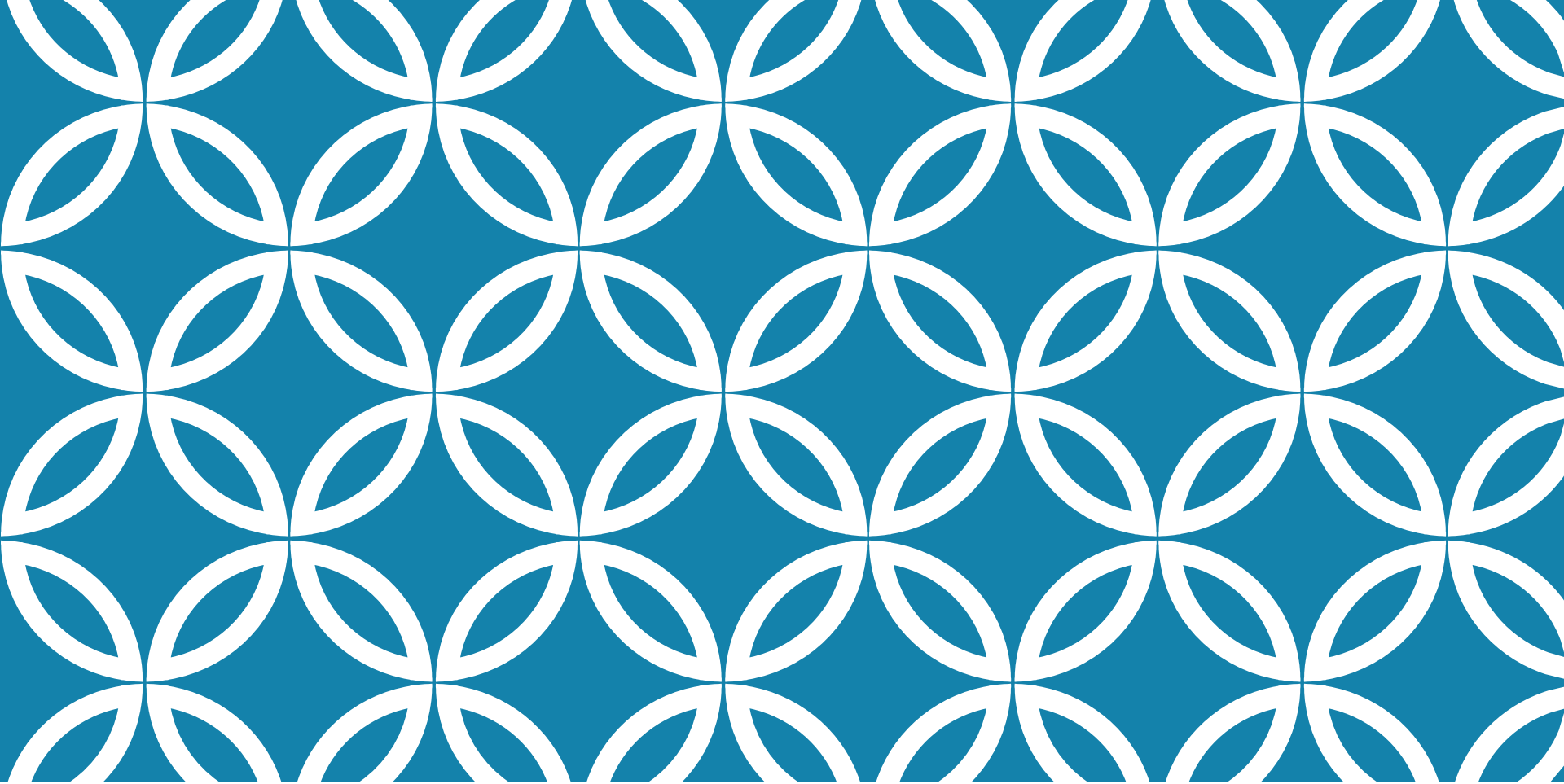


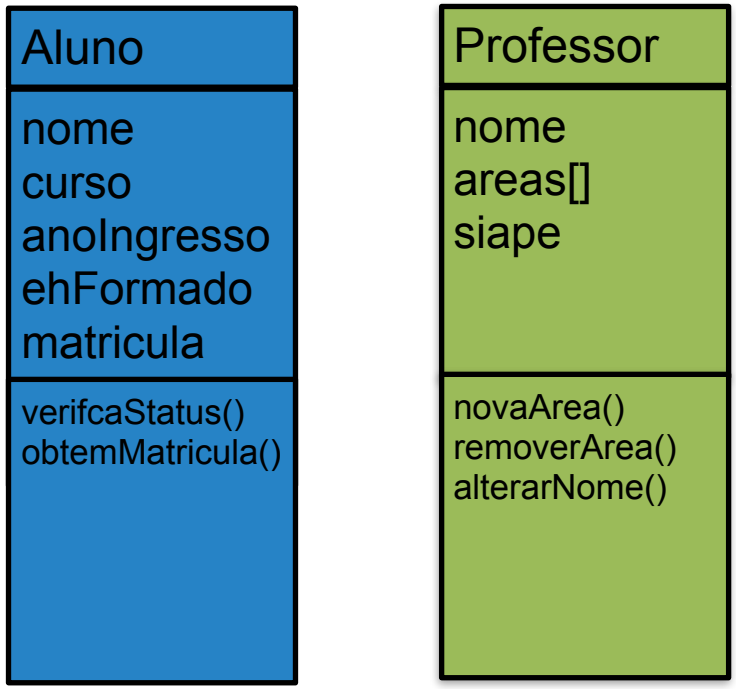
Programação II

Rafael Vieira Coelho

Relembrando…

Tarefas

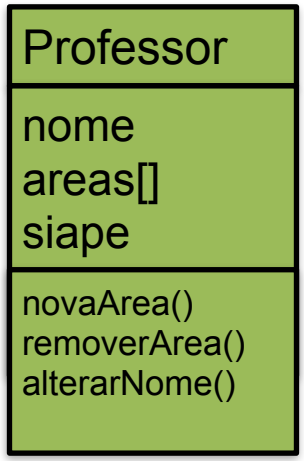
1) Otimize as classes Aluno e Professor adicionando seus métodos get/set, tornando seus atributos privados e métodos públicos. E crie construtores para estas classes.



Relembrando…

```java
public class Professor {

    private String nome;
    private String areas[];
    private long siape;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String[] getAreas() {
        return areas;
    }

    public void setAreas(String[] areas) {
        this.areas = areas;
    }

    public long getSiape() {
        return siape;
    }

    public void setSiape(long siape) {
        this.siape = siape;
    }
```

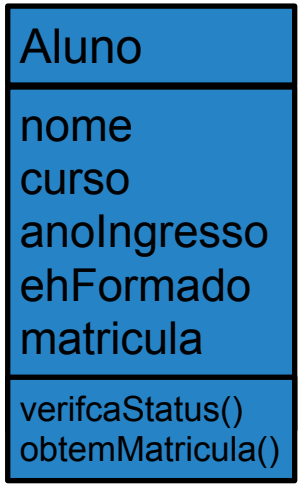
```java
public Professor(String nome, long siape) {
    this.nome = nome;
    this.siape = siape;
}

public Professor(String nome, long siape, String[] areas) {
    this.nome = nome;
    this.areas = areas;
    this.siape = siape;
}

public Professor(String nome, long siape, int numeroAreas) {
    this.nome = nome;
    this.siape = siape;
    this.areas = new String[numeroAreas];
}
```



Relembrando…

```java
public class Aluno {

    private String nome;
    private Curso curso;
    private int anoIngresso;
    private boolean ehFormado;
    private long matricula;

    public Aluno() {

    }

    Aluno(String nome) {
        this.nome = nome;
    }

    public Aluno(String nome, Curso curso, int anoIngresso, long matricula) {
        this.nome = nome;
        this.curso = curso;
        this.anoIngresso = anoIngresso;
        this.matricula = matricula;
        this.ehFormado = false;
    }

    public Aluno(String nome, Curso curso, int anoIngresso, boolean ehFormado,
            long matricula) {
        this.nome = nome;
        this.curso = curso;
        this.anoIngresso = anoIngresso;
        this.ehFormado = ehFormado;
        this.matricula = matricula;
    }
```

```java
public Curso getCurso() {
    return curso;
}

public void setCurso(Curso curso) {
    this.curso = curso;
}

public int getAnoIngresso() {
    return anoIngresso;
}

public void setAnoIngresso(int anoIngresso) {
    this.anoIngresso = anoIngresso;
}

public boolean isEhFormado() {
    return ehFormado;
}

public void setEhFormado(boolean ehFormado) {
    this.ehFormado = ehFormado;
}

public long getMatricula() {
    return matricula;
```



Relembrando…

Tarefas

2) Complemente a classe ProgramaPrincipal na qual é dada uma série de opções para o usuário e que use as classes criadas nos exercícios da aula passada. Inicialmente, o usuário deve definir se é um aluno, professor ou do setor de ensino. Dependendo disto, as suas próximas opções serão diferentes.

Relembrando…

```
11  public class ProgramaPrincipal {
12
13      public static void main(String[] args) throws IOException {
14          BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
15          Aluno[] alunos = new Aluno[1000];
16          SetorEnsino ensino = new SetorEnsino("Pâmela Perini", "Vitor Valente");
17          int opcao = 4;
18
19          do {
20              opcao = menu("MENU 1: \n [1] Aluno \n [2] Professor \n [3] Setor de Ensino \n [4] Sair", br);
21              switch (opcao) {
22                  case 1:
23                      menu_alunos("MENU 2: \n [1] Ver Cursos [2] Ver notas",
24                              ensino,
25                              alunos,
26                              br);
27                      break;
28                  case 2:
29                      System.out.println("Qual o seu número de siape, professor?");
30                      int siape = Integer.parseInt(br.readLine());
31                      int posicao_professor = login_professor(siape, ensino, br);
32
33                      if (posicao_professor != -1) {
34                          menu_professor("MENU 2: \n [1] Dar Notas de uma disciplina [2] Alterar uma nota [3] Adiciona
35                                  posicao_professor,
36                                  ensino,
37                                  br);
38                      } else {
39                          System.err.println("Nome inválido.");
40                      }
41                      break;
42                  case 3:
43                      menu_ensino("MENU 2: \n [1] Cadastrar Aluno [2] Cadastrar Curso [3] Adicionar Disciplina ao Curso
44                              ensino,
45                              alunos,
46                              br);
47              }
48          } while (opcao != 4);
```

Relembrando…

```
51    private static int menu(String opcoes, BufferedReader br) throws IOException {
52        System.out.println(opcoes);
53        String texto = br.readLine();
54
55        int opcao = Integer.parseInt(texto);
56        return opcao;
57    }
```

Relembrando…

```
59      ////////////////// ALUNO ////////////////////////////////////////
60      private static void menu_alunos(String opcoes, SetorEnsino ensino, Aluno alunos[], BufferedReader br)
61          int opcao = menu(opcoes, br);
62
63          switch (opcao) {
64              case 1:
65                  ver_cursos(ensino);
66                  break;
67              case 2:
68                  System.out.println("Qual a sua matrícula, caro discente?");
69                  long matricula = Long.parseLong(br.readLine());
70
71                  ver_notas(ensino, alunos, matricula);
72                  break;
73          }
74      }
```

```java
    ////////////////// ENSINO ////////////////////////////////////////////
    private static void menu_ensino(String opcoes, SetorEnsino ensino, Aluno[] alunos, BufferedReader br)
        int opcao = menu(opcoes, br);

        switch (opcao) {
            case 1:
                System.out.println("Qual o curso?");
                String nome_curso = br.readLine();
                Curso c = encontra_curso(ensino, nome_curso);

                if (c == null) {
                    System.err.println("Curso não encontrado. Cadastre-o.");
                    c = cadastra_curso(ensino, br);
                }
                System.out.println("Qual o professor?");
                String nome_professor = br.readLine();
                Professor p = encontra_professor(ensino, nome_professor);

                if (p == null) {
                    System.err.println("Professor não encontrado. Cadastre-o.");
                    p = cadastra_professor(br, ensino);
                }
                if (cadastra_disciplina(ensino, c, p, br)) {
                    System.out.println("Disciplina cadastrada com sucesso.");
                } else {
                    System.err.println("O limite de disciplinas foi excedido.");
                }
                break;
            case 2:
                novo_aluno(ensino, alunos, br);
                break;
            case 3:
                cadastra_curso(ensino, br);
                break;
            case 4:
                cadastra_professor(br, ensino);
                break;
        }
```

```java
private static Aluno cadastra_aluno(SetorEnsino ensino, BufferedReader br, Aluno[] alunos)
    Aluno a = cria_aluno(ensino, br);

    for (int i = 0; i < alunos.length; i++) {
        Aluno aluno = alunos[i];

        if (aluno == null) {
            alunos[i] = a;
        }
    }
    return a;
}

private static Aluno cria_aluno(SetorEnsino ensino, BufferedReader br) throws IOException {
    Aluno a = new Aluno();

    System.out.println("Nome:");
    a.setNome(br.readLine());
    System.out.println("Curso:");
    String nome_curso = br.readLine();
    Curso c = encontra_curso(ensino, nome_curso);

    a.setCurso(c);
    System.out.println("Matricula:");
    a.setMatricula(Long.parseLong(br.readLine()));
    System.out.println("Ingresso:");
    a.setAnoIngresso(Integer.parseInt(br.readLine()));
    a.setEhFormado(false);
    return a;
}
```

```
76      private static void novo_aluno(SetorEnsino ensino, Aluno[] alunos, BufferedReader br) throws IOException {
77          Aluno a = cadastra_aluno(ensino, br, alunos);
78
79          cadastra_disciplinas_aluno(br, ensino, a);
80      }
81
82      private static void ver_notas(SetorEnsino ensino, Aluno alunos[], long matricula) {
83          boolean aluno_nao_encontrado = true;
84
85          for (Aluno aluno : alunos) {
86              if (aluno.getMatricula() == matricula) {//aluno matriculado
87                  aluno_nao_encontrado = false;
88                  Curso cursos[] = ensino.getCursos();
89
90                  for (Curso curso : cursos) {
91                      Disciplina disciplinas[] = curso.getDisciplinas();
92
93                      for (Disciplina disciplina : disciplinas) {
94                          Aluno a[] = disciplina.getAlunos();
95                          int i = 0;
96
97                          while (i != a.length && a[i].getMatricula() != matricula) {
98                              i++;
99                          }
100                         float nota = disciplina.getNotas()[i];
101
102                         System.out.println("A nota do aluno " + a[i].getNome() + " é de " + nota + " na discip
103                         break;
104                     }
105                 }
106             }
107         }
108         if (aluno_nao_encontrado) {
109             System.err.println("Aluno não matriculado no sistema.");
110         }
111     }
```

Relembrando…

```
144     ////////////////// PROFESSOR ////////////////////////////////////////
145     private static void menu_professor(String opcoes, int posicao_professor, SetorEnsino ensino, BufferedReader br)
207
208     private static Professor cadastra_professor(BufferedReader br, SetorEnsino ensino) throws IOException {...11 li
219
220     private static Professor encontra_professor(SetorEnsino ensino, String nome_professor) {...8 linhas }
228
229     private static Professor cria_professor(BufferedReader br) throws IOException {...16 linhas }
245
246     private static int login_professor(int siape, SetorEnsino ensino, BufferedReader br) {...8 linhas }
254
255     private static boolean nova_area(int pos_professor, SetorEnsino ensino, String area) {...11 linhas }
266
267     private static boolean remover_area(int pos_professor, SetorEnsino ensino, String area) {...11 linhas }
278
279     private static boolean alterar_nota(SetorEnsino ensino, String nome_disciplina, String nome_curso, String nome_
282
283     private static void dar_notas(SetorEnsino ensino, String disciplina, String nome_curso, BufferedReader br) throw
```

Atributos Estáticos

- No ProgramaPrincipal.java

- Podemos tornar dados que não serão alterados em estáticos e finais.

- Ex:

```
public static final int OPCAO_SAIR = 4;
```

```
17          SetorEnsino ensino = new SetorEnsino("Pâmela Perini", "Vitor Valente");
18          int opcao = 4;
19
20          do {
21              opcao = menu("MENU 1: \n [1] Aluno \n [2] Professor \n [3] Setor de
22              switch (opcao) {
23                  case 1:
24                      menu_alunos("MENU 2: \n [1] Ver Cursos [2] Ver notas",
25                              ensino,
26                              alunos,
27                              br);
28                      break;
29                  case 2:
30                      System.out.println("Qual o seu número de siape, professor?"
31                      int siape = Integer.parseInt(br.readLine());
32                      int posicao_professor = login_professor(siape, professores,
33
34                      if (posicao_professor != -1) {
35                          menu_professor("MENU 2: \n [1] Dar Notas de uma discipl
36                                  posicao_professor,
37                                  br);
38                      } else {
39                          System.err.println("Nome inválido.");
40                      }
41                      break;
42                  case 3:
43                      menu_ensino("MENU 2: \n [1] Cadastrar Aluno [2] Cadastrar Cu
44              }
45          } while (opcao != 4);
46      }
```

```java
public class ProgramaPrincipal {

    private static final BufferedReader br = new BufferedReader(new InputStreamR

    private static final int TOTAL_ALUNOS = 1000;
    private static final int TOTAL_PROFESSORES = 60;

    private static final String DIRETOR_ENSINO = "Pâmela Perini";
    private static final String COORDENADOR_ENSINO = "Vitor Valente";

    private static final int OP_ALUNO = 1;
    private static final int OP_PROFESSOR = 2;
    private static final int OP_ENSINO = 3;
    private static final int OP_SAIR = 4;

    private static final int OP_ALUNO_VER_CURSOS = 1;
    private static final int OP_ALUNO_VER_NOTAS = 2;
    private static final int OP_ALUNO_VOLTAR = 3;

    private static final int OP_PROFESSOR_DAR_NOTAS = 1;
    private static final int OP_PROFESSOR_ALTERAR_NOTA = 2;
    private static final int OP_PROFESSOR_ADICIONAR_AREA = 3;
    private static final int OP_PROFESSOR_REMOVER_AREA = 4;
    private static final int OP_PROFESSOR_VOLTAR = 5;

    private static final int OP_ENSINO_NOVO_ALUNO = 1;
    private static final int OP_ENSINO_NOVO_CURSO = 2;
    private static final int OP_ENSINO_NOVA_DISCIPLINA = 3;
    private static final int OP_ENSINO_NOVO_PROFESSOR = 4;
    private static final int OP_ENSINO_VOLTAR = 5;
```

Método equals

- É o método utilizado para verificar se dois objetos são iguais (tem o mesmo conteúdo).

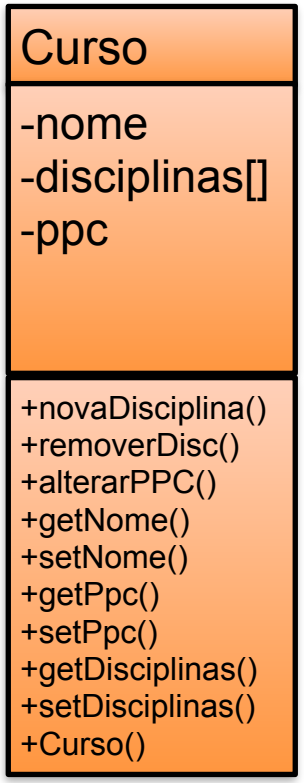
- Por exemplo:

```
String x = “abacaxi”;
String y = “abacaxi”;
if (x == y) {
   System.out.println(“OK 1");
}
if (x.equals(y)) {
   System.out.println(“OK 2”);
}
```

Classe Curso

```
public class Curso {

    private String nome;

    private String ppc;

    private Disciplina disciplinas[];
```

```
75      @Override
        public boolean equals(Object obj) {
77          if (obj != null
78                  && getClass() == obj.getClass()) {
79              final Curso other = (Curso) obj;
80
81              if (this.nome.equals(other.nome)) {
82                  return true;
83              }
84          }
85          return false;
86      }
87
```



Classe Disciplina

```
public class Disciplina {
    private String nome;
    private Aluno alunos[];
    private Professor professor;
    private int ano;
    private float notas[];
```

```
107         @Override
            public boolean equals(Object obj) {
109             if (obj != null
110                     && getClass() == obj.getClass()) {
111                 final Disciplina other = (Disciplina) obj;
112
113                 if (this.professor.equals(other.professor)
114                         && this.nome.equals(other.nome)) {
115                     return true;
116                 }
117             }
118             return false;
119         }
```

Disciplina
- alunos[] - professor - ano - notas[] - nome
+ registrarNota() + novoAluno() + removerAluno() + alterarNota() +getNome() +setNome() +getAno() +setAno() +getNotas() +setNotas() +getAlunos() +setAlunos() +getProfessor() +setProfessor() +Disciplina()

Classe Aluno

public class Aluno {

private long matricula;

private String nome;

```
86      @Override
        public boolean equals(Object obj) {
88          if (obj != null
89                  && getClass() == obj.getClass()) {
90              final Aluno other = (Aluno) obj;
91
92              if (this.matricula == other.matricula
93                      && this.nome.equals(other.nome)) {
94                  return true;
95              }
96          }
97          return false;
98      }
```

Aluno
nome curso anoIngresso ehFormado matricula
verifcaStatus() obtemMatricula() equals()

Método toString

- O método toString, assim como o método equals é definido na classe Object e pode ser reescrito (*override*).

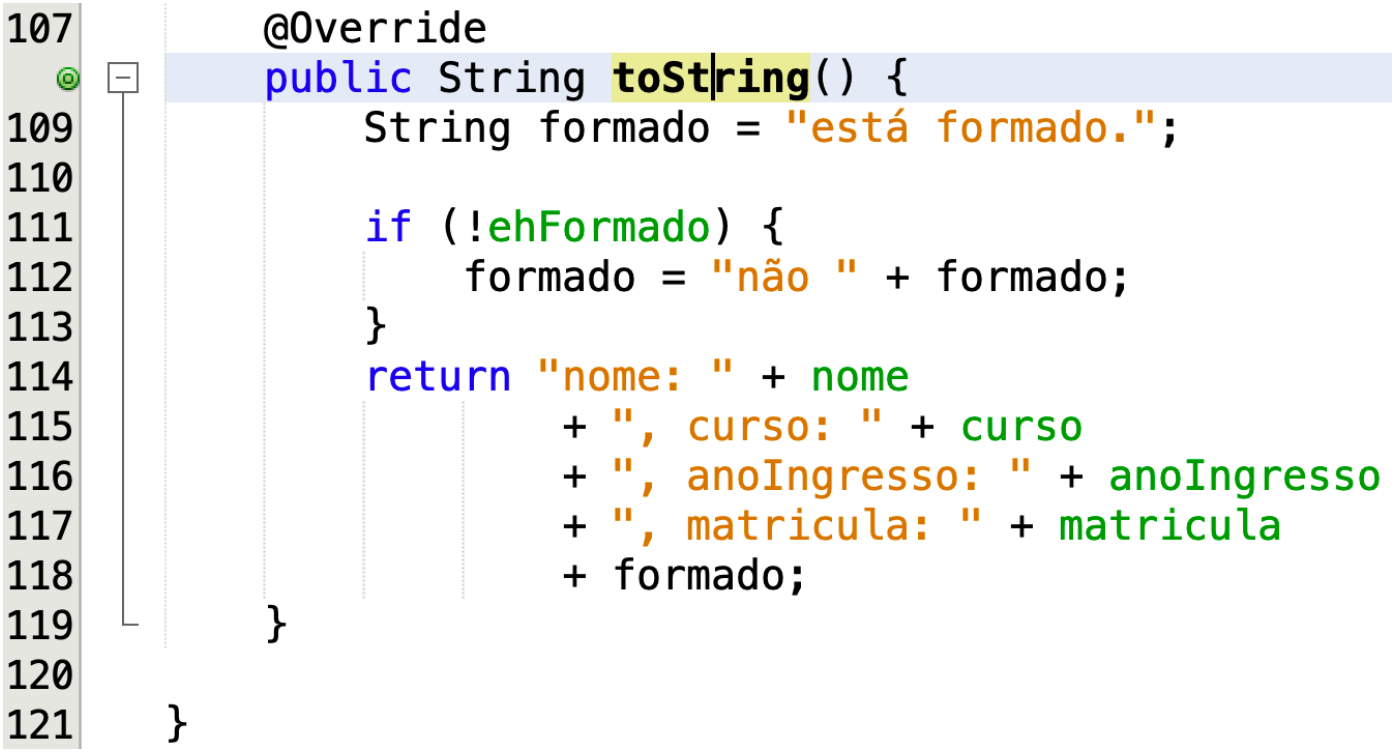
- Ele retorna uma representação textual do objeto.

Classe Aluno

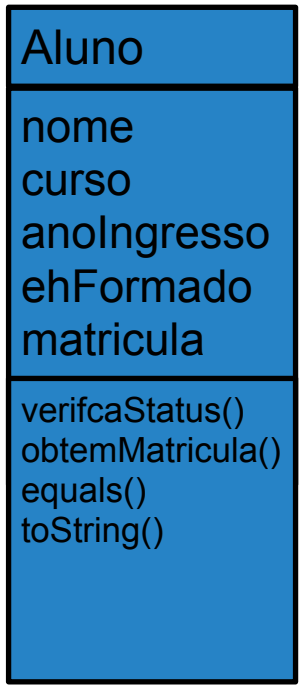
public class Aluno {

private long matricula;

private String nome;



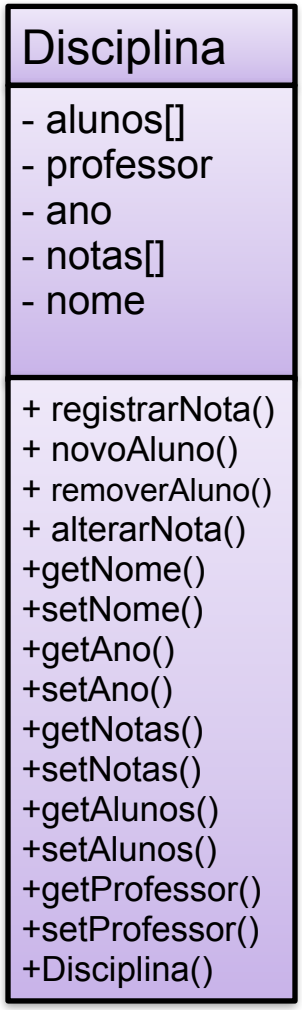
```
107         @Override
            public String toString() {
109             String formado = "está formado.";
110
111             if (!ehFormado) {
112                 formado = "não " + formado;
113             }
114             return "nome: " + nome
115                     + ", curso: " + curso
116                     + ", anoIngresso: " + anoIngresso
117                     + ", matricula: " + matricula
118                     + formado;
119         }
120
121     }
```



Classe Disciplina

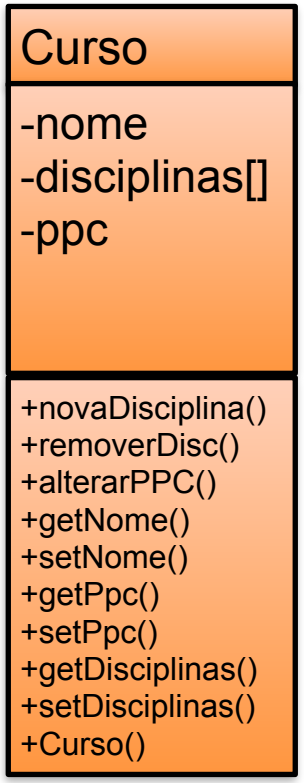
```
public class Disciplina {
    private String nome;
    private Aluno alunos[];
    private Professor professor;
    private int ano;
    private float notas[];
```

```
@Override
public String toString() {
    String notas_alunos = "";

    for (int i = 0; i < alunos.length; i++) {
        Aluno aluno = alunos[i];
        float nota = notas[i];

        notas_alunos += aluno.toString() + " Nota: " + nota + "\n";
    }
    return "professor=" + professor
            + ", nome=" + nome
            + ", ano=" + ano
            + "Notas: \n" + notas_alunos ;
}
```



Classe Curso

```
public class Curso {

    private String nome;

    private String ppc;

    private Disciplina disciplinas[];
```




```
95      @Override
        public String toString() {
97          String disciplina = "";
98
99          for (Disciplina d : disciplinas) {
100             disciplina += d.toString() + "\n";
101         }
102         return "nome: " + nome + ", ppc: " + ppc + ", disciplinas: \n" + disciplina;
103     }
```

Tarefas

1) No ProgramaPrincipal, altere todos os dados fixos para **atributos estáticos** e finais.

2) Crie os métodos **equals** e **toString** das classes Professor e SetorEnsino.

3) Modifique o ProgramaPrincipal para que use os métodos criados no exercício 2 e faça testes para corrigir possíveis problemas (um erro comum é não testar se o objeto ou o array contém valor nulo, gerando assim a exceção NullPointerException).