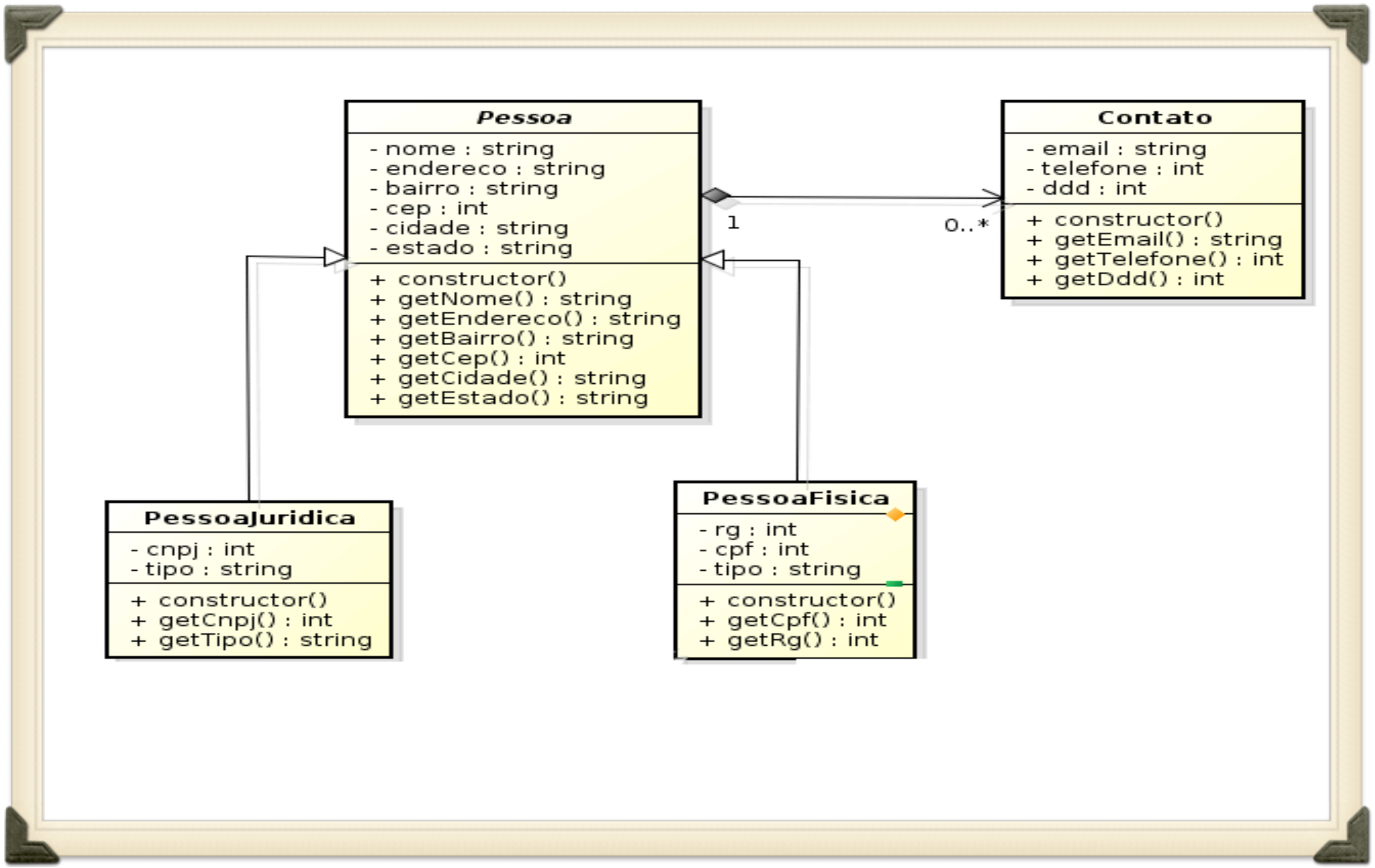
Relembrando…



Relembrando…

1. Implementar um programa principal no qual seja possível cadastrar, remover, atualizar e pesquisar por pessoas jurídicas e físicas.

```
17  public class Principal {
18
19      private static final int PESSOA = 1;
20      private static final int EMPRESA = 2;
21
22      public static void main(String[] args) throws IOException {
23          BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
24          ArrayList<PessoaFisica> pessoas = new ArrayList();
25          ArrayList<PessoaJuridica> empresas = new ArrayList();
26          int tipo_pessoa = lerInteiro("MENU 1: \n [1] Pessoa Física "
27                  + "\n [2] Pessoa Jurídica", br);
28          int opcao = 5;
29
30          do {
31              opcao = lerInteiro("MENU 2: \n [1] Cadastrar \n [2] Remover "
32                      + "\n [3] Atualizar \n [4] Pesquisar \n [5] Sair", br);
33              switch (opcao) {
34                  case 1:
35                      cadastrar(pessoas, empresas, tipo_pessoa, br);
36                      break;
37                  case 2:
38                      remover(br, tipo_pessoa, pessoas, empresas);
39                      break;
40                  case 3:
41                      atualizar(br, tipo_pessoa, pessoas, empresas);
42                      break;
43                  case 4:
44                      pesquisar(br, tipo_pessoa, pessoas, empresas);
45                      break;
```

Relembrando…

1. Implementar um programa principal no qual seja possível cadastrar, remover, atualizar e pesquisar por pessoas jurídicas e físicas.

```
170     private static void cadastrar(ArrayList<PessoaFisica> pessoas,
171             ArrayList<PessoaJuridica> empresas,
172             int tipo_pessoa, BufferedReader br) throws IOException {
173         String nome = lerTexto("Qual o seu nome?", br);
174         String endereco = lerTexto("Qual o seu endereço?", br);
175         String bairro = lerTexto("Qual o seu bairro?", br);
176         String cidade = lerTexto("Qual a sua cidade?", br);
177         String estado = lerTexto("Qual o seu estado?", br);
178         int cep = lerInteiro("Qual o seu CEP?", br);
179         Pessoa p;
180 
181         switch (tipo_pessoa) {
182             case PESSOA:
183                 adicionaPessoa(br, nome, endereco, bairro, cidade, estado, cep, pessoas);
184                 break;
185             case EMPRESA:
186                 adicionaEmpresa(br, nome, endereco, bairro, cidade, estado, cep, empresas);
187                 break;
188             default:
189                 System.err.println("Opção inválida.");
190         }
191     }
```

```
193     private static int lerInteiro(String opcoes, BufferedReader br) throws IOException {
194         int opcao = Integer.parseInt(lerTexto(opcoes, br));
195 
196         return opcao;
197     }
198 
199     private static String lerTexto(String mensagem, BufferedReader br) throws IOException {
200         System.out.println(mensagem);
201         return br.readLine();
202     }
```

Relembrando…

1. Implementar um programa principal no qual seja possível cadastrar, remover, atualizar e pesquisar por pessoas jurídicas e físicas.

```
90      private static void adicionaPessoa(BufferedReader br, String nome,
91              String endereco, String bairro, String cidade, String estado,
92              int cep, ArrayList<PessoaFisica> pessoas) throws IOException {
93          Pessoa p;
94          int rg = lerInteiro("Qual o seu RG?", br);
95          int cpf = lerInteiro("Qual o seu CPF?", br);
96          String profissao = lerTexto("Qual a sua profissão?", br);
97          p = new PessoaFisica(rg, cpf, profissao,
98                  nome, endereco, bairro, cidade, estado, cep);
99          pessoas.add((PessoaFisica) p);
100     }
101
102     private static void adicionaEmpresa(BufferedReader br, String nome,
103             String endereco, String bairro, String cidade, String estado,
104             int cep, ArrayList<PessoaJuridica> empresas) throws IOException {
105         Pessoa p;
106         int cnpj = lerInteiro("Qual o seu CNPJ?", br);
107         String tipo_empresa = lerTexto("Qual o seu tipo?", br);
108         p = new PessoaJuridica(cnpj, tipo_empresa,
109                 nome, endereco, bairro, cidade, estado, cep);
110         empresas.add((PessoaJuridica) p);
111     }
```

Classes Abstratas

Programação Orientada a Objetos

Rafael Vieira Coelho

O QUE SERIA UMA CLASSE ABSTRATA?

- Nunca pode ser instanciada.

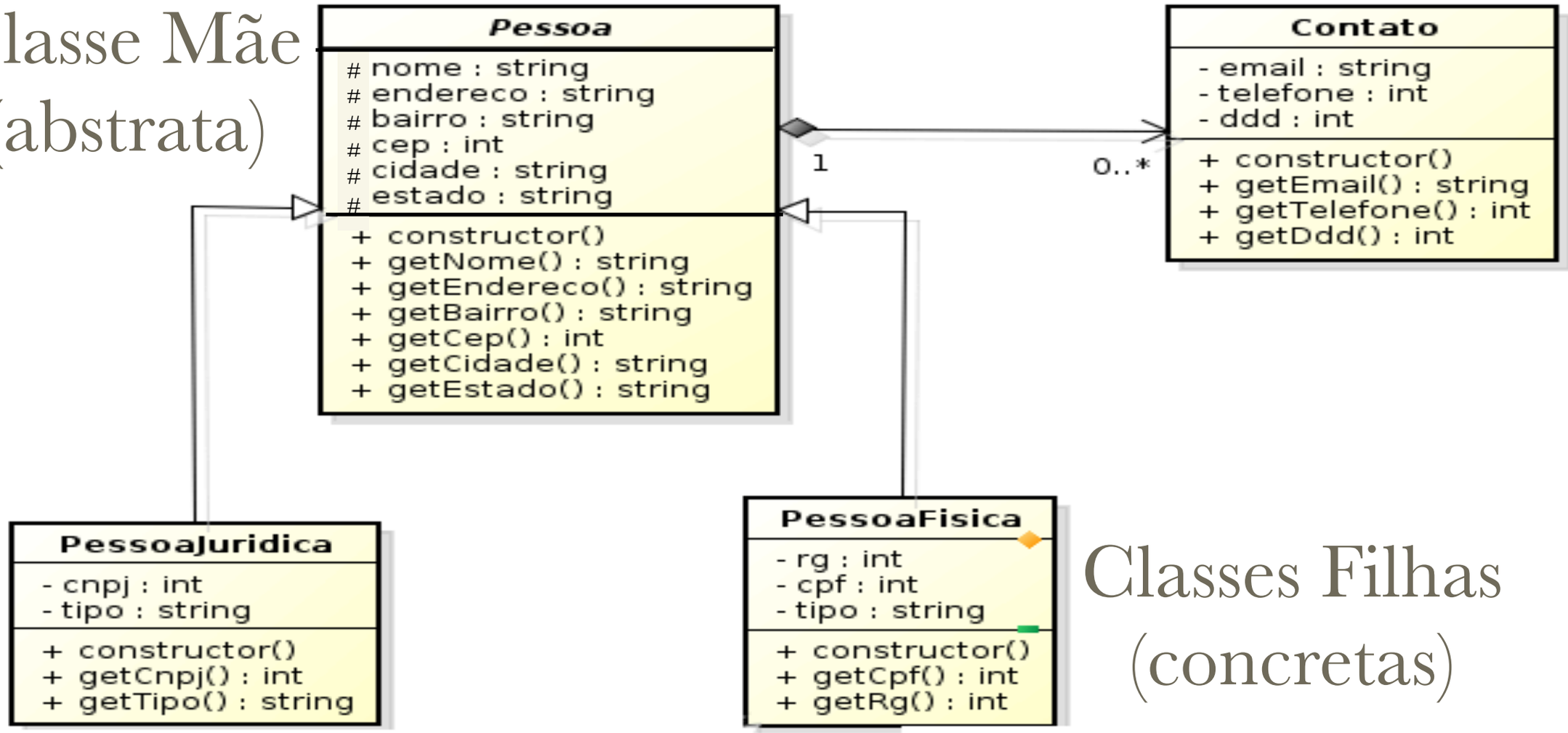
- Pode ter métodos concretos.

- Pode ter métodos abstratos (sem implementação).

- Nunca teremos um objeto Pessoa, apenas PessoaFisica ou PessoaJuridica.

Implementaremos

Classe Mãe (abstrata)



Classes Filhas (concretas)

Classe pessoa (abstrata)

- A palavra reservada ***abstract*** deve ser adicionada na declaração da classe.
- Método abstrato **mostraContatos** deve ser implementado pelas subclasses.

```
16  public abstract class Pessoa {
17      protected ArrayList<Contato> contatos;
18      protected String nome, endereco, bairro, cidade, estado;
19      protected int cep;
20
21      public abstract void mostraContatos();
22
23      public boolean adicionaContato(Contato c) {
24          return this.contatos.add(c);
25      }
26
27      public boolean removeContato(Contato c) {
28          return this.contatos.remove(c);
29      }
30
31      public boolean existeContato(Contato c) {
32          return this.contatos.contains(c);
33      }
34
35      public void apagaContatos() {
36          this.contatos.clear();
```

Classe PessoaFisica (subclasse)

- Extende a classe Pessoa (extends)
- Atributos privados (private)
- Métodos get/set
- Implementa o método mostraContatos.

```
 7  public class PessoaFisica extends Pessoa {
 8      private int rg, cpf;
 9      private String tipo;
10
11      @Override
12      public void mostraContatos() {
13          for (Contato contato : super.contatos) {
14              System.out.println(contato);
15          }
16      }
17
18      public int getRg() {
19          return rg;
20      }
```

Classe PessoaJuridica (subclasse)

- O método mostraContatos() da classe PessoaJuridica é diferente do método de mesmo nome da classe PessoaFísica.

```
12  public class PessoaJuridica extends Pessoa {
13
14      private int cnpj;
15      private String tipo;
16
17      @Override
        public void mostraContatos() {
19          for (Contato contato : super.contatos) {
20              System.out.println("Email: " + contato.getEmail());
21              System.out.println("Telefone: (" + contato.getDdd() + ") " + contato.getTelefone());
22          }
23      }
```

Enquanto isto...na classe contato

- Adicionaremos o método toString()

```
69      @Override
        public String toString() {
71          return "Contato{" + "email=" + email + ", telefone=" + telefone + ", ddd=" + ddd + '}';
72      }
```

Exemplo de uso

- Mas professor, você não disse que classe abstrata não pode ser instanciada?

- O que acontece quando chamamos os métodos mostraContatos()?

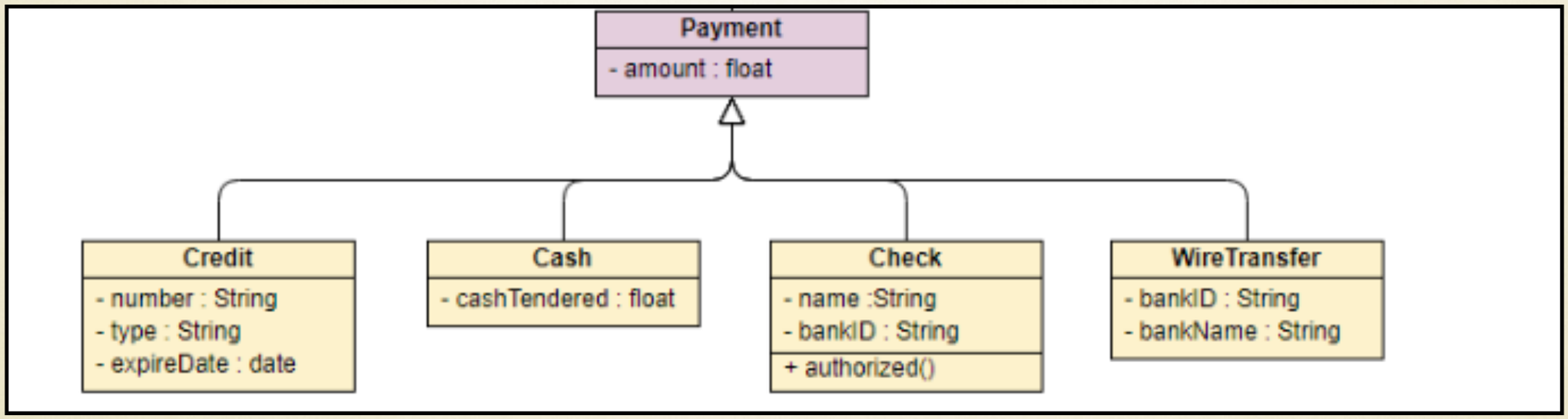
```
12  public class ExemploUso {
13      public static void main(String[] args) {
14          Pessoa p1, p2;
15
16          p1 = new PessoaFisica(107169, 8220051, "Engenheiro",
17              "Rafael Vieira Coelho", "R. Carlos Maggioni", "Centro",
18              "Farroupilha", "RS", 95180000);
19
20          p2 = new PessoaJuridica(10637926, "Instituição de Ensino",
21              "IFRS - Campus Farroupilha", "Av. São Vicente, 785",
22              "Cinquentenário", "Farroupilha", "RS", 95174274);
23
24          Contato c = new Contato("rafaelvc2@gmail.com", 89898298, 54);
25
26          p1.adicionaContato(c);
27          p2.adicionaContato(c);
28
29          p1.mostraContatos();
30          System.out.println("");
31          p2.mostraContatos();
32      }
```

```
Saída - Aula12Exemplos (run)
run:
Contato{email=rafaelvc2@gmail.com, telefone=89898298, ddd=54}

Email: rafaelvc2@gmail.com
Telefone: (54) 89898298
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Tarefas

1. Implemente a hierarquia de classes abaixo.



2. Crie um programa principal no qual o usuário possa escolher a forma de pagamento e informar os dados correspondentes.