TI-220 Java Orientado a Objetos

ANTONIO CARVALHO - TREINAMENTOS

Java Orientação a Objetos

Dependências

Tipos de acesso

Acesso da classe, significa :

- Instanciar um objeto daquela classe
- Herdar a classe
- Acessar algum método ou variável da classe

Estes tipos de acesso também são usados para reaproveitar código (quando usados com a finalidade de acessar métodos localizados em outra classe)

O acesso a classe pode ser resumido em dois tipos de acessos, são eles :

- **Associação** (agregação ou composição)
- **Herança**

Agregação

Agregação ocorre quando uma classe possui uma variável do tipo de outra classe, seja como variável estática ou não, seja de instância ou local.

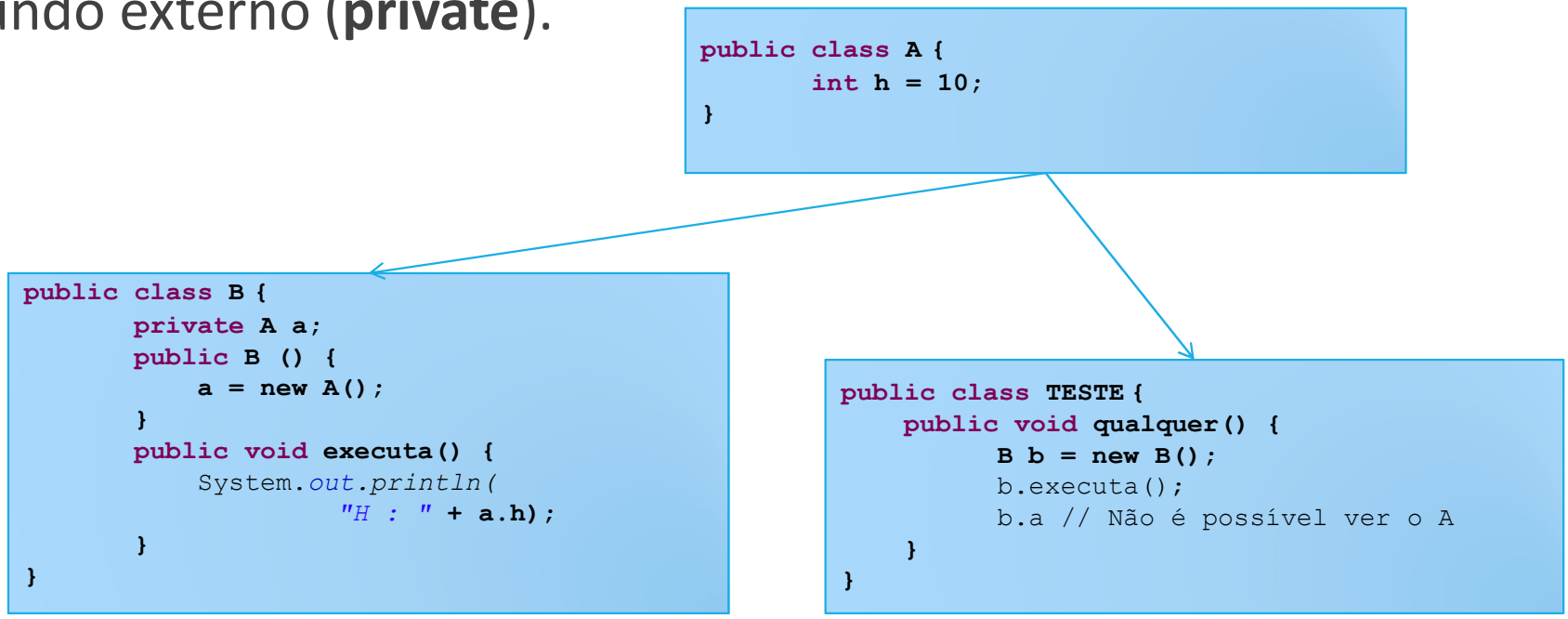
```
public class A {
        public String h = "H";
        private String j = "J";

        public String getJ() {return j;}
        public void setJ(String j) {
                this.j = j;
        }
}
```

```
public class B {
        A a;
}
```

```
public class B {
    public void executa( A a ) {
        System.out.println("H : " + a.h);
        System.out.println("J : " + a.getJ());
}
```

Composição

Composição ocorre quando uma classe possui uma variável do tipo de outra classe, sendo esta variável de instância e não acessível ao mundo externo (**private**).



Agregação e Composição (diferenças)

Segundo Bezerra (2015), “A agregação representa relacionamentos que são formados entre objetos durante a execução do sistema”, e “Na composição os objetos parte pertencem a um único todo. Por esta razão, a composição é também denominada agregação não compartilhada”

As diferenças básicas entre agregação e composição são sutis em relação ao código. A agregação permite que o objeto seja acessível externamente, enquanto que na composição o objeto não pode ser compartilhado com o mundo externo.

Herança

Herança, serve para transmitir os membros de uma classe para outra classe.

Através da **herança** uma classe mais **genérica** pode ser **extendida** para se tornar uma classe mais **especialista**

- A classe mais **genérica** é conhecida como classe **pai** ou **superclasse** e a classe mais **especialista** é conhecida como classe **filho** ou **subclasse**
- A **subclasse** recebe todos os membros (que não forem **privados**) da **superclasse**

Herança

A declaração da classe que herdará outra classe segue o mesmo principio de declaração de uma classe comum porém, adicionando a palavra chave ***extends***

Sintaxe:

[modificadores] **class** <nome> [**extends** <superclass>] {

}

A classe receberá todos os membros da superclasse que não estão marcados como **private**

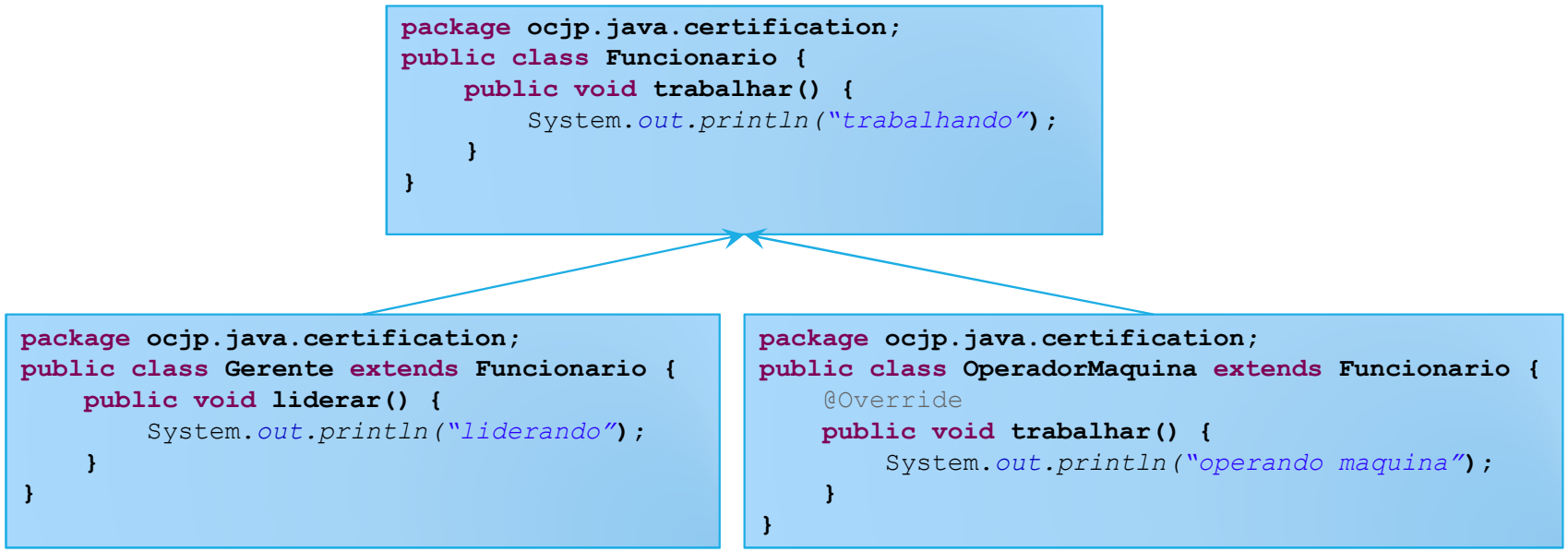
Recebido estes membros, pode-se entender que a classe possui estes membros.

Nota : Não é possível herdar várias classes, da mesma forma que a interface pode herdar várias interfaces.

Nota : Se a classe não for acessível por estar em outro **package**, então nenhum de seus membros será acessível

Herança

Neste exemplo será criada uma classe chamada **funcionario** a qual é mais genérica, e a partir desta classe será criada outra chamada **gerente** a qual terá comportamentos mais específicos.



```
package ocjp.java.certification;
public class Funcionario {
    public void trabalhar() {
        System.out.println("trabalhando");
    }
}
```

```
package ocjp.java.certification;
public class Gerente extends Funcionario {
    public void liderar() {
        System.out.println("liderando");
    }
}
```

```
package ocjp.java.certification;
public class OperadorMaquina extends Funcionario {
    @Override
    public void trabalhar() {
        System.out.println("operando maquina");
    }
}
```

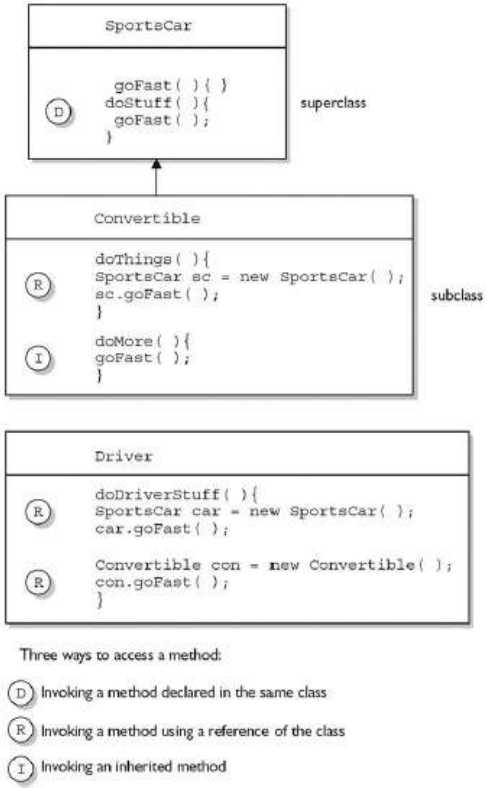
Herança

Perceba que no exemplo anterior foram criadas duas classes **Gerente** e **OperadorMaquina** que herdam a classe **Funcionario**

- A classe **Gerente** recebe por herança o método **trabalhar** que é membro da classe **Funcionario**
- Além disso a classe **Gerente** cria outro método chamado **liderar** especializando mais aquilo que foi recebido da classe **Funcionario**
- A classe **OperadorMaquina** também recebe por herança o método **trabalhar** porém ela modifica o comportamento do método, sobrescrevendo seu código.

Observe a figura 1-2 no livro da Katy Sierra

Herança



Fonte : SCJP Sun Certified Programmer for Java 6 Study Guide (Exam 310065)

Dúvidas

Quiz Time

Acesse o Quiz no endereço :
http://triv.in/107963

Dependência - Exercício

1. Criar exercício de dependência sobre Aeroporto
   - Aeronave
   - Pessoa
   - Piloto
   - Passageiro
   - Aeroporto
2. Desenhe o diagrama de classe dessas entidades, especificando as associações entre elas. Cada classe deve conter pelo menos 3 características e ao menos 1 comportamento, não se esqueça de fazer também a herança.
3. Escreva o código em Java das classes
4. Crie uma função main que instancie ao menos 2 instâncias de cada classe, preenchendo as associações.
5. Desenhe o diagrama de memória do Java (Stack e Heap) conforme as instâncias criadas no exercício 3.