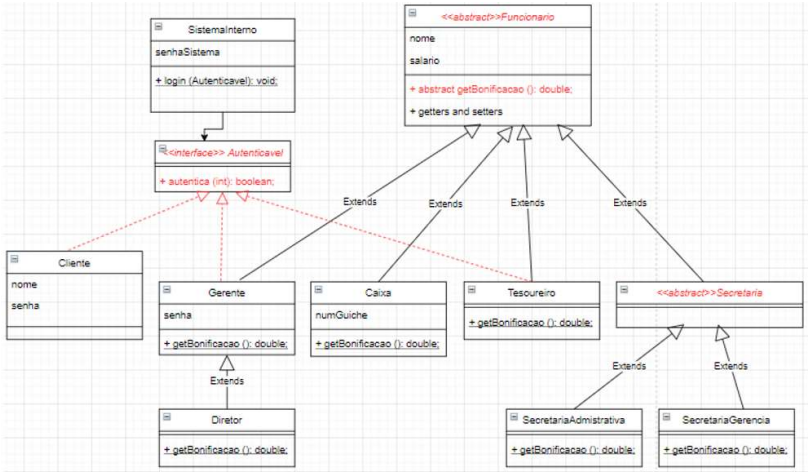
Aula 09 – parte 2
Exercício projeto bancov011. Exercício para ser feito em classe.

1) A partir de bancov010 crie o projeto bancov011. Abra o projeto e feche todos os outros.
2) Observe bem a hierarquia abaixo.



3) Crie a interface Autenticavel.

```
3 public interface Autenticavel {
4     boolean autentica(int senha);
5 }
```

4) Pegue toda a lógica de autenticação contida em FuncionarioAutenticavel e coloque em Gerente e Tesoureiro. O atributo senha deverá voltar a ser private e essas classes voltarão a herdar de Funcionario. As duas classes também deverão implementar a interface Autenticavel. Veja como deve ficar.

Gerente

```
2  public class Gerente extends Funcionario implements Autenticavel{
3      private int senha;
4
5      public int getSenha() {
6          return this.senha;
7      }
8      public void setSenha(int senha) {
9          this.senha = senha;
10     }
11     public boolean autentica(int senha) {
12         if (this.senha == senha)
13             return true;
14         return false;
15     }
16     @Override
17     public double getBonificacao() {
18         return this.salario * 0.30;
19     }
20 }
```

Tesoureiro

```java
public class Tesoureiro extends Funcionario implements Autenticavel{
    private int senha;
    public int getSenha() {
        return this.senha;
    }
    public void setSenha(int senha) {
        this.senha = senha;
    }
    public boolean autentica(int senha) {
        if (this.senha == senha)
            return true;
        return false;
    }
    @Override
    public double getBonificacao() {
        //super é mais descritivo uma vez que o atributo está na superclasse
        return super.salario * 0.20;
    }
}
```

5) A classe FuncionarioAutenticavel não tem mais nenhuma utilidade. Exclua essa classe.
6) Ainda assim, as classes de teste acusam erro. Precisamos alterar a assinatura do método login de SistemaInterno para que as classes de teste voltem a funcionar.

```java
public class SistemaInterno {
    private int senhaDoSiatema = 123;

    public void login(Autenticavel a) {
        if(a.autentica(this.senhaDoSiatema)==true)
            System.out.println("Acesso liberado!");
        else
            System.out.println("Acesso negado.");
    }
}
```

7) Experimente retirar o método autentica de Tesoureiro ou Gerente. Uma vez implementada a interface, a implementação desses métodos passa a ser obrigatória. Ao implementar Autenticavel você garante que fornecerá uma implementação para autentica.
8) Para nosso sistema ficar completo, precisamos fazer cliente implementar Autenticavel e fornecer uma assinatura para autentica. O próprio Eclipse vai se oferecer para adicionar os métodos não implementados. Aproveite para colocar a anotação @Override em Gerente e Tesoureiro. Veja como deve ficar essa classe.

```java
public class Cliente implements Autenticavel{
    private String nome;
    private String cpf;
    private String email;
    private int senha;
    public int getSenha() {
        return this.senha;
    }
    public void setSenha(int senha) {
        this.senha = senha;
    }
    @Override
    public boolean autentica(int senha) {
        if (this.senha == senha)
            return true;
        return false;
    }
    //Construtores
    public Cliente(String nome) {
        this.setNome(nome);//Aproveitando as regras do setNome
    }
    public Cliente(String nome, String cpf) {
        this(nome);//Invocando o outro construtor que já tem as regras
        this.setCpf(cpf);//Reaproveitando a regra do cpf
    }
    public Cliente() {
```

9) Modifique TestaSistemaInterno2 para autenticar um Cliente também. Você verá que tudo continuará funcionando perfeitamente. Afinal, Cliente, Gerente,Diretor e Tesoureiro são Autenticavel!

```java
public class TestaSistemaInterno2 {
    public static void main(String[] args) {
        SistemaInterno si = new SistemaInterno();
        Gerente g1,g2;
        g1 = new Gerente();
        g2 = new Gerente();
        g1.setNome("Rafael");
        g1.setSenha(123);
        g2.setNome("Renata");
        g2.setSenha(124);
        Diretor d1 = new Diretor();
        d1.setNome("Reinaldo");
        d1.setSenha(123);
        Tesoureiro t = new Tesoureiro();
        t.setNome("Nilzete");
        t.setSenha(123);
        Cliente cli = new Cliente("Fulano");
        cli.setSenha(123);

        si.login(g1);
        si.login(g2);
        si.login(d1);
        si.login(t);
        si.login(cli);
    }
}
```

10) Podem surgir novas regras em nosso sistema. E se Gerente não puder mais autenticar no sistema e Gerente puder? Problema nenhum! E se Caixa precisar se autenticar? Continuamos sem problemas. Se alguma nova classe surgir e precisar se autenticar no sistema? Continua sendo fácil resolver. Com o uso de interfaces, ganhamos polimorfismo com baixo acoplamento e muita flexibilidade! Faça testes e verifique as situações sugeridas acima.