

Programação Orientada a Objetos

profº Mauricio Conceição Mario

Conceitos de Orientação a Objetos

- **Herança:** ocorre quando uma classe passa a herdar características (atributos e métodos) definidas em outra classe, especificada como *ancestral* ou *superclasse*. A classe receptora de recursos é denominada subclasse.

Encapsulamento: disponibilização de uma interface pública para manipular os estados e executar as operações de um objeto. Os atributos de um objeto podem ser escondidos de outros objetos por uma interface pública de métodos, de modo a impedir acessos indevidos.

No encapsulamento os métodos podem ser definidos, através dos modificadores de acesso, como:

- `private;`
- `protected;`
- `public;`
- `static.`
- atributo disponível para consulta: método `getnomeatributo()` [accessor method].
- alteração de atributo: `setnomeatributo()` [mutator method].
- Métodos que executam operações ou funções: [workes methods].

Referência `this`: é a referência do próprio objeto, representa a localização em memória na qual o objeto recorrente foi instanciado.

exemplo: `this.objeto = objeto;`

atributo de instância parâmetro

- src
 - Organiza_Cidade
 - Bairro_de_Santos.java
 - Cidade_de_Santos.java
 - Prefeitura.java
 - Região_de_Santos.java
 - Secretaria
 - Matricula.java
 - Quarto_Ciclo_Sl.java
 - Secretaria_Escolar.java
- JRE System Library [jdk1.8.0_45]

```
package Organiza_Cidade;

public class Cidade_de_Santos {

    private String região;
    private int população;

    Cidade_de_Santos( ){

    }

    Cidade_de_Santos(String s){

        if (s == "r")
            System.out.println("Secretaria de Regionais");
        if (s == "b")
            System.out.println("Secretaria de Bairros");
        }

        public void setRegião(String região){
            this.região = região;
        }

        public void setPopulação(int população){
            this.população = população;
        }

        public String getRegião(){
            return região;
        }

        public int getPopulação(){
            return população;
        }
    }
}
```

```
package Organiza_Cidade;

public class Região_de_Santos extends Cidade_de_Santos {
```

```
    Região_de_Santos(){
        super("r");
    }
```

```
    Região_de_Santos(String f)
    {super(f);}
```

```
    private String bairro;
```

```
    public void setBairro(String bairro){
        this.bairro = bairro;
    }
```

```
    public String getBairro(){
        return bairro;
    }
}
```

```
package Organiza_Cidade;
```

```
public class Bairro_de_Santos extends Região_de_Santos {
```

```
    Bairro_de_Santos(){
        super("b");
    }
```

```
    private String rua;
```

```
    public void setRua(String rua){
        this.rua = rua;
    }
```

```
    public String getRua(){
        return rua;
    }
```

```
}
```

```

package Organiza_Cidade;

public class Prefeitura {

    public static void main(String args[]){

        Região_de_Santos r = new Região_de_Santos();
        Bairro_de_Santos b = new Bairro_de_Santos();
        Cidade_de_Santos c = new Cidade_de_Santos();

        /*classe de hierarquia mais inferior dá acesso
        a todos os atributos*/
        b.setBairro("Jardim Radio Clube");
        b.setRegião("Zona Noroeste");
        b.setRua("Álvaro Guimarães");
        b.setPopulação(100000);
        System.out.println("BAIRRO: " + b.getBairro());
        System.out.println("REGIÃO: " + b.getRegião());
        System.out.println("RUA: " + b.getRua());
        System.out.println("POPULAÇÃO: " + b.getPopulação());

        /*não dá acesso aos atributos da classe inferior */
        r.setRegião("Zona Leste");
        r.setBairro("Marapé");
        r.setPopulação(99000);
        System.out.println("BAIRRO: " + r.getBairro());
        System.out.println("REGIÃO: " + r.getRegião());
        System.out.println("POPULAÇÃO: " + r.getPopulação());

        c.setRegião("Região Central");
        c.setPopulação(20000);
        System.out.println("REGIÃO: " + c.getRegião());
        System.out.println("POPULAÇÃO: " + c.getPopulação());
    }
}

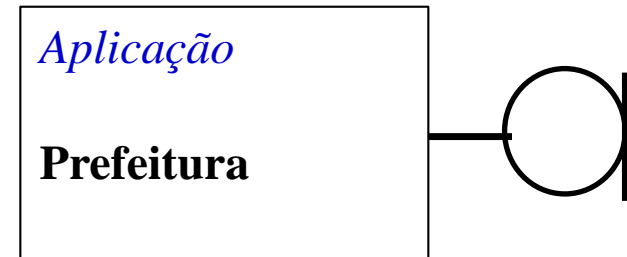
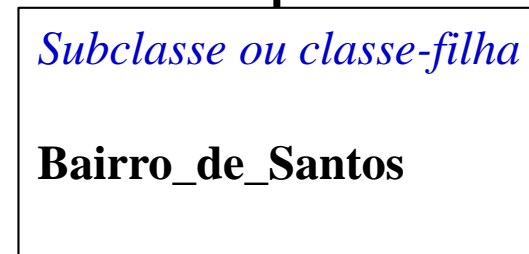
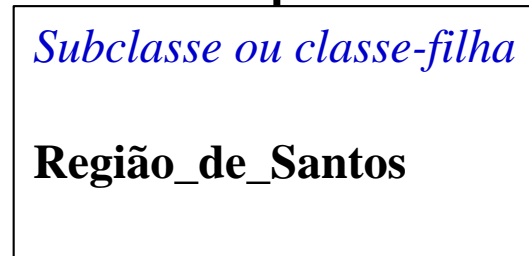
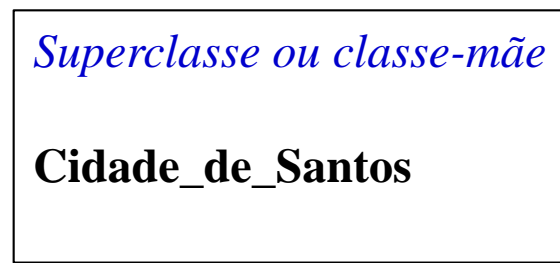
```

<terminated> Prefeitura [Java Application] C:\Arquivos

```

Secretaria de Regionais
Secretaria de Bairros
BAIRRO: Jardim Radio Clube
REGIÃO: Zona Noroeste
RUA: Álvaro Guimarães
POPULAÇÃO: 100000
BAIRRO: Marapé
REGIÃO: Zona Leste
POPULAÇÃO: 99000
REGIÃO: Região Central
POPULAÇÃO: 20000

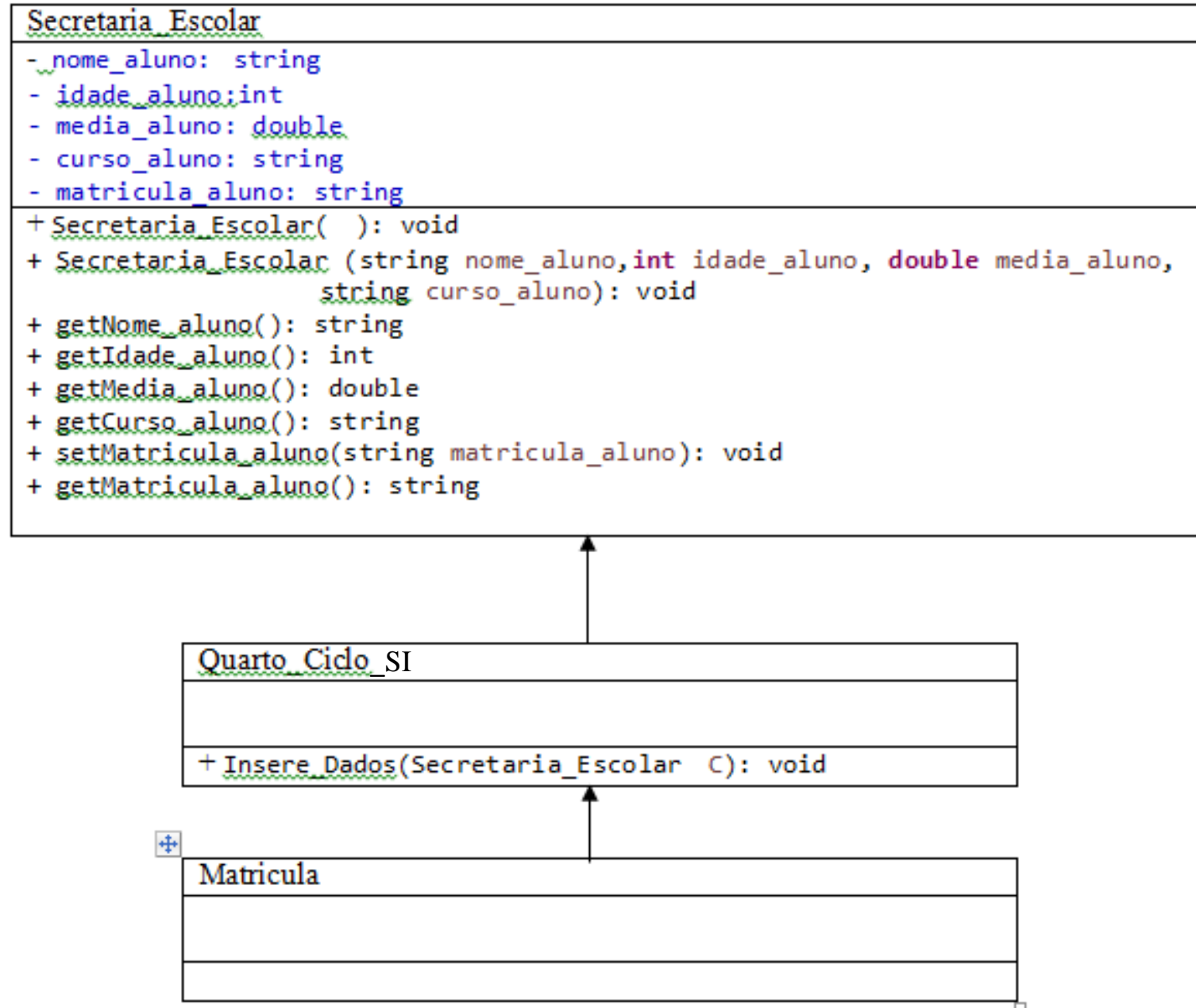
```



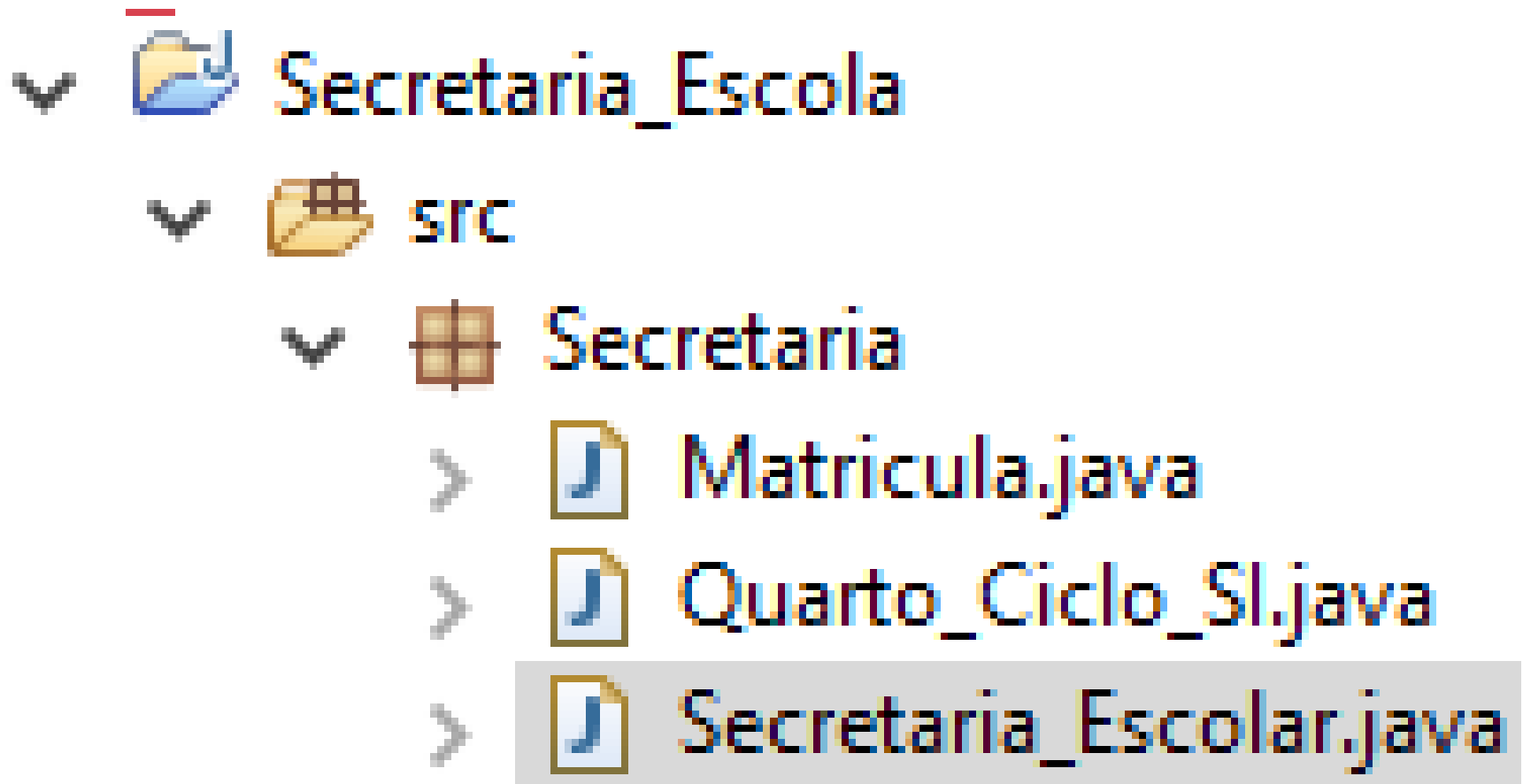
Exercícios:

1. Editar, compilar e executar as classes da aplicação relativa às unidades de uma cidade.
2. Adicionar à aplicação uma classe chamada “Logradouros”, com extensão para Região_de_Santos, com os atributos “tipo_logradouro” e “local_logradouro”, ambos do tipo string.
3. Inserir na superclasse, no método construtor, uma mensagem impressa como foi feito para região e bairro. Acessar o método construtor da superclasse através do método super().
4. Verificar que atributos e métodos a nova classe “Logradouros” tem acesso.
5. Construir, compilar e executar a aplicação Secretaria_Escola a seguir.
6. Fazer uma nova atribuição de matrícula, na classe “Matricula”, utilizando a chamada ao método super().

Diagrama de classes da aplicação



Criar projeto Java no Eclipse considerando o nome do pacote “Secretaria” e os nomes das classes citadas abaixo:



Classe Secretaria_Escolar

```
1 package Secretaria;
2
3 public class Secretaria_Escolar {
4     //atributos
5     private String nome_aluno;
6     private int idade_aluno;
7     private double media_aluno;
8     private String curso_aluno;
9     private String matricula_aluno;
10
11     //método construtor
12     public Secretaria_Escolar (){}
13
14     //método construtor
15     public Secretaria_Escolar (String nome_aluno,int idade_aluno, double media_aluno,
16         String curso_aluno){
17         this.nome_aluno = nome_aluno;
18         this.idade_aluno = idade_aluno;
19         this.media_aluno = media_aluno;
20         this.curso_aluno = curso_aluno;
21     }
22
23     //métodos get( )
24     public String getNome_aluno(){
25         return nome_aluno;}
26 }
```

Classe Secretaria_Escolar



```
26  
27 public int getIdade_aluno(){  
28     return idade_aluno;}  
29  
30 public double getMedia_aluno(){  
31     return media_aluno;}  
32  
33 public String getCurso_aluno(){  
34     return curso_aluno;}  
35  
36 //métodos set( )  
37 public void setMatricula_aluno(String matricula_aluno){  
38     this.matricula_aluno = matricula_aluno;}  
39  
40 public String getMatricula_aluno(){  
41     return matricula_aluno; }  
42 }  
43
```

Classe Quarto_Ciclo_SI

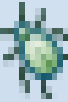
```
PessoaJuridi...  UsaFunciona...  Secretaria_...  Quarto_Cicl...  Matricula.java  » 48

1 package Secretaria;
2
3 public class Quarto_Ciclo_SI extends Secretaria_Escolar {
4
5     public static void main (String args[]) {
6
7         //objeto do tipo Secretaria_Escolar
8         Secretaria_Escolar A = new Secretaria_Escolar();
9         Inere_Dados(A);
10        //objeto do tipo Secretaria_Escolar
11        Secretaria_Escolar B = new Secretaria_Escolar("Maria", 22, 7.7, "Sistemas para Internet");
12        Inere_Dados(B);
13
14    }
15
16    public static void Inere_Dados(Secretaria_Escolar C){
17        System.out.println("Nome do aluno : \t" + C.getNome_aluno());
18        System.out.println("Idade do aluno: \t" + C.getIdade_aluno());
19        System.out.println("Média do aluno: \t" + C.getMedia_aluno());
20        System.out.println("Curso do aluno: \t" + C.getCurso_aluno());
21    }
22 }
23
```

Execução da classe Quarto_Ciclo_SI

 Problems  Javadoc  Declaration

 Console 



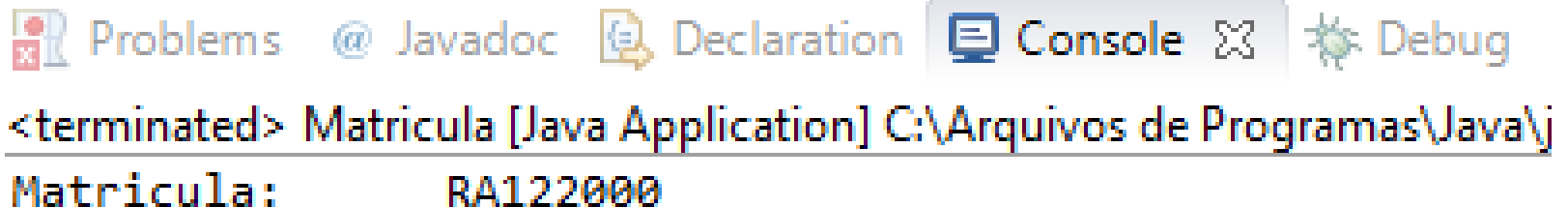
<terminated> Quarto_Ciclo_SI [Java Application] C:\Arquivos de P

```
Nome do aluno :      null
Idade do aluno:      0
Média do aluno:      0.0
Curso do aluno:      null
Nome do aluno :      Maria
Idade do aluno:      22
Média do aluno:      7.7
Curso do aluno:      Sistemas para Internet
```

Classe Matricula

```
1 package Secretaria;  
2  
3 public class Matricula extends Quarto_Ciclo_SI{  
4 public static void main(String args[]){  
5  
6     //objeto do tipo Quarto_Ciclo  
7     Quarto_Ciclo_SI E = new Quarto_Ciclo_SI();  
8     E.setMatricula_aluno("RA122000");  
9     System.out.println("Matricula: \t" + E.getMatricula_aluno() );  
10 }  
11  
12 }  
13
```

Execução da classe Matricula



The screenshot shows the bottom of a Java IDE window. The 'Console' tab is active, displaying the output of the program. The text '<terminated>' indicates the program has finished execution. Below this, the output shows 'Matricula: RA122000', which matches the value set in the code above. The IDE interface includes tabs for 'Problems', 'Javadoc', 'Declaration', 'Console', and 'Debug'.

```
<terminated> Matricula [Java Application] C:\Arquivos de Programas\Java\j  
Matricula: RA122000
```