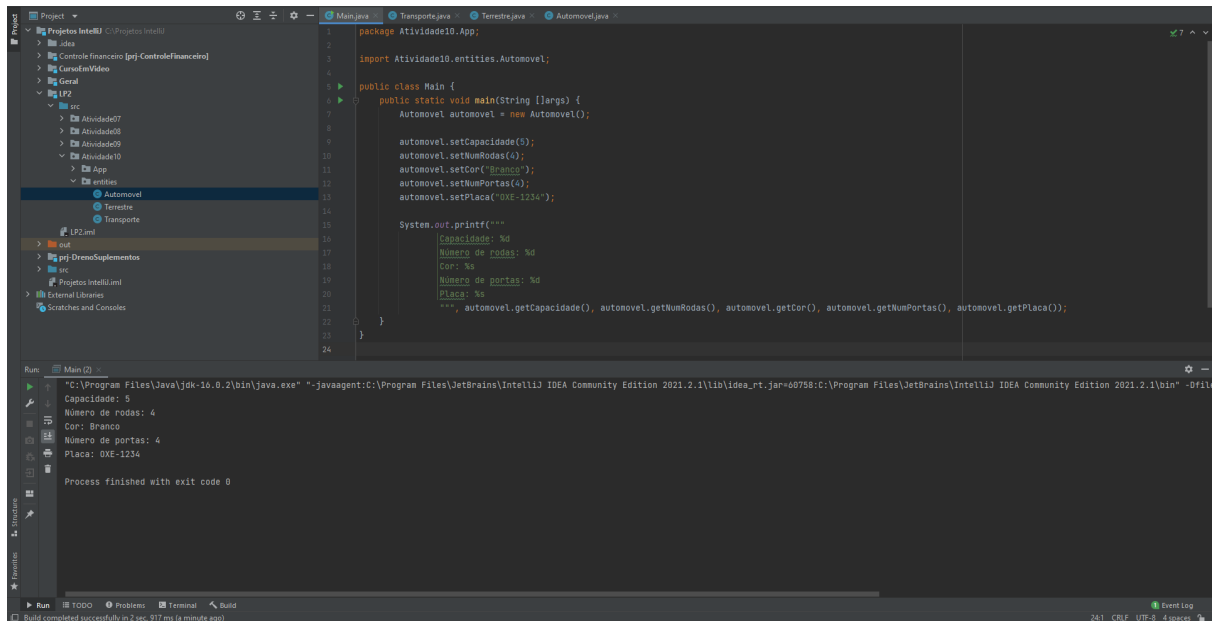
## Classe Terrestre

```
1 package Atividade10.entities;
2
3 public class Terrestre extends Transporte {
4     private int numRodas;
5
6     public Terrestre(int capacidade, int numRodas) {
7         super(capacidade);
8         this.numRodas = numRodas;
9     }
10
11    public Terrestre() {
12
13    }
14
15    public Terrestre(int numRodas) {
16        this.numRodas = numRodas;
17    }
18
19    public int getNumRodas() {
20        return numRodas;
21    }
22
23    public void setNumRodas(int numRodas) {
24        this.numRodas = numRodas;
25    }
26 }
27
```

## Classe Automovel

```
1 package Atividade10.entities;
2
3 public class Automovel extends Terrestre {
4     private String cor;
5     private int numPortas;
6     private String placa;
7
8     public Automovel(int capacidade, int numRodas, String cor, int numPortas, String placa) {
9         super(capacidade, numRodas);
10        this.cor = cor;
11        this.numPortas = numPortas;
12        this.placa = placa;
13    }
14
15    public Automovel() {
16
17    }
18
19    public Automovel(String cor, int numPortas, String placa) {
20        this.cor = cor;
21        this.numPortas = numPortas;
22        this.placa = placa;
23    }
24
25    public Automovel(int numRodas, String cor, int numPortas, String placa) {
26        super(numRodas);
27        this.cor = cor;
28        this.numPortas = numPortas;
29        this.placa = placa;
30    }
31
32    public String getCor() {
33        return cor;
34    }
35
36    public void setCor(String cor) {
37        this.cor = cor;
38    }
39
40    public int getNumPortas() {
41        return numPortas;
42    }
43
44    public void setNumPortas(int numPortas) {
45        this.numPortas = numPortas;
46    }
47
48    public String getPlaca() {
49        return placa;
50    }
51
52    public void setPlaca(String placa) {
53        this.placa = placa;
54    }
55 }
```

## Main



```
package Atividade10.App;

import Atividade10.entities.Automovel;

public class Main {
    public static void main(String[] args) {
        Automovel automovel = new Automovel();

        automovel.setCapacidade(5);
        automovel.setNumRodas(4);
        automovel.setCor("Branco");
        automovel.setNumPortas(4);
        automovel.setPlaca("0XE-1234");

        System.out.printf("Capacidade: %d\n", automovel.getCapacidade());
        System.out.printf("Numero de rodas: %d\n", automovel.getNumRodas());
        System.out.printf("Cor: %s\n", automovel.getCor());
        System.out.printf("Numero de portas: %d\n", automovel.getNumPortas());
        System.out.printf("Placa: %s\n", automovel.getPlaca());
    }
}
```

Run: Main (2)

```
"C:\Program Files\Java\jdk-10.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.1\lib\idea_rt.jar=60758:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.1\bin" -Dfile.encoding=UTF-8
Capacidade: 5
Numero de rodas: 4
Cor: Branco
Numero de portas: 4
Placa: 0XE-1234

Process finished with exit code 0
```

3. Hoje, os celulares estão cada vez mais sofisticados, mp3, mp4, acesso web, câmera digital... Crie uma hierarquia de classes que possui no topo da hierarquia o celular mais básico, aquele que simplesmente liga e atende ligações, e que vai sendo refinada com várias outras classes que definem celulares mais sofisticados e modernos. Para isso, use a hierarquia e nos construtores das classes use o operador super. Na hierarquia, defina no mínimo três classes. Em seguida, acrescente agora os métodos que darão as funcionalidades dos telefones móveis (celulares). Uma observação é que os métodos ligar() e atender() não precisam ser criados novamente nas classes filhas. Uma vez que ele deve estar presente na classe Mãe da hierarquia (celular mais simples).

## Classe Iphone5

```
1 package Atividade10.entities;
2
3 public class Iphone5 {
4     private String numero;
5     private boolean estaLigando;
6     private boolean estaAtendendo;
7
8     public Iphone5(String numero, boolean estaLigando, boolean estaAtendendo) {
9         this.numero = numero;
10        this.estaLigando = estaLigando;
11        this.estaAtendendo = estaAtendendo;
12    }
13
14    public void iniciarLigacao() {
15        this.estaLigando = true;
16    }
17
18    public void finalizarLigacao() {
19        this.estaLigando = false;
20    }
21
22    public void atenderLigacao() {
23        this.estaAtendendo = true;
24    }
25
26    public String getNumero() {
27        return numero;
28    }
29
30    public void setNumero(String numero) {
31        this.numero = numero;
32    }
33
34    public boolean isEstaLigando() {
35        return estaLigando;
36    }
37
38    public void setEstaLigando(boolean estaLigando) {
39        this.estaLigando = estaLigando;
40    }
41
42    public boolean isEstaAtendendo() {
43        return estaAtendendo;
44    }
45
46    public void setEstaAtendendo(boolean estaAtendendo) {
47        this.estaAtendendo = estaAtendendo;
48    }
49 }
```

## Classe Iphone6

```
1 package Atividade10.entities;
2
3 public class Iphone6 extends Iphone5{
4     private boolean estaFotografando;
5     private boolean estaTocandoMusica;
6
7     public Iphone6(String numero, boolean estaLigando, boolean estaAtendendo, boolean estaFotografando, boolean estaToca
8         super(numero, estaLigando, estaAtendendo);
9         this.estaFotografando = estaFotografando;
10        this.estaTocandoMusica = estaTocandoMusica;
11    }
12
13    public void fotografar() {
14        this.estaFotografando = true;
15    }
16
17    public void pararFotografar() {
18        this.estaFotografando = false;
19    }
20
21    public void tocarMusica() {
22        this.estaTocandoMusica = true;
23    }
24
25    public void pararMusica() {
26        this.estaTocandoMusica = false;
27    }
28
29    public boolean isEstaFotografando() {
30        return estaFotografando;
31    }
32
33    public void setEstaFotografando(boolean estaFotografando) {
34        this.estaFotografando = estaFotografando;
35    }
36
37    public boolean isEstaTocandoMusica() {
38        return estaTocandoMusica;
39    }
40
41    public void setEstaTocandoMusica(boolean estaTocandoMusica) {
42
43        this.estaTocandoMusica = estaTocandoMusica;
44    }
45 }
```

## Classe Iphone7

```
1 package Atividade10.entities;
2
3 public class Iphone7 extends Iphone6{
4     private boolean isCpuBoostOn;
5     private boolean isAiOn; //AI -> Artificial Intelligence
6
7     public Iphone7(String numero, boolean estaLigando, boolean estaAtendendo, boolean estaFotografando, boolean estaTocandoMusica, boolean isCpuBoostOn, boolean isAiOn) {
8         super(numero, estaLigando, estaAtendendo, estaFotografando, estaTocandoMusica);
9         this.isCpuBoostOn = isCpuBoostOn;
10        this.isAiOn = isAiOn;
11    }
12
13    public void ligarCpuBoost() {
14        this.isCpuBoostOn = true;
15    }
16
17    public void desligarCpuBoost() {
18        this.isCpuBoostOn = false;
19    }
20
21    public void ligarAi() {
22        this.isAiOn = true;
23    }
24
25    public void desligarAi() {
26        this.isAiOn = false;
27    }
28
29    public boolean isCpuBoostOn() {
30        return isCpuBoostOn;
31    }
32
33    public void setCpuBoostOn(boolean cpuBoostOn) {
34        isCpuBoostOn = cpuBoostOn;
35    }
36
37    public boolean isAiOn() {
38        return isAiOn;
39    }
40
41    public void setAiOn(boolean aiOn) {
42        isAiOn = aiOn;
43    }
44 }
```

### 4. Responda:

- Para que serve o modificador de acesso protected? Como ele funciona no caso de herança entre classes?

R/ O modificador protected só permite que o membro declarado com essa modificação seja acessível em classes que estejam em um mesmo pacote ou que estejam herdando a classe que contém os membros declarados como protected. No caso de herança entre classes, o modificador permitirá o acesso dos membros normalmente.

- Para que serve a palavra-chave super? Dê um exemplo concreto do seu funcionamento.

R/ Utilizar o super() em um construtor de uma classe filha, o método irá invocar o construtor da classe mãe.

Exemplo:

## Classe Mãe

```
1 package entities.pessoa;
2
3 public class Pessoa {
4     private String nome;
5     private Integer idade;
6     private String cpf;
7     private String sexo;
8     private String telefone;
9     private String email;
10    private String endereco;
11
12    public Pessoa() {
13
14    }
15
16    public Pessoa(String nome, Integer idade, String cpf, String sexo, String telefone, String email, String endereco) {
17        this.nome = nome;
18        this.idade = idade;
19        this.cpf = cpf;
20        this.sexo = sexo;
21        this.telefone = telefone;
22        this.email = email;
23        this.endereco = endereco;
24    }
25
26    public String getNome() { return nome; }
27
28    public void setNome(String nome) { this.nome = nome; }
29
30    public Integer getIdade() { return idade; }
31
32    public void setIdade(Integer idade) { this.idade = idade; }
33
34    public String getCpf() { return cpf; }
35
36    public void setCpf(String cpf) { this.cpf = cpf; }
37
38    public String getSexo() { return sexo; }
39
40    public void setSexo(String sexo) { this.sexo = sexo; }
41
42    public String getTelefone() { return telefone; }
43
44    public void setTelefone(String telefone) { this.telefone = telefone; }
45
46    public String getEmail() { return email; }
47
48    public void setEmail(String email) { this.email = email; }
49
50    public String getEndereco() { return endereco; }
51
52    public void setEndereco(String endereco) { this.endereco = endereco; }
53
54    @Override
55    public String toString() {
56        return "Pessoa{" +
57            "nome=" + nome + '\n' +
58            "idade=" + idade +
59            "cpf=" + cpf + '\n' +
60            "sexo=" + sexo + '\n' +
61            "telefone=" + telefone + '\n' +
62            "email=" + email + '\n' +
63            "endereco=" + endereco + '\n' +
64            '}';
65    }
66 }
```

```
58    public String getTelefone() { return telefone; }
59
60    public void setTelefone(String telefone) { this.telefone = telefone; }
61
62    public String getEmail() { return email; }
63
64    public void setEmail(String email) { this.email = email; }
65
66    public String getEndereco() { return endereco; }
67
68    public void setEndereco(String endereco) { this.endereco = endereco; }
69
70    @Override
71    public String toString() {
72        return "Pessoa{" +
73            "nome=" + nome + '\n' +
74            "idade=" + idade +
75            "cpf=" + cpf + '\n' +
76            "sexo=" + sexo + '\n' +
77            "telefone=" + telefone + '\n' +
78            "email=" + email + '\n' +
79            "endereco=" + endereco + '\n' +
80            '}';
81    }
82 }
```



## Classe Filha (com chamada do super())

```
1 package entities.pessoa;
2
3 public class Cliente extends Pessoa {
4     private Integer idCliente;
5
6     public Cliente(Integer idCliente) {
7         this.idCliente = idCliente;
8     }
9
10    public Cliente(String nome, Integer idade, String cpf, String sexo, String telefone, String email, String endereco, Integer idCliente) {
11        super(nome, idade, cpf, sexo, telefone, email, endereco);
12        this.idCliente = idCliente;
13    }
14
15    public Integer getIdCliente() {
16        return idCliente;
17    }
18
19    public void setIdCliente(Integer idCliente) {
20        this.idCliente = idCliente;
21    }
22 }
23
24
```

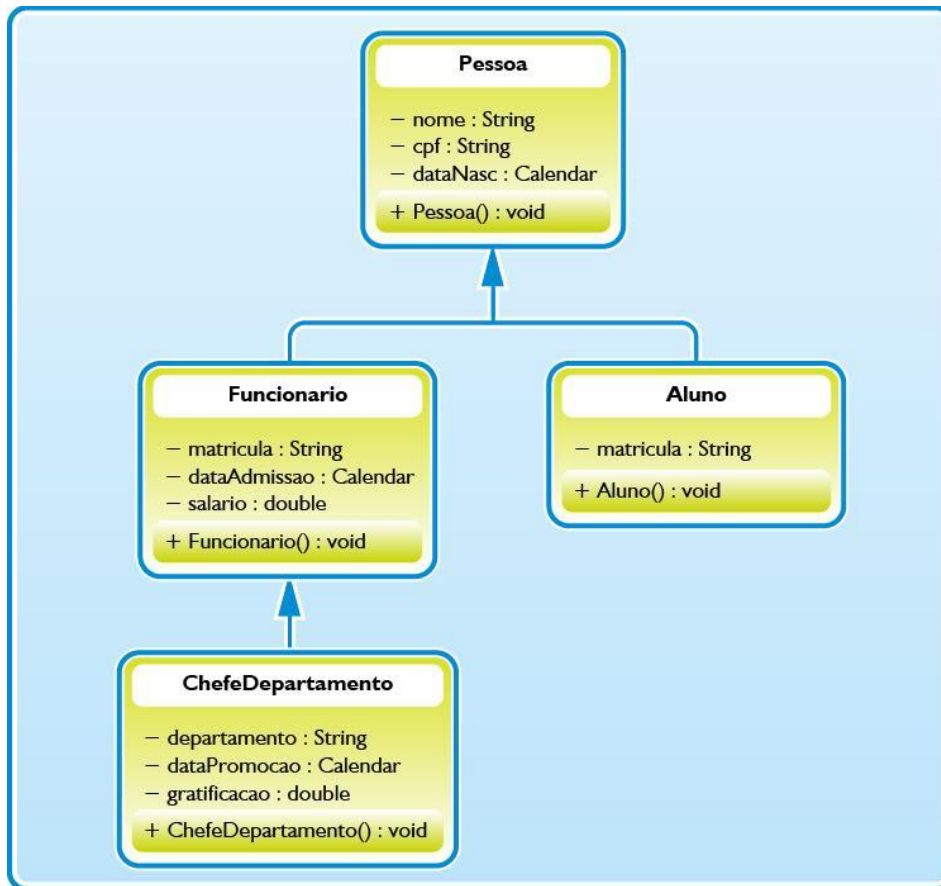
- c. Existe alguma diferença entre o funcionamento da herança para os atributos e métodos?

R/ A herança irá permitir que a classe filha tenha acesso aos métodos e atributos públicos ou protegidos da superclasse, contudo os métodos e atributos privados não poderão ser herdados e não serão acessíveis com a palavra chave super.

- d. Descreva o que acontece com o acesso aos atributos e métodos quando são do tipo: public, private e protected

R/ Para a declaração como public ou protected, a herança irá permitir o acesso de toda a classe mãe, contudo a declaração como private o mesmo não acontecerá.

5. Crie as classes utilizando o princípio da herança, obedecendo à hierarquia da figura abaixo (obs.: para facilitar, substitua na figura o tipo Calendar por String).



- a. Acrescente aos construtores a lista de parâmetros necessária para instanciar o objeto. Por exemplo, a classe Pessoa deve ter nome, CPF e dataNasc. E essa lista é acumulativa, ou seja, o construtor da classe Funcionário deve ter a lista de seus atributos mais os atributos necessários para a classe Pessoa. Dica: não deixe de usar a palavra-chave super em cada um dos construtores para chamar o construtor da classe mãe, passando os atributos que são mantidos por ela e seus ancestrais.

## Classe pessoa:

```
1 package Atividade10.entities;
2
3 public class Pessoa {
4     private String nome;
5     private String cpf;
6     private String dataNasc;
7
8     public Pessoa(String nome, String cpf, String dataNasc) {
9         this.nome = nome;
10        this.cpf = cpf;
11        this.dataNasc = dataNasc;
12    }
13
14    public String getNome() {
15        return nome;
16    }
17
18    public void setNome(String nome) {
19        this.nome = nome;
20    }
21
22    public String getCpf() {
23        return cpf;
24    }
25
26    public void setCpf(String cpf) {
27        this.cpf = cpf;
28    }
29
30    public String getDataNasc() {
31        return dataNasc;
32    }
33
34    public void setDataNasc(String dataNasc) {
35        this.dataNasc = dataNasc;
36    }
37 }
```

## Classe Funcionario

```
1 package Atividade10.entities;
2
3 public class Funcionario extends Pessoa{
4     private String matricula;
5     private String dataAdmissao;
6     private Double salario;
7
8     public Funcionario(String nome, String cpf, String dataNasc, String matricula, String dataAdmissao, Double salario) {
9         super(nome, cpf, dataNasc);
10        this.matricula = matricula;
11        this.dataAdmissao = dataAdmissao;
12        this.salario = salario;
13    }
14
15    public String getMatricula() {
16        return matricula;
17    }
18
19    public void setMatricula(String matricula) {
20        this.matricula = matricula;
21    }
22
23    public String getDataAdmissao() {
24        return dataAdmissao;
25    }
26
27    public void setDataAdmissao(String dataAdmissao) {
28        this.dataAdmissao = dataAdmissao;
29    }
30
31    public Double getSalario() {
32        return salario;
33    }
34
35    public void setSalario(Double salario) {
36        this.salario = salario;
37    }
38 }
```

## Classe ChefeDepartamento

```
1 package Atividade10.entities;
2
3 public class ChefeDepartamento extends Funcionario{
4     private String departamento;
5     private String dataPromocao;
6     private Double gratificacao;
7
8     public ChefeDepartamento(String nome, String cpf, String dataNasc, String matricula, String dataAdmissao, Double salario, String departamento, String dataPromocao, Double gratificacao) {
9         super(nome, cpf, dataNasc, matricula, dataAdmissao, salario);
10        this.departamento = departamento;
11        this.dataPromocao = dataPromocao;
12        this.gratificacao = gratificacao;
13    }
14
15    public String getDepartamento() {
16        return departamento;
17    }
18
19    public void setDepartamento(String departamento) {
20        this.departamento = departamento;
21    }
22
23    public String getDataPromocao() {
24        return dataPromocao;
25    }
26
27    public void setDataPromocao(String dataPromocao) {
28        this.dataPromocao = dataPromocao;
29    }
30
31    public Double getGratificacao() {
32        return gratificacao;
33    }
34
35    public void setGratificacao(Double gratificacao) {
36        this.gratificacao = gratificacao;
37    }
38 }
```

## Classe Aluno

```
1 package Atividade07.Entidades;
2
3 public class Aluno extends Pessoa{
4     private String matricula;
5
6     public Aluno(String nome, String cpf, String dataNasc, String matricula) {
7         super(nome, cpf, dataNasc);
8         this.matricula = matricula;
9     }
10
11    public String getMatricula() {
12        return matricula;
13    }
14
15    public void setMatricula(String matricula) {
16        this.matricula = matricula;
17    }
18 }
```

- b. Insira os seguintes métodos para apresentar os valores dos atributos das classes, `mostrarPessoa()`, `mostrarFuncionario()`, `mostrarChefe()` e `mostrarAluno()`, respectivamente, às classes `Pessoa`, `Funcionário`, `ChefeDepartamento` e `Aluno`. Para imprimir os atributos, use o método `System.out.println()` em cada um dos métodos.

## mostrarPessoa()

```
38 public void mostrarPessoa() {
39     System.out.println("Pessoa:" +
40         "\nnome = " + nome +
41         "\ncpf = " + cpf +
42         "\ndataNasc = " + dataNasc);
43 }
44 }
```

```
1 package Atividade10.App;
2
3 import Atividade10.entities.Automovel;
4 import Atividade10.entities.Pessoa;
5
6 public class Main {
7     public static void main(String []args) {
8         Pessoa pessoa = new Pessoa( nome: "Lucas", cpf: "123", dataNasc: "19/11/01");
9         pessoa.mostrarPessoa();
10    }
11 }
```

```
Run: Main (2) x
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program Files\J
Pessoa:
nome = Lucas
cpf = 123
dataNasc = 19/11/01

Process finished with exit code 0
```

## mostrarFuncionario()

```
39 public void mostrarFuncionario() {
40     System.out.println("Funcionario:" +
41         "\nmatricula = " + matricula +
42         "\ndataAdmissao = " + dataAdmissao +
43         "\nsalario = " + salario);
44 }
45 }
```

```
1 package Atividade10.App;
2
3 import Atividade10.entities.Automovel;
4 import Atividade10.entities.Funcionario;
5 import Atividade10.entities.Pessoa;
6
7 public class Main {
8     public static void main(String []args) {
9         Funcionario funcionario = new Funcionario( nome: "Lucas", cpf: "123", dataNasc: "19/11/01", matricula: "00001", dataAdmissao: "30/09/21", salario: 99999.9);
10        funcionario.mostrarFuncionario();
11    }
12 }
13 }
```

```
Run: Main (2) x
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program
Funcionario:
matricula = 00001
dataAdmissao = 30/09/21
salario = 99999.9
Process finished with exit code 0
```

**mostrarChefe()**

```
39 public void mostrarChefe() {
40     System.out.println("Chefe:" +
41         "\ndepartamento = " + departamento +
42         "\ncdataPromocao = " + dataPromocao +
43         "\ngratificacao = " + gratificacao);
44 }
45 }
```

```
1 package Atividade10.App;
2
3 import Atividade10.entities.Automovel;
4 import Atividade10.entities.ChefeDepartamento;
5 import Atividade10.entities.Funcionario;
6 import Atividade10.entities.Pessoa;
7
8 public class Main {
9     public static void main(String []args) {
10         ChefeDepartamento chefeDepartamento = new ChefeDepartamento( nome: "Lucas", cpf: "123", dataNasc: "19/11/01",
11             matricula: "0001", dataAdmissao: "30/09/21", salario: 999999.9, departamento: "Financeiro", dataPromocao: "29/09/21",
12             gratificacao: 500.50);
13         chefeDepartamento.mostrarChefe();
14     }
15 }
```

```
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe
Chefe:
departamento = Financeiro
cdataPromocao = 29/09/21
gratificacao = 500.5
```

## mostrarAluno()

```
19 public void mostrarAluno() {  
20     System.out.println("Aluno:" +  
21         "\nmatricula = " + matricula);  
22 }
```

```
1 package Atividade10.App;  
2  
3 import Atividade10.entities.*;  
4  
5 public class Main {  
6     public static void main(String []args) {  
7         Aluno aluno = new Aluno( nome: "Lucas", cpf: "123", dataNasci: "19/11/01", matricula: "00002");  
8         aluno.mostrarAluno();  
9     }  
10 }
```

```
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:0  
Aluno:  
matricula = 00002  
  
Process finished with exit code 0
```

- c. Crie uma classe TestaTudo com um método main(), que instancia um objeto de cada uma das classe e exibe os valores dos atributos através de chamadas aos métodos mostrarPessoa(), mostrarFuncionario(), mostrarChefe() e mostrarAluno().

```

1 package Atividade10.App;
2
3 import Atividade10.entities.Aluno;
4 import Atividade10.entities.ChefeDepartamento;
5 import Atividade10.entities.Funcionario;
6 import Atividade10.entities.Pessoa;
7
8 public class TestaTudo {
9     public static void main(String[] args) {
10         Pessoa pessoa = new Pessoa( nome: "Lucas", cpf: "123", dataNasc: "19/11/01");
11         Funcionario funcionario = new Funcionario( nome: "Lucas", cpf: "123", dataNasc: "19/11/01", matricula: "00001", dataAdmissao: "30/09/21", salario: 99999.9);
12         Aluno aluno = new Aluno( nome: "Lucas", cpf: "123", dataNasc: "19/11/01", matricula: "00002");
13         ChefeDepartamento chefeDepartamento = new ChefeDepartamento( nome: "Lucas", cpf: "123", dataNasc: "19/11/01",
14             matricula: "0001", dataAdmissao: "30/09/21", salario: 999999.9, departamento: "Financeiro", dataPromocao: "29/09/21",
15             gratificacao: 500.50);
16         pessoa.mostrarPessoa();
17         funcionario.mostrarFuncionario();
18         chefeDepartamento.mostrarChefe();
19         aluno.mostrarAluno();
20     }
21 }

```

```

"C:\Program Files\Java\jdk-16.0.2\bin
Pessoa:
nome = Lucas
cpf = 123
dataNasc = 19/11/01
Funcionario:
matricula = 00001
dataAdmissao = 30/09/21
salario = 99999.9
Chefe:
departamento = Financeiro
cdataPromocao = 29/09/21
gratificacao = 500.5
Aluno:
matricula = 00002
|
Process finished with exit code 0

```

## 6. O que é polimorfismo?

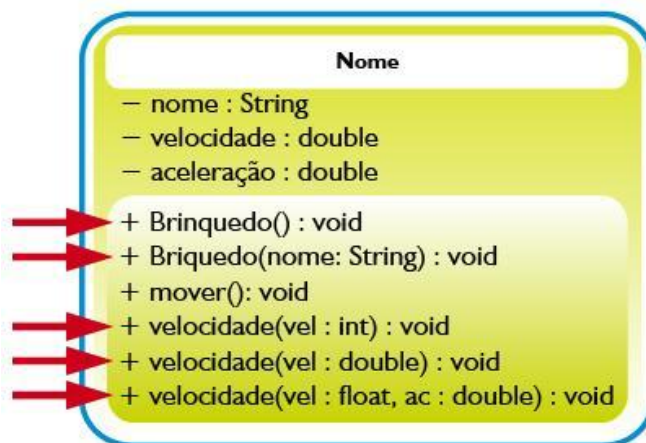
R/ Poli = Muitas, Morfismo = Formas, logo no java, o polimorfismo é aplicado nos métodos. O polimorfismo é uma forma de selecionar funcionalidades que serão utilizadas de forma dinâmica através do princípio que parte da derivação de classes de uma classe base. Essas classes derivadas poderão invocar métodos que por mais que apresentem a mesma assinatura, terão comportamentos diferentes para cada uma das classes derivadas criadas.



7. Quais são os tipos de polimorfismo? Como eles funcionam?

Os 3 principais tipos de polimorfismo são o de sobreposição (Override), que permite o programador fazer a reescrita do método, tornando possível reescrever nas classes filhas os métodos criados na classe mãe, adicionando algo a mais ou não, porém sempre mantendo a mesma assinatura. O de sobrecarga (Overload) irá permitir que existam métodos com mesmo nome, mas com a assinatura levemente alterada, podendo variar o número e o tipo dos parâmetros. Já o de inclusão é presente quando temos um método que irá sobrescrever um outro método que foi herdado da classe mãe.

8. Implemente em Java a classe Brinquedo apresentada na Figura a seguir, aplicando o polimorfismo de sobrecarga nos métodos apontados pelas setas. Em seguida, escreva um método main que cria diferentes brinquedos fazendo chamadas para seus diferentes métodos construtores e chamando diferentes métodos velocidade().



```

1 package Atividade10_pt2.entities;
2
3 public class Brinquedo {
4     protected String nome;
5     protected Double velocidade;
6     protected Double aceleracao;
7
8     public Brinquedo() {
9     }
10
11     public Brinquedo(String nome, Double velocidade, Double aceleracao) {
12         this.nome = nome;
13         this.velocidade = velocidade;
14         this.aceleracao = aceleracao;
15     }
16
17     public String getNome() {
18         return nome;
19     }
20
21     public void setNome(String nome) {
22         this.nome = nome;
23     }
24
25     public Double getVelocidade() {
26         return velocidade;
27     }
28
29     public void setVelocidade(Double velocidade) {
30         this.velocidade = velocidade;
31     }
32
33     public Double getAceleracao() {
34         return aceleracao;
35     }
36
37     public void setAceleracao(Double aceleracao) {
38         this.aceleracao = aceleracao;
39     }
40
41     @
    public void mover(Brinquedo brinquedo) {

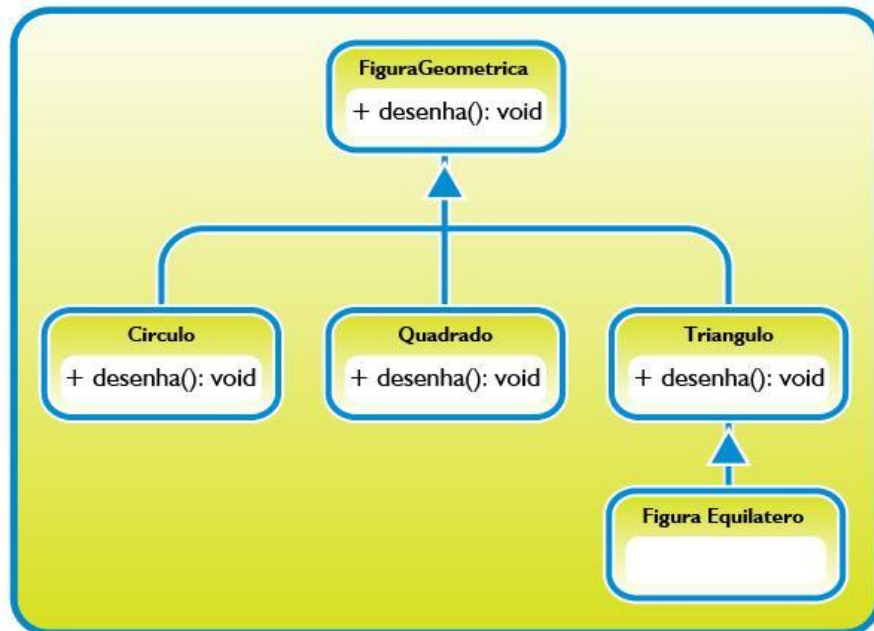
```

```

42         System.out.println("O brinquedo " + brinquedo.getNome() + " está movimentando");
43     }
44
45     @
    public void velocidade(Brinquedo brinquedo, int velocidade) {
46         System.out.println("O " + brinquedo.getNome() + " está com a velocidade de " + velocidade);
47     }
48
49     @
    public void velocidade(Brinquedo brinquedo, double velocidade) {
50         this.velocidade = velocidade;
51         System.out.println("O " + brinquedo.getNome() + " está com a velocidade de " + velocidade);
52     }
53
54     @
    public void velocidade(Brinquedo brinquedo, double velocidade, double aceleracao) {
55         this.velocidade = velocidade;
56         this.aceleracao = aceleracao;
57         System.out.println("O " + brinquedo.getNome() + " está com a velocidade de " + velocidade + " e aceleração de " + aceleracao);
58     }
59 }

```

9. Implemente as classes da hierarquia da classe `FiguraGeometrica` mostrada na Figura abaixo em Java, aplicando o polimorfismo de sobreposição para o método `desenha()`. Em seguida, crie uma classe `Principal` com um método `main` que cria um objeto de cada uma das classes e chama seus respectivos métodos `desenha()`.



```
1 package Atividade10_pt2.entities;
2
3 public class FiguraGeometrica {
4     public void desenha() {
5         System.out.println("Desenhando uma figura geometrica");
6     }
7 }
```

```
1 package Atividade10_pt2.entities;
2
3 public class Circulo extends FiguraGeometrica {
4     public void desenha() {
5         System.out.println("Desenhando um circulo");
6     }
7 }
```

```
1 package Atividade10_pt2.entities;
2
3 public class Quadrado extends FiguraGeometrica{
4     public void desenha() {
5         System.out.println("Desenhando um quadrado");
6     }
7 }
```

```

1 package Atividade10_pt2.entities;
2
3 public class Triangulo extends FiguraGeometrica {
4     public void desenha() {
5         System.out.println("Desenhando um triangulo");
6     }
7 }

```

```

1 package Atividade10_pt2.entities;
2
3 public class TrianguloEquilatero extends Triangulo{
4     public void desenha() {
5         System.out.println("Desenhando um triangulo equilatero");
6     }
7 }

```

## Main

```

1 package Atividade10_pt2.app;
2
3 import Atividade10_pt2.entities.*;
4
5 public class Main9 {
6     public static void main(String []args) {
7         FiguraGeometrica figuraGeometrica = new FiguraGeometrica();
8         Circulo circulo = new Circulo();
9         Quadrado quadrado = new Quadrado();
10        Triangulo triangulo = new Triangulo();
11        TrianguloEquilatero trianguloEquilatero = new TrianguloEquilatero();
12
13        figuraGeometrica.desenha();
14        circulo.desenha();
15        quadrado.desenha();
16        triangulo.desenha();
17        trianguloEquilatero.desenha();
18    }
19 }

```

```

"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaad
Desenhando uma figura geometrica
Desenhando um circulo
Desenhando um quadrado
Desenhando um triangulo
Desenhando um triangulo equilatero

Process finished with exit code 0

```

10. Implemente o diagrama de classes representado pela Figura abaixo. Para a classe CadastroPessoas considere o atributo pessoas como um array do tipo Pessoa.

O método cadastrarPessoa(): deve acrescentar ao arraypessoas um objeto descendente da classe Pessoa.

O método mostraCadastro(): deve percorrer todo o array de pessoas e mostrar todos os dados do descendente de Pessoa.

Aplique os tipos de polimorfismo em cada uma das situações solicitadas:

- a. **Polimorfismo de Sobrecarga:** crie mais de um método construtor para cada classe: Pessoa, Cliente, Funcionario e Gerente.

### Classe Pessoa

```
1 package Atividade10_pt2.entities;
2
3 public class Pessoa {
4     protected String nome;
5     protected String cpf;
6
7     public Pessoa() {
8     }
9
10    public Pessoa(String nome, String cpf) {
11        this.nome = nome;
12        this.cpf = cpf;
13    }
14
15    public String getNome() {
16        return nome;
17    }
18
19    public void setNome(String nome) {
20        this.nome = nome;
21    }
22
23    public String getCpf() {
24        return cpf;
25    }
26
27    public void setCpf(String cpf) {
28        this.cpf = cpf;
29    }
30
31    public String mostraDados() {
32        return "Pessoa{" +
33            "nome='" + nome + '\'' +
34            ", cpf='" + cpf + '\'' +
35            '}';
36    }
37 }
```

## Classe Cliente

```
1 package Atividade10_pt2.entities;
2
3 public class Cliente extends Pessoa {
4     protected int codigo;
5
6     public Cliente() {
7     }
8
9     public Cliente(String nome, String cpf, int codigo) {
10         super(nome, cpf);
11         this.codigo = codigo;
12     }
13
14     public int getCodigo() {
15         return codigo;
16     }
17
18     public void setCodigo(int codigo) {
19         this.codigo = codigo;
20     }
21
22     public String mostraDados() {
23         return "Cliente{" +
24             "codigo=" + codigo +
25             ", nome=" + nome + '\n' +
26             ", cpf=" + cpf + '\n' +
27             '}';
28     }
29 }
```

## Classe Funcionario

```
1 package Atividade10_pt2.entities;
2 public class Funcionario extends Pessoa {
3
4     protected String matricula;
5     protected float salario;
6
7     public Funcionario() {
8     }
9
10    public Funcionario(String nome, String cpf, String matricula, float salario) {
11        super(nome, cpf);
12        this.matricula = matricula;
13        this.salario = salario;
14    }
15
16    public String getMatricula() {
17        return matricula;
18    }
19
20    public void setMatricula(String matricula) {
21        this.matricula = matricula;
22    }
23
24    public float getSalario() {
25        return salario;
26    }
27
28    public void setSalario(float salario) {
29        this.salario = salario;
30    }
31
32    public String mostraDados() {
33        return "Funcionario{" +
34            "matricula=" + matricula + '\n' +
35            ", salario=" + salario +
36            ", nome=" + nome + '\n' +
37            ", cpf=" + cpf + '\n' +
38            '}';
39    }
40 }
```

## Classe Gerente

```
1 package Atividade10_pt2.entities;
2
3 public class Gerente extends Funcionario {
4     protected int area;
5
6     public Gerente() {
7     }
8
9     public Gerente(String nome, String cpf, String matricula, float salario, int area) {
10         super(nome, cpf, matricula, salario);
11         this.area = area;
12     }
13
14     public int getArea() {
15         return area;
16     }
17
18     public void setArea(int area) {
19         this.area = area;
20     }
21
22     public String mostraDados() {
23         return "Gerente{" +
24             "matricula=" + matricula + '\n' +
25             ", salario=" + salario +
26             ", area=" + area +
27             ", nome=" + nome + '\n' +
28             ", cpf=" + cpf + '\n' +
29             '}';
30     }
31 }
```

- b. **Polimorfismo de Sobreposição:** faça com que o método `mostraCadastro()` utilize o método `mostraDados()` correto, dependendo se a Pessoa é um Cliente, Funcionario ou Gerente.

```
1 package Atividade10_pt2.app;
2
3 import Atividade10_pt2.entities.Cliente;
4 import Atividade10_pt2.entities.Funcionario;
5 import Atividade10_pt2.entities.Gerente;
6 import Atividade10_pt2.entities.Pessoa;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 public class Main10 {
11
12     public List<Pessoa> listaPessoa = new ArrayList<Pessoa>();
13
14     public void cadastrarPessoa(Pessoa pessoa) {
15         listaPessoa.add(pessoa);
16     }
17
18     public void mostrarCadastro() {
19         for(Pessoa pessoa : listaPessoa) {
20             System.out.println(pessoa.mostraDados());
21         }
22     }
23
24     public static void main(String[] args) {
25         Main10 main10 = new Main10();
26         Pessoa pessoa = new Pessoa( nome: "Lucas", cpf: "010101");
27         Pessoa cliente = new Cliente( nome: "Jucelio", cpf: "0202002", codigo: 2);
28         Pessoa funcionario = new Funcionario( nome: "Jose", cpf: "0303030", matricula: "3", salario: 3560);
29         Pessoa gerente = new Gerente( nome: "Josefa", cpf: "040404", matricula: "4", salario: 5640, area: 4);
30
31         main10.cadastrarPessoa(pessoa);
32         main10.cadastrarPessoa(cliente);
33         main10.cadastrarPessoa(funcionario);
34         main10.cadastrarPessoa(gerente);
35         main10.mostrarCadastro();
36     }
37 }
```



- c. **Polimorfismo de Inclusão:** quando for adicionar ao array `listasPessoas` uma nova pessoa que pode ser de um dos tipos descendentes de `Pessoa`.

```
1 package Atividade10_pt2.app;
2
3 import Atividade10_pt2.entities.Cliente;
4 import Atividade10_pt2.entities.Funcionario;
5 import Atividade10_pt2.entities.Gerente;
6 import Atividade10_pt2.entities.Pessoa;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 public class Main10 {
11
12     public List<Pessoa> listasPessoa = new ArrayList<Pessoa>();
13
14     public void cadastrarPessoa(Pessoa pessoa) {
15         listasPessoa.add(pessoa);
16     }
17
18     public void mostrarCadastro() {
19         for(Pessoa pessoa : listasPessoa) {
20             System.out.println(pessoa.mostraDados());
21         }
22     }
23
24     public static void main(String[] args) {
25         Main10 main10 = new Main10();
26         Pessoa pessoa = new Pessoa( nome: "Lucas", cpf: "010101");
27         Pessoa cliente = new Cliente( nome: "Jucelio", cpf: "0202002", codigo: 2);
28         Pessoa funcionario = new Funcionario( nome: "Jose", cpf: "0303030", matricula: "3", salario: 3560);
29         Pessoa gerente = new Gerente( nome: "Josefa", cpf: "040404", matricula: "4", salario: 5640, area: 4);
30
31         main10.cadastrarPessoa(pessoa);
32         main10.cadastrarPessoa(cliente);
33         main10.cadastrarPessoa(funcionario);
34         main10.cadastrarPessoa(gerente);
35         main10.mostrarCadastro();
36     }
37 }
38
```

