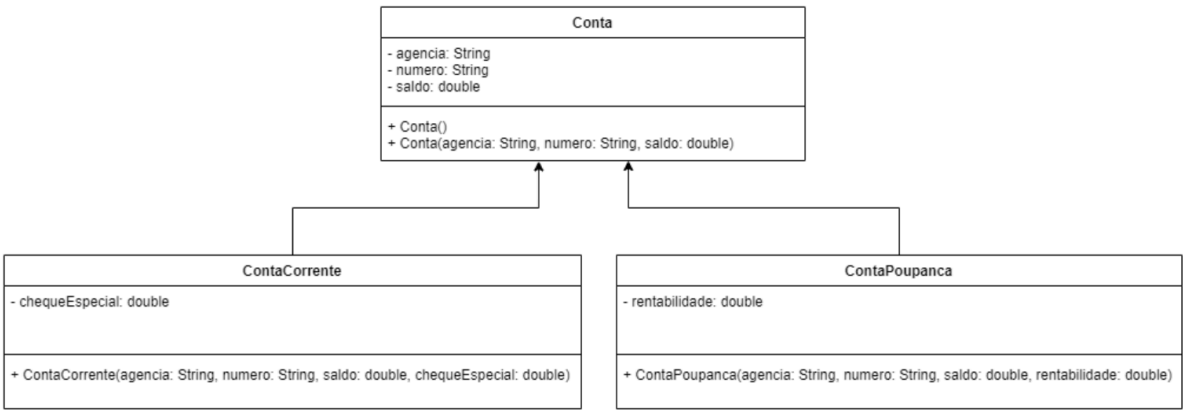
Aula 09 - Herança

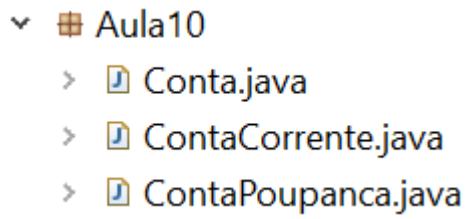
1. Herança

Utilizamos herança na programação, quando queremos reaproveitar o código já desenvolvido anteriormente em uma classe. A Herança é aplicada em uma classe e permite **herdar atributos, métodos e construtores**. Quando aplicamos herança, temos a figura da super classe e da sub classe.

Em UML representamos a herança da seguinte forma:



Vamos desenvolver o seguinte projeto:



Com o seguinte conteúdo:

Conta.java (Super Classe)

```
package Ala10;

public class Conta {
    private String agencia;
    private String numero;
    private double saldo;

    public String getAgencia() {
        return agencia;
    }
```

```
    public void setAgencia(String agencia) {
        this.agencia = agencia;
    }

    public String getNumero() {
        return numero;
    }

    public void setNumero(String numero) {
        this.numero = numero;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    protected String exibirMensagem() {
        return "Bem vindo ao Internet Banking do Prof. Joseffe!";
    }

    Conta(){

    }

    Conta(String agencia, String numero, double saldo){
        this.agencia = agencia;
        this.numero = numero;
        this.saldo = saldo;
    }
}
```

ContaCorrente.java (Sub Classe)

```
package Aula10;

public class ContaCorrente extends Conta{
    private double chequeEspecial;

    public double getChequeEspecial() {
```

```
        return chequeEspecial;
    }

    public void setChequeEspecial(double chequeEspecial) {
        this.chequeEspecial = chequeEspecial;
    }

    public ContaCorrente() {
    }

    public ContaCorrente(String agencia, String numero, double saldo,
double chequeEspecial) {
        super(agencia, numero, saldo);

        this.chequeEspecial = chequeEspecial;
    }
}
```

ContaPoupanca.java (Sub Classe)

```
package Aula10;

public class ContaPoupanca extends Conta{
    private double rentabilidade;

    public double getRentabilidade() {
        return rentabilidade;
    }

    public void setRentabilidade(double rentabilidade) {
        this.rentabilidade = rentabilidade;
    }

    public ContaPoupanca() {
    }

    public ContaPoupanca(String agencia, String numero, double saldo,
double rentabilidade) {
        super(agencia, numero, saldo);

        this.rentabilidade = rentabilidade;
    }
```

```
}
```

Nesse caso, a classe Conta é a super classe, onde temos tudo já feito e desenvolvido. Já as classes ContaCorrente e ContaPoupanca são novas classes e estão herdando tudo (construtor, métodos e atributos) da classe Conta.

Perceba que até o construtor foi herdado com a palavra reservada “super”. Entretanto, um novo atributo foi incluído na classe ContaPoupanca, chamado “rendimento”.

Esse atributo só existe nessa classe. Ou seja, a classe ContaPoupanca tem tudo o que a classe Conta tem e também mais algumas coisas específicas dela.

O mesmo aconteceu com a classe ContaCorrente, com o atributo chequeEspecial.

Programa

```
package Aula10;

import java.util.Scanner;

public class Programa {

    public static void main(String[] args) {
        Scanner ler = new Scanner(System.in);

        ContaCorrente cc = new ContaCorrente();

        cc.setAgencia("0001");
        cc.setNumero("14769");
        cc.setSaldo(100);

        cc.setChequeEspecial(500);

        System.out.printf("Conta Corrente: Ag: %s, Num: %s, Saldo:
%.2f, Chq Esp: %.2f", cc.getAgencia(), cc.getNumero(), cc.getSaldo(),
cc.getChequeEspecial() );



        ContaPoupanca cp = new ContaPoupanca();

        cp.setAgencia("0002");
        cp.setNumero("32456");
        cp.setSaldo(2000);
```

```
        cp.setRentabilidade(2);

        System.out.printf("\n\n\nConta Poupança: Ag: %s, Num: %s,
Saldo: %.2f, Rent: %.2f", cp.getAgencia(), cp.getNumero(),
cp.getSaldo(), cp.getRentabilidade() );
    }

}
```

1.1 Especificadores de acesso com Herança

Por padrão, na linguagem JAVA, temos os seguintes modificadores de acesso e suas respectivas visibilidades em Classe, Pacote, Subclasse e Globalmente, veja:

Modificador	Classe	Pacote	Subclasse	Globalmente
Public	*Sim*	*Sim*	*Sim*	*Sim*
Protected	*Sim*	*Sim*	*Sim*	*Não*
Sem Modificador (Padrão)	*Sim*	*Sim*	*Não*	*Não*
Private	*Sim*	*Não*	*Não*	*Não*

Exercícios

33. Vamos fazer um sistema de cadastro de contas (apenas inclusão). O sistema deve ter as opções Conta Corrente, Conta Poupança e Conta Salário. Utilize Herança nas classes. Faça o Diagrama de Classe (UML) desse sistema também.

Menu:

1 - Conta Corrente
2 - Conta Poupança
3 - Conta Salário
4 - Exibir Contas
5 - Sair

Conta.java

```
package Ex33;

public class Conta {
    private String agencia;
    private String numero;
    private double saldo;
```

```
    public String getAgencia() {
        return agencia;
    }

    public void setAgencia(String agencia) {
        this.agencia = agencia;
    }

    public String getNumero() {
        return numero;
    }

    public void setNumero(String numero) {
        this.numero = numero;
    }

    public double getSaldo() {
        return saldo;
    }

    Conta(){
        this.saldo = 0.0;
    }

    Conta(String agencia, String numero){
        this.agencia = agencia;
        this.numero = numero;
        this.saldo = 0.0;
    }
}
```

ContaCorrente.java

```
package Ex33;


public class ContaCorrente extends Conta {
    private double chequeEspecial;

    public double getChequeEspecial() {
        return chequeEspecial;
    }

    public void setChequeEspecial(double chequeEspecial) {
```

```
        this.chequeEspecial = chequeEspecial;
    }

    public ContaCorrente() {
    }

    public ContaCorrente(String agencia, String numero, double
chequeEspecial) {
        super(agencia, numero);
        this.chequeEspecial = chequeEspecial;
    }
}
```

ContaPoupanca.java

```
package Ex33;

public class ContaPoupanca extends Conta {
    private double rentabilidade;

    public double getRentabilidade() {
        return rentabilidade;
    }

    public void setRentabilidade(double rentabilidade) {
        this.rentabilidade = rentabilidade;
    }

    public ContaPoupanca() {
    }

    public ContaPoupanca(String agencia, String numero, double
rentabilidade) {
        super(agencia, numero);
        this.rentabilidade = rentabilidade;
    }
}
```

ContaSalario.java

```
package Ex33;

public class ContaSalario extends Conta {


```

```
    public ContaSalario() {

    }

    public ContaSalario(String agencia, String numero) {
        super(agencia, numero);
    }
}
```

Programa.java

```
package Ex33;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

public class Programa {

    public static void main(String[] args) throws IOException {

        Scanner ler = new Scanner(System.in);

        ArrayList<Conta> listaConta = new ArrayList<>();

        int opcao;
        int numConta = 4022;

        do {
            System.out.printf("===> Sistemas de operações <===\n\n");
            System.out.printf("Escolha uma opção:\n");
            System.out.printf("1 - Conta corrente \n" +
                              "2 - Conta poupança \n" +
                              "3 - Conta salário \n" +
                              "4 - Listar contas \n" +
                              "5 - Sair\n\n");

            System.out.printf("Digite a opção desejada: ");
            opcao = ler.nextInt();

            if (opcao == 1) {
                ContaCorrente contaCorrente = new ContaCorrente();

                System.out.printf("Digite a agência: ");
```

```
                contaCorrente.setAgencia(ler.next());

                System.out.printf("Digite o número da conta: ");
                contaCorrente.setNumero(ler.next());

                //numConta++;

                System.out.println("Digite o valor do cheque especial:
");
                contaCorrente.setChequeEspecial(ler.nextDouble());

                listaConta.add(contaCorrente);


                System.out.printf("Conta cadastrada com sucesso!");
                System.in.read();
            }
            else if (opcao == 2) {
                ContaPoupanca contaPoupanca = new ContaPoupanca();

                System.out.printf("Digite a agência: ");
                contaPoupanca.setAgencia(ler.next());

                System.out.printf("Digite o número da conta: ");
                contaPoupanca.setNumero(ler.next());

                //numConta++;

                System.out.println("Digite o valor da rentabilidade:
");
                contaPoupanca.setRentabilidade(ler.nextDouble());

                listaConta.add(contaPoupanca);


                System.out.printf("Conta cadastrada com sucesso!");
                System.in.read();
            }
            else if (opcao == 3) {
                ContaSalario contaSalario = new ContaSalario();

                System.out.printf("Digite a agência: ");
                contaSalario.setAgencia(ler.next());
```

```
                System.out.printf("Digite o número da conta: ");
                contaSalario.setNumero(ler.next());

                //numConta++;

                listaConta.add(contaSalario);

                System.out.printf("Conta cadastrada com sucesso!");
                System.in.read();
            }
            else if (opcao == 4) {
                for (Conta c: listaConta) {

                    if (c instanceof ContaCorrente) {
                        ContaCorrente cc = (ContaCorrente) c;
                        System.out.println("Tipo: CC | Agência: " +
cc.getAgencia() + " | Número: " + cc.getNumero() + " | Cheque Especial:
" + cc.getChequeEspecial());
                    }
                    else if(c instanceof ContaPoupanca) {
                        ContaPoupanca cp = (ContaPoupanca) c;
                        System.out.println("Tipo: CP | Agência: " +
cp.getAgencia() + " | Número: " + cp.getNumero() + " | Rentabilidade: "
+ cp.getRentabilidade());
                    }
                    else {
                        System.out.println("Tipo: CS | Agência: " +
c.getAgencia() + " | Número: " + c.getNumero());
                    }
                }

                System.in.read();
            }
        } while ( (opcao >= 1) && (opcao <= 4) );
    }
}
```

34. Vamos fazer um sistema de cadastro de pessoas (apenas inclusão). O sistema deve ter as opções Pessoa Física e Pessoa Jurídica. Utilize Herança nas classes. Faça o Diagrama de Classe (UML) desse sistema também.

Menu:

1 - Pessoa Física
2 - Pessoa Jurídica
3 - Exibir Pessoas
4 - Sair

Pessoa.java

```java
package Ex34;

public class Pessoa {

    private String nome;
    private String sobrenome;
    private int idade;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getSobrenome() {
        return sobrenome;
    }

    public void setSobrenome(String sobrenome) {
        this.sobrenome = sobrenome;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }

    Pessoa() {
    }

    Pessoa(String nome, String sobrenome, int idade){
        this.nome = nome;
```

```
        this.sobrenome = sobrenome;
        this.idade = idade;
    }
}
```

PessoaFisica.java

```
package Ex34;

public class PessoaFisica extends Pessoa{

    String CPF;

    public String getCPF() {
        return CPF;
    }

    public void setCPF(String CPF) {
        this.CPF = CPF;
    }

    PessoaFisica(){

    }
    PessoaFisica(String nome, String sobrenome, int idade, String CPF){
        super(nome, sobrenome, idade);
        this.CPF = CPF;
    }
}
```

PessoaJuridica.java

```
package Ex34;

public class PessoaJuridica extends Pessoa{
    String CNPJ;

    public String getCNPJ() {
        return CNPJ;
    }

    public void setCNPJ(String CNPJ) {
        this.CNPJ = CNPJ;
    }
```

```
    PessoaJuridica(){

    }

    PessoaJuridica(String nome, String sobrenome, int idade, String
CNPJ){
        super(nome, sobrenome, idade);
        this.CNPJ = CNPJ;
    }
}
```

Programa.java

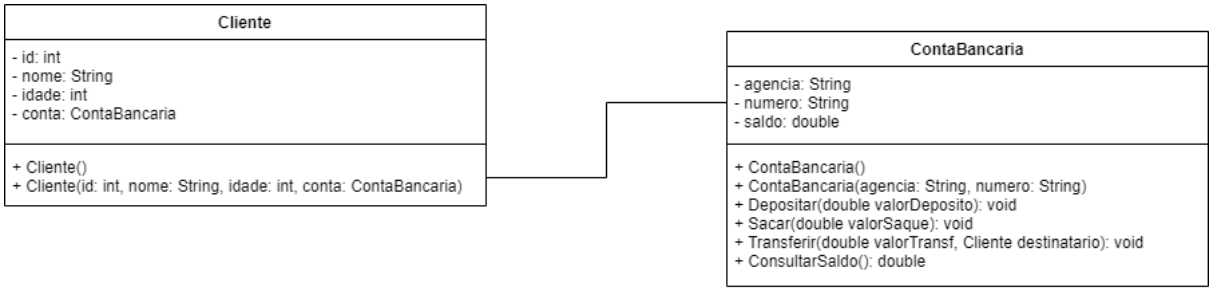
```
package Ex34;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

public class Programa {
    public static void main(String[] args) throws IOException {

        Scanner ler = new Scanner(System.in);

        ArrayList<Pessoa> listaPessoa = new ArrayList<>();

        int opcao;

        do {
            System.out.printf("===> Sistemas de operações <===\n\n");
            System.out.printf("Escolha uma opção:\n");
            System.out.printf("1 - Pessoa Fisica \n" +
                    "2 - Pessoa Juridica \n" +
                    "3 - Listar Pessoas \n" +
                    "4 - Sair\n\n");

            System.out.printf("Digite a opção desejada: ");
            opcao = ler.nextInt();

            if (opcao == 1) {
                PessoaFisica PessoaFisica = new PessoaFisica();

                System.out.println("Digite o seu nome: ");
                PessoaFisica.setNome(ler.next());
```

```
            System.out.println("Digite o seu Sobrenome: ");
            PessoaFisica.setSobrenome(ler.next());

            System.out.println("Digite a sua idade: ");
            PessoaFisica.setIdade(ler.nextInt());

            System.out.println("Digite o seu CPF: ");
            PessoaFisica.setCPF(ler.next());


            listaPessoa.add(PessoaFisica);


            System.out.printf("Pessoa cadastrada com sucesso!");
            System.in.read();
        }
        else if (opcao == 2) {
            PessoaJuridica PessoaJuridica = new PessoaJuridica();


            System.out.println("Digite o seu nome: ");
            PessoaJuridica.setNome(ler.next());

            System.out.println("Digite o seu Sobrenome: ");
            PessoaJuridica.setSobrenome(ler.next());

            System.out.println("Digite a sua idade: ");
            PessoaJuridica.setIdade(ler.nextInt());

            System.out.println("Digite o seu CNPJ: ");
            PessoaJuridica.setCNPJ(ler.next());

            listaPessoa.add(PessoaJuridica);


            System.out.printf("Pessoa cadastrada com sucesso!");
            System.in.read();
        }
        else if (opcao == 3) {
            for (Pessoa c: listaPessoa) {
                if (c instanceof PessoaFisica)
                    System.out.println("|Pessoa Fisica|");
```

```
                    else
                        System.out.println("|Pessoa Juridica|");


                    System.out.println("Nome: " + c.getNome());
                    System.out.println("Sobrenome: " +
c.getSobrenome());
                    System.out.println("Idade: " + c.getIdade());

                    if (c instanceof PessoaFisica) {
                        PessoaFisica cc = (PessoaFisica) c;
                        System.out.println("CPF: " + cc.getCPF() +
"\n--------------------------------");
                    }
                    else if(c instanceof PessoaJuridica) {
                        PessoaJuridica cp = (PessoaJuridica) c;
                        System.out.println("CNPJ: " + cp.getCNPJ() +
"\n--------------------------------");
                    }
                }


                System.in.read();
            }
        } while ( (opcao >= 1) && (opcao <= 3) );
    }
}
```

35. Crie um sistema para o Banco Macgyver. O sistema deverá permitir o cadastro de clientes e criação de conta. Além disso, o usuário poderá realizar depósitos, saques e transferências entre clientes. Implemente também validações em seu sistema, por exemplo: Não permitir realizar saques ou transferências se o cliente não tiver saldo suficiente, etc.

Diagrama de classe:



Layout do sistema:

Banco Macgyver

1 - Criar cliente/conta
2 - Depósito
3 - Saque
4 - Transferência
5 - Consulta de Saldo
6 - Sair