

Classe Funcionário, Estagiário e Programador

Criar um projeto -> Aula07

Criar um pacote chamado entity.

Criar uma classe abstrata de Funcionario contendo os atributos (código, nome, sexo e salario), os construtores vazio e cheio, toString, e getters e setters.

```
package entity;

public abstract class Funcionario {

    private Integer codigo;
    private String nome;
    private String sexo;
    private Double salario;

    public Funcionario() {

    }

    public Funcionario(Integer codigo, String nome, String sexo,
Double salario) {
        super();
        this.codigo = codigo;
        this.nome = nome;
        this.sexo = sexo;
        this.salario = salario;
    }

    @Override
    public String toString() {
        return "Funcionario [codigo=" + codigo + ", nome=" +
nome + ", sexo=" + sexo + ", salario=" + salario + "];"
    }

    public Integer getCodigo() {
        return codigo;
    }

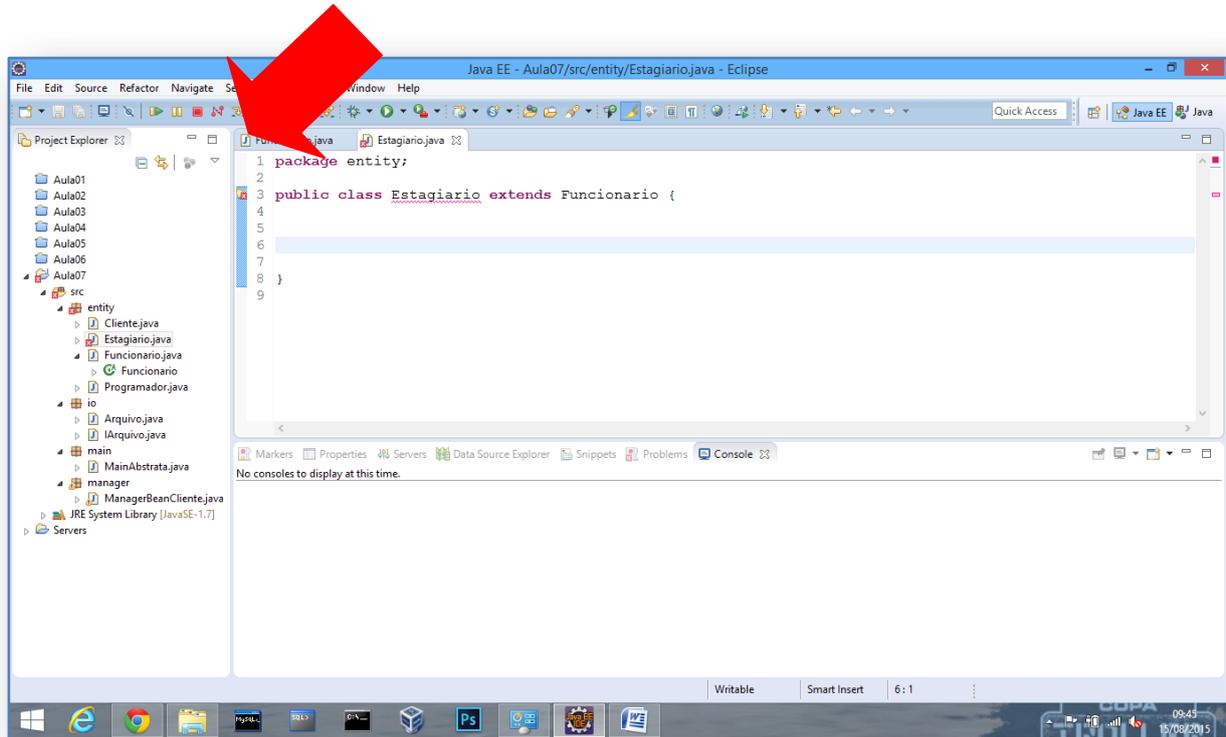
    public void setCodigo(Integer codigo) {
        this.codigo = codigo;
    }
}
```

```
}  
public String getNome() {  
    return nome;  
}  
public void setNome(String nome) {  
    this.nome = nome;  
}  
public String getSexo() {  
    return sexo;  
}  
public void setSexo(String sexo) {  
    this.sexo = sexo;  
}  
public Double getSalario() {  
    return salario;  
}  
public void setSalario(Double salario) {  
    this.salario = salario;  
}
```

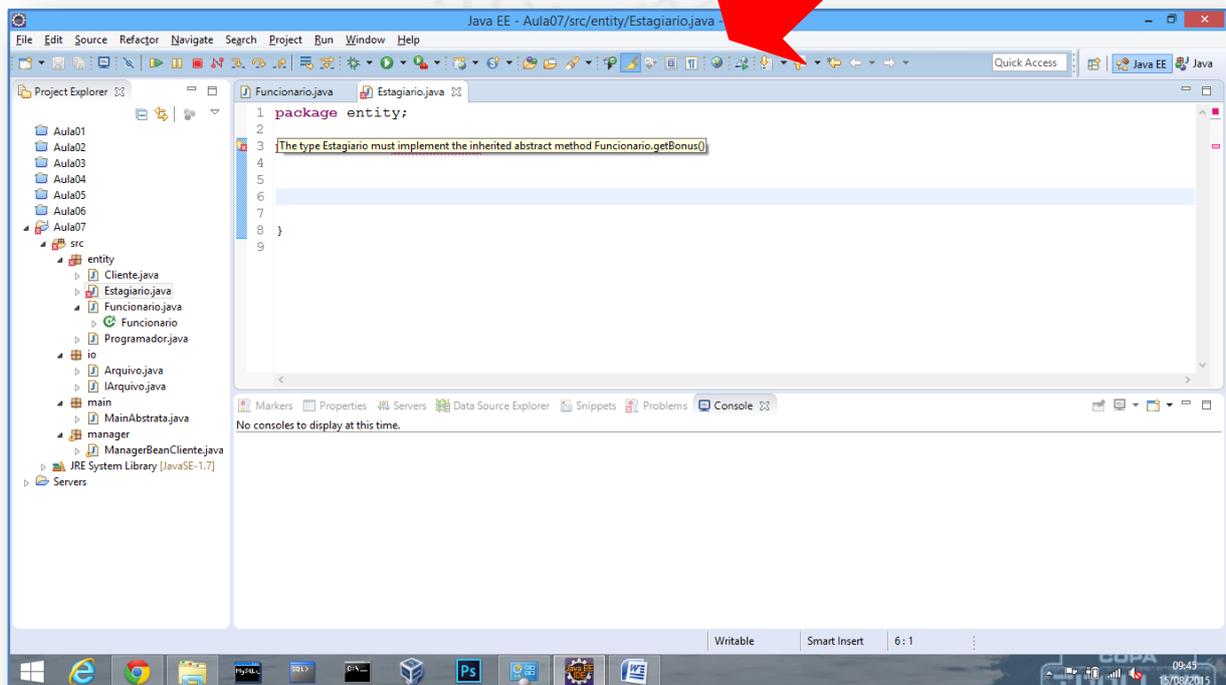
Criar também o método abstrato `getBonus` que será implementado na classe de herança e o `getMessage` que retornará uma mensagem específica.

```
public abstract Double getBonus();  
  
public String getMessage(){  
    return "Classe Abstrata";  
}  
}
```

Criar a classe `Estagiario` que herdará através da palavra "extends" da classe `Funcionario`, implementará o método abstrato "getBonus" que estava na classe `Funcionario` clicando na lâmpada ao lado do nome da classe. Nessa classe, o `Estagiario` terá um bônus de 200, conforme retorno.



Será exibida uma mensagem pedindo para o método ser implementado... Só clicar!



```
package entity;
```

```
public class Estagiario extends Funcionario {
```

Haverá uma sobrescrita no método getBonus.

```
    @Override
    public Double getBonus() {
        return 200.;
    }

    public String getMessage() {
        return "Sub Classe Estagiario";
    }
}
```

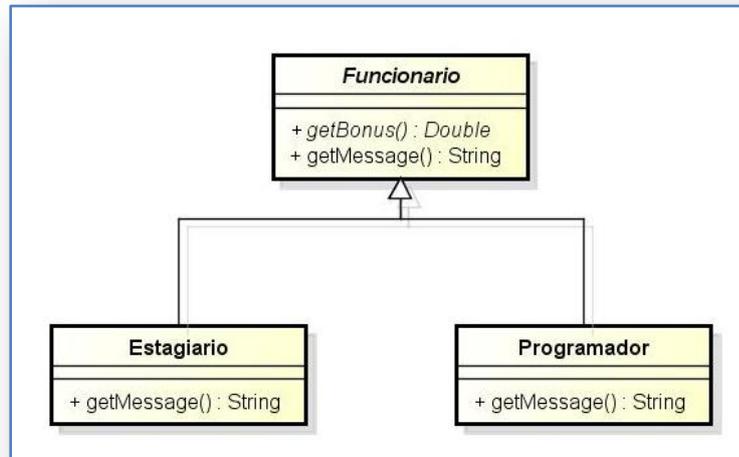
Criar uma outra classe chamada Programador. Fazer o mesmo com essa classe, herdar da classe Funcionario e implementar o método, colocando um corpo, o retorno no "getBonus" de 1500. Isto é, o Programador terá um bônus de 1500.

```
package entity;
```

```
public class Programador extends Funcionario {
```

```
    @Override
    public Double getBonus() {
        return 1500.;
    }

    public String getMessage() {
        return "Sub Classe Programador";
    }
}
```



Criar uma classe chamada MainAbstrata para testar. Criar um objeto para Funcionario e sobrescrever o método getBonus passando novo valor para ele. Onde Funcionário terá um bônus de 100. Passar através do set: o nome e o salário. Na impressão pedir para imprimir o nome desse funcionário e o resultado da soma do salário com o bônus.

Criamos novo Funcionário e instanciamos o Estagiario, passamos seus dados (código, nome e salário)

```

package main;

import entity.Estagiario;
import entity.Funcionario;

public class MainAbstrata {

    public static void main(String[] args) {
        Funcionario f = new Funcionario(){
            public Double getBonus(){
                return 100.;
            }
        };

        f.setNome("joao");
        f.setSalario(10000.);
        System.out.println("Nome: " + f.getNome() + ", " +
"Salário Final: R$ " + (f.getSalario() + f.getBonus()));
    }
}
  
```



```
Funcionario f1= new Estagiario();
    f1.setCodigo(10);
    f1.setNome("carlos");
    f1.setSalario(2000.);

    System.out.println("Bonus do estagiário: R$ " +
f1.getBonus());
    System.out.println("Mensagem: " + f1.getMessage());
}}
```

No console...

```
Nome: joao, Salário Final: R$ 10100.0
Bonus do estagiario: R$ 200.0
Mensagem: Sub Classe Estagiario
```

