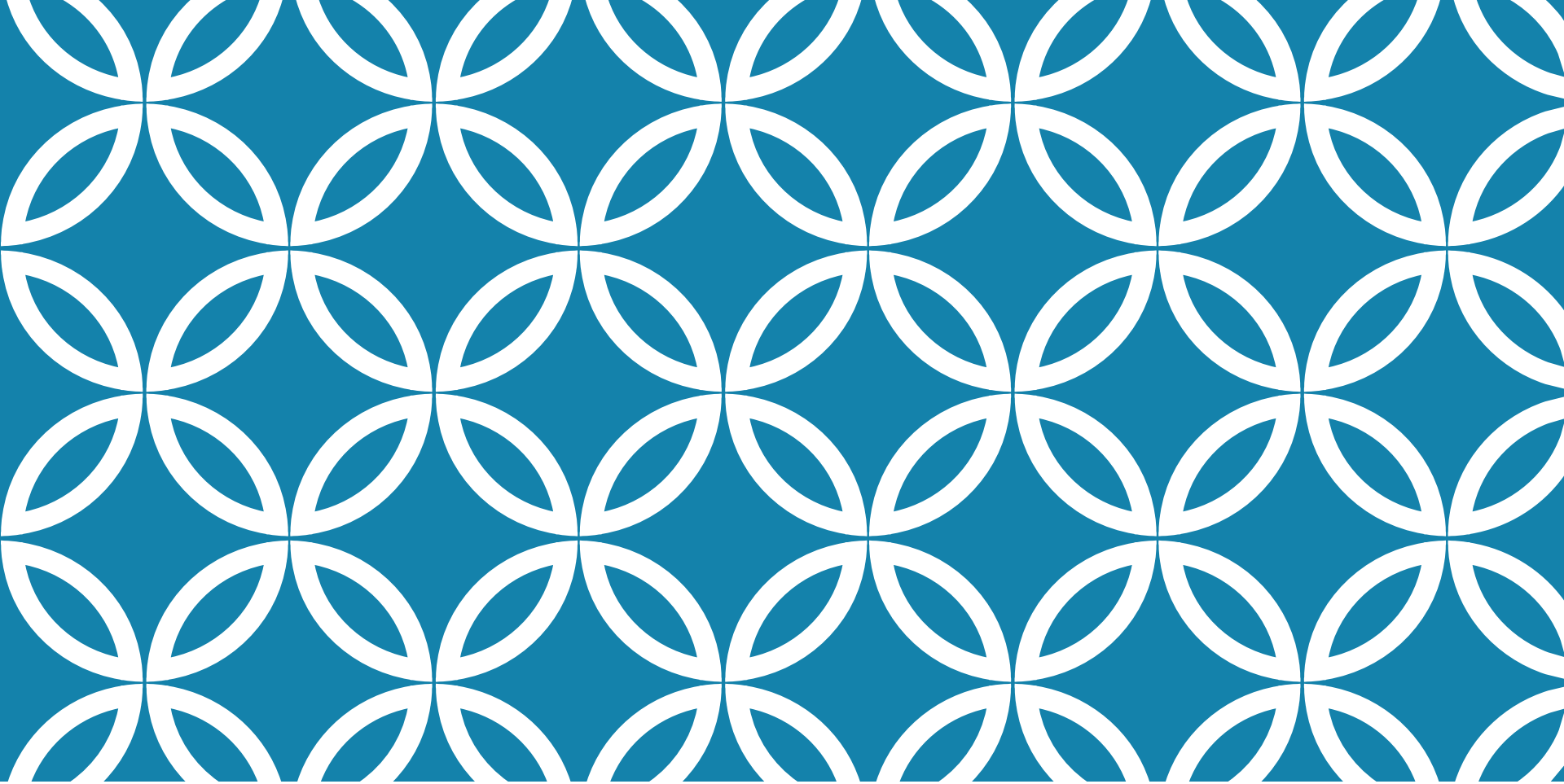
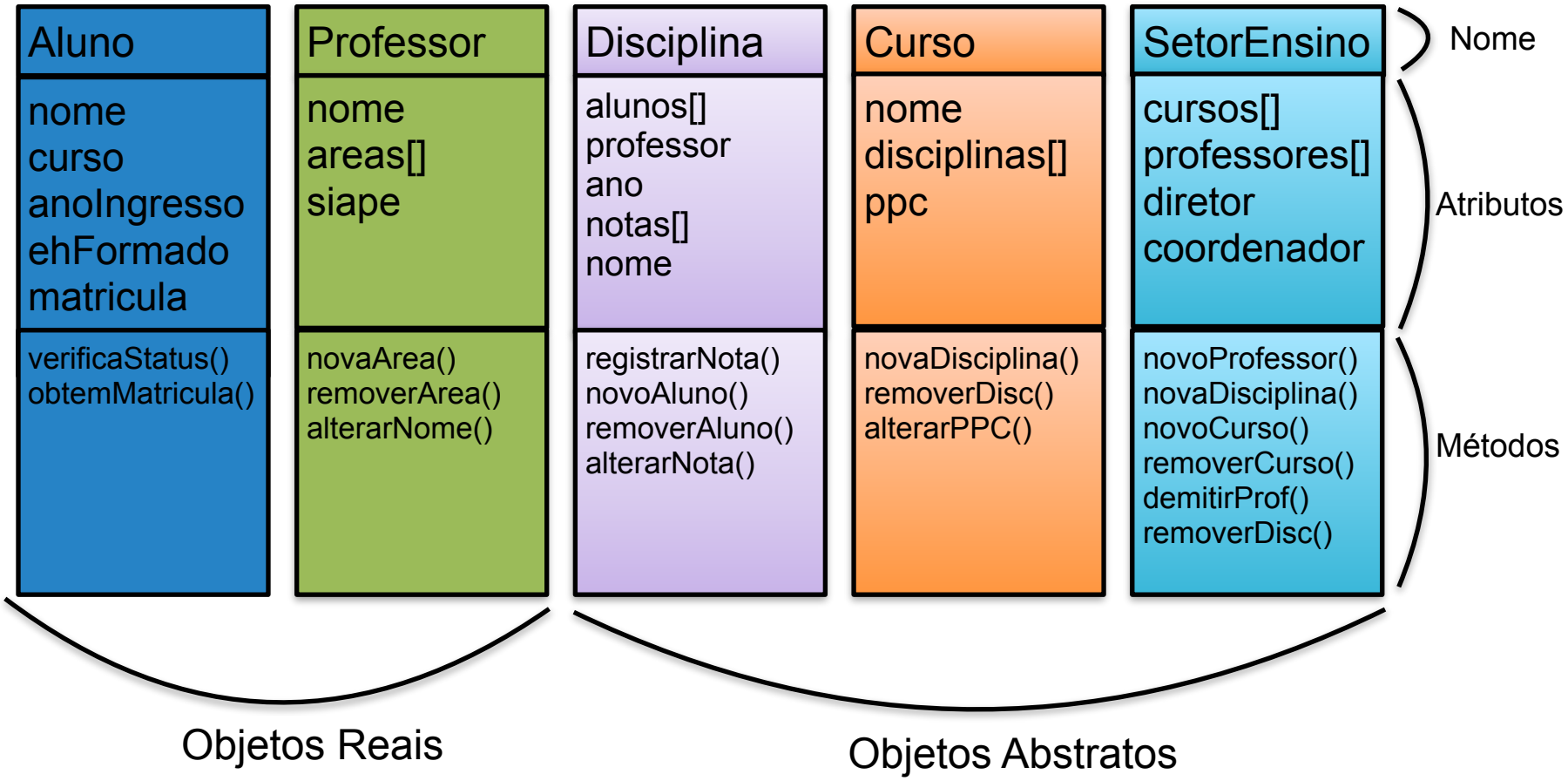


Programação II

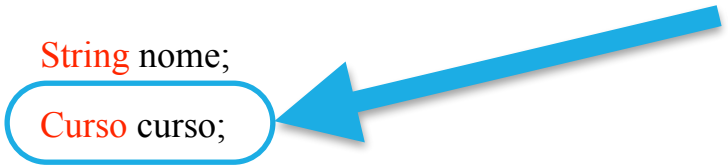
Rafael Vieira Coelho

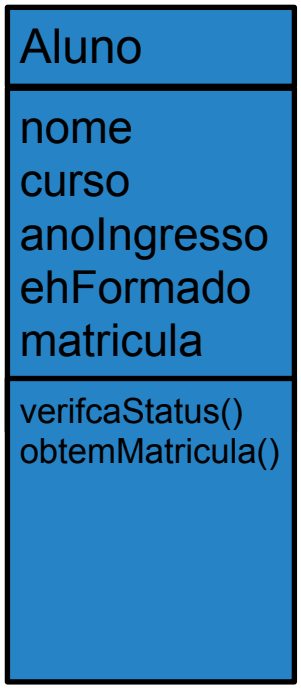
Relembrando…



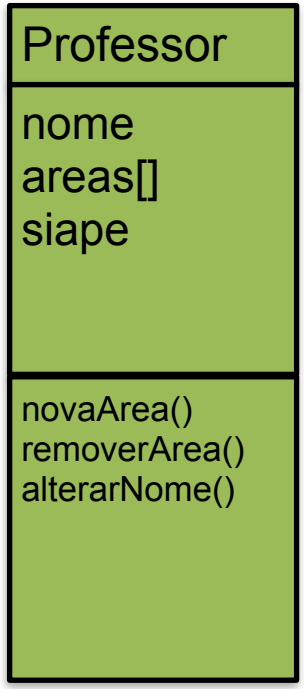
Classe Aluno

```
public class Aluno {
    //Atributos
    String nome;
    Curso curso;
    int anoIngresso;
    boolean ehFormado;
    long matricula;
    //Métodos
    String verificaStatus () {
        if (ehFormado) {
            return "O aluno ainda não completou os créditos";
        }
        return "O aluno entrou no ano" + anoIngresso +  "e se formou";
    }
    long obtemMatricula() {
        return matricula;
    }
}
```





Classe Professor




```
public class Professor {
    //Atributos
    String nome;
    String areas[];
    long siape;
    //Métodos
    void alterarNome (String novoNome) {
        this.nome = novoNome;
    }
    boolean novaArea(String area) {
        for (int i; i < areas.length; i++)
            if (areas[i] == null) {
                areas[i] = area;
                return true;
            }
        return false;
    }
}
```

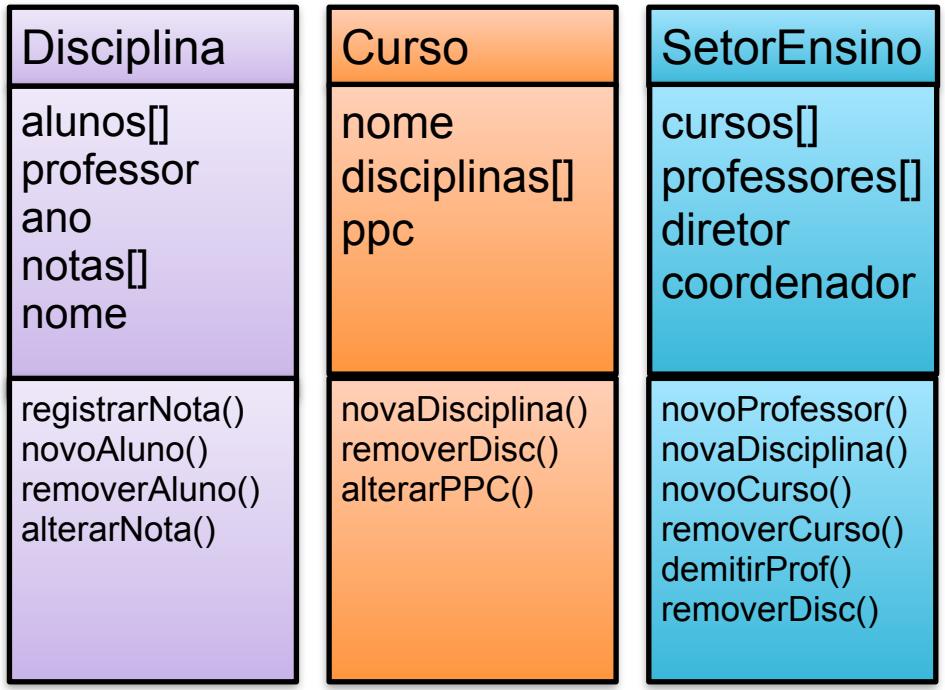
```
boolean removerArea(String area) {
    for (int i; i < areas.length; i++)
        if (areas[i].equals(area)) {
            areas[i] = null;
            return true;
        }
    return false;
}
```

Tarefas da Aula Passada

1) Crie um projeto chamado IFRS e adicione ao projeto as classes dadas como exemplo anteriormente (Aluno, Professor e as de teste)

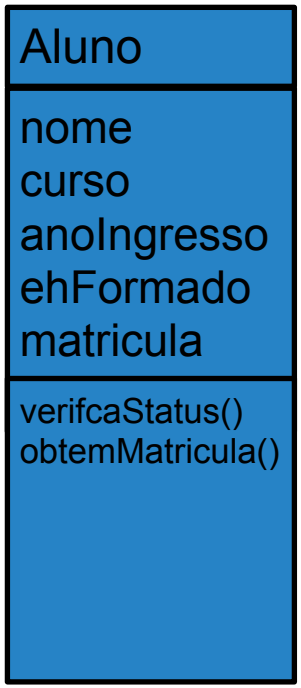
2) Modularize as classes de teste.

3) Implemente as três classes abaixo.

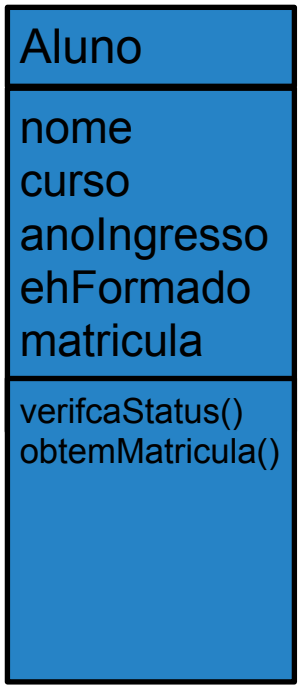


2) Modularize as classes de teste.

```
public class ProgramaTeste1 {
    public static void main(String []args) {
        Aluno maria;

        maria = new Aluno();
        maria.nome = "Maria da Graça Souza";
        maria.curso = "Análise e Desenvolvimento de Sistemas (ADS)";
        maria.anoIngresso = 2013;
        maria.ehFormado = true;
        maria.matricula = 154090;
    }
}
```



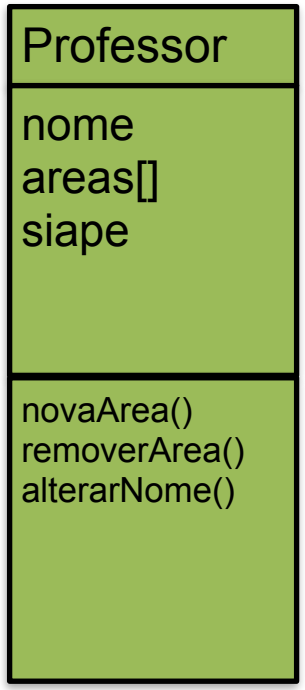
2) Modularize as classes de teste.



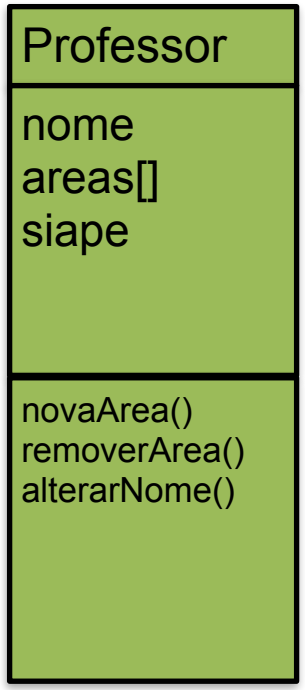

```
public class ProgramaTeste1Modularizado {
    public static void main(String []args) {
        Curso curso = new Curso();
        curso.nome = "Análise e Desenvolvimento de Sistemas (ADS)";
        Aluno maria = criaAluno("Maria da Graça Souza",
        curso,
        2013,
        154090,
        true);
    }
    public static Aluno criaAluno(String nome, Curso curso, int ano, long matricula, boolean formado) {
        Aluno a = new Aluno();
        a.nome = nome;
        a.curso = curso;
        a.anoIngresso = ano;
        a.ehFormado = formado;
        a.matricula = matricula;
        return a;
    }
}
```

2) Modularize as classes de teste.

```
public class ProgramaTeste2 {
    public static void main(String []args) {
        Professor coelho;

        coelho = new Professor();
        coelho.nome = “Rafael Vieira Coelho”;
        coelho.siape = 1804250;
        coelho.areas = new String[3];
        coelho.areas[0] = “Programação de Computadores";
        coelho.areas[1] = “Redes de Computadores”;
        coelho.areas[2] = “Segurança de Sistemas”;
    }
}
```



2) Modularize as classes de teste.




```
public class ProgramaTeste2Modularizado {
    public static void main(String []args) {
        Professor coelho = criaProfessor("Rafael Vieira Coelho",
        1804250,
        3);
        coelho.novaArea("Programação de Computadores");
        coelho.novaArea("Redes de Computadores");
        coelho.novaArea("Segurança de Sistemas");
    }
    public static Professor criaProfessor(String nome, long siape, int numeroAreas) {
        Professor p = new Professor();
        p.nome = nome;
        p.siape = siape;
        p.areas = new String[numeroAreas];
        return p;
    }
}
```

2) Modularize as classes de teste.

```
public class ProgramaTeste3 {
    public static void main(String []args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Escolha [1] para professor"
        + "e [2] para aluno:");
        int opcao = s.nextInt(); s.nextLine();
        if (opcao == 1) {
            Professor p = new Professor();
            System.out.println("Nome:");
            p.nome = s.nextLine();
            System.out.println("SIAPE:");
            p.siape = s.nextLong();
            System.out.println("Quantas áreas?");
            quantAreas = s.nextInt();
            p.areas = new String[quantAreas];
            System.out.println("Informe as áreas:");
            for (int i = 0; i < quantAreas; i++) {
                p.areas[0] = s.nextLine();
            }
        } else {
            Aluno a = new Aluno();
            System.out.println("Nome:");
            a.nome = s.nextLine();
            System.out.println("Curso:");
            Curso c = new Curso();
            c.nome = s.nextLine();
            a.curso = c;
            System.out.println("Matricula:");
            a.matricula = s.nextLong();
            System.out.println("Ingresso:");
            a.anoIngresso = s.nextInt();
            a.ehFormado = false;
        }
    }
}
```

2) Modularize as classes de teste.

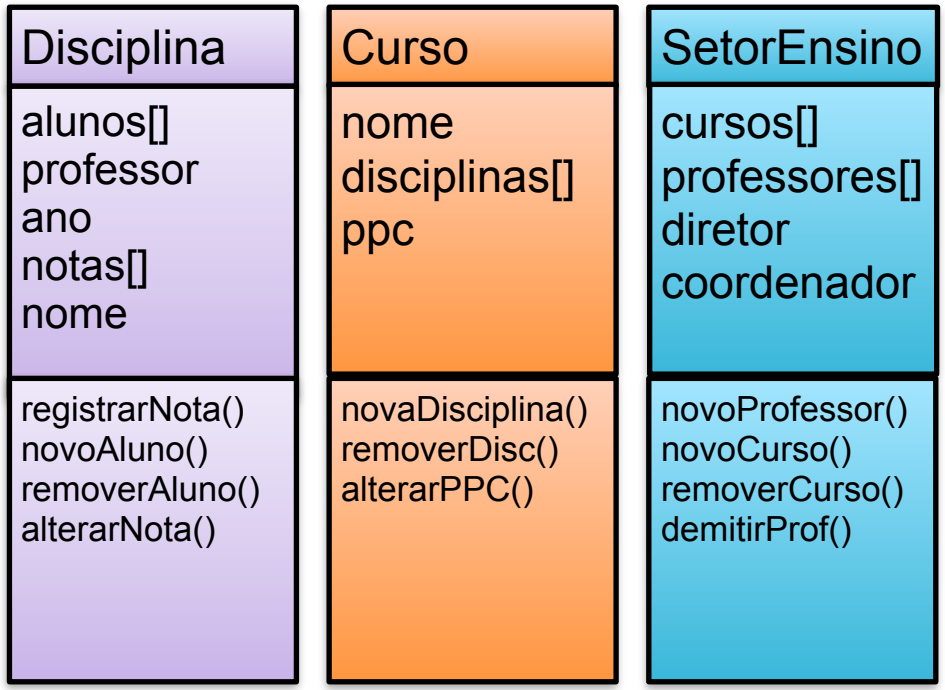
```
public class ProgramaTeste3Modularizado {
    public static void main(String []args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Escolha [1] para professor"
        + "e [2] para aluno:");
        int opcao = s.nextInt(); s.nextLine();
        if (opcao == 1) {
            System.out.println("Nome:");
            String nome = s.nextLine();
            System.out.println("SIAPE:");
            long siape = s.nextLong();
            System.out.println("Quantas áreas?");
            int quantAreas = s.nextInt();
            Professor p = ProgramaTeste2Modularizado.criaProfessor(nome, siape, quantAreas);
            System.out.println("Informe as áreas:");
            for (int i = 0; i < quantAreas; i++) {
                p.novaArea(s.nextLine());
            }
```

2) Modularize as classes de teste.

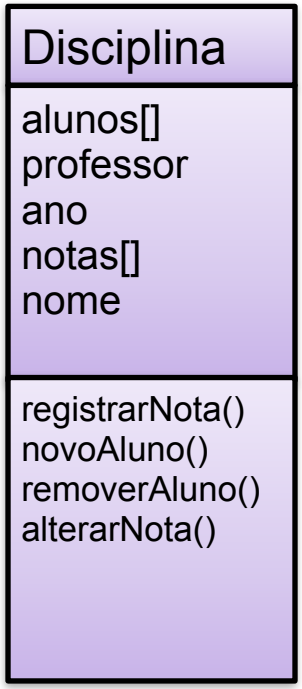
```
        } else {
            System.out.println("Nome:");
            String nome = s.nextLine();
            System.out.println("Curso:");
            Curso curso = new Curso();
            curso.nome = s.nextLine();
            System.out.println("Matricula:");
            long matricula = s.nextLong();
            System.out.println("Ingresso:");
            int anoIngresso = s.nextInt();
            boolean formado = false;
            Aluno a = ProgramaTeste1Modularizado.criaAluno(nome, curso, anoIngresso, matricula, formado);
        }
    }
}
```

Tarefas da Aula Passada

3) Implemente as três classes abaixo.



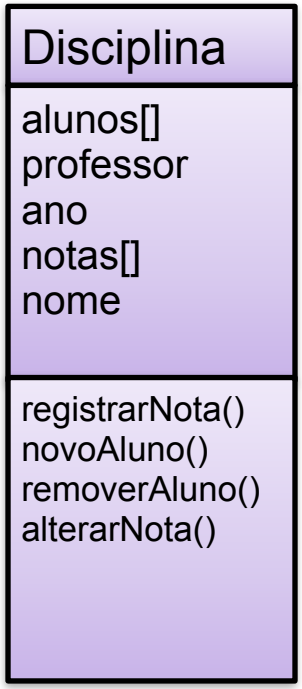
Classe Disciplina




```
public class Disciplina {

    //Atributos
    String nome;
    Aluno alunos[];
    Professor professor;
    int ano;
    float notas[];


    //Métodos
    boolean registrarNota(float nota, String nome) {
        for ( int i = 0; i < notas.length; i++) {
                if (alunos[i] !=null && alunos[i].nome.equals(nome)) {
                    notas[i] = nota;
                    return true;
                }
        }
        return false;
    }
```

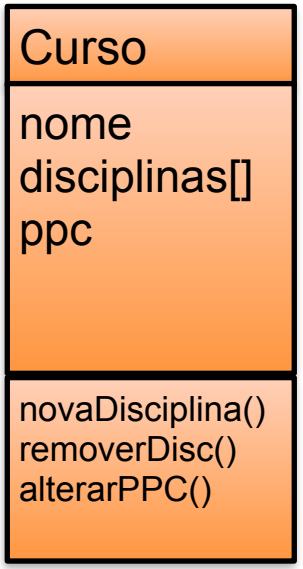
```
    boolean novoAluno(String nome) {
        for ( int i = 0; i < alunos.length; i++) {
                if (alunos[i]==null) {
                    alunos[i] = new Aluno();
                    alunos[i].nome = nome;
                    return true;
                }
        }
        return false;
    }
```

Classe Disciplina




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

    //Métodos
    boolean removerAluno(String nome) {
        for ( int i = 0; i < alunos.length; i++) {
            if (alunos[i] != null && alunos[i].nome.equals(nome)) {
                alunos[i] = null;
                return true;
            }
        }
        return false;
    }

    boolean alterarNota(float nota, String nome) {
        return registrarNota(nota, nome);
    }
}
```

Classe Curso

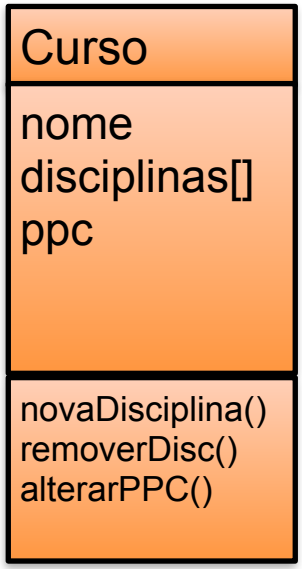



```
public class Curso {

    //Atributos

    String nome;

    String ppc;

    Disciplina disciplinas[];

    //Métodos

    boolean novaDisciplina(String nome, int ano, Professor professor) {

        for ( int i = 0; i < disciplinas.length; i++) {

                if (disciplinas[i] != null) {

                        disciplinas[i] = new Disciplina();

                        disciplinas[i].nome = nome;

                        disciplinas[i].ano = ano;

                        disciplinas[i].professor = professor;

                        return true;

                }

        }

        return false;

    }
```

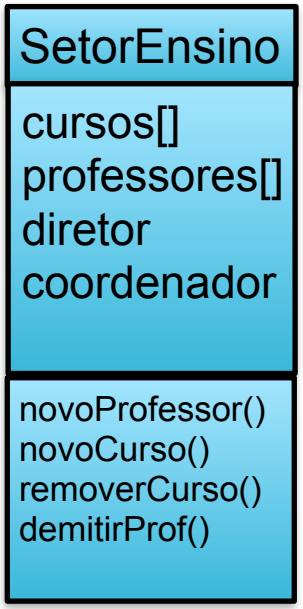
```
    boolean removerDisciplina(String nome) {

        for ( int i = 0; i < disciplinas.length; i++) {

                if (disciplinas[i] != null &&

                disciplinas[i] .nome.equals(nome)) {

                        disciplinas[i] = null;

                        return true;

                }

        }

        return false;

    }
```

Classe Curso

```
public class Curso {

    //Atributos

    String nome;

    String ppc;

    Disciplina disciplinas[];


    //Métodos

    void alterarPPC(String ppc) {

        this.ppc = ppc;

    }
```



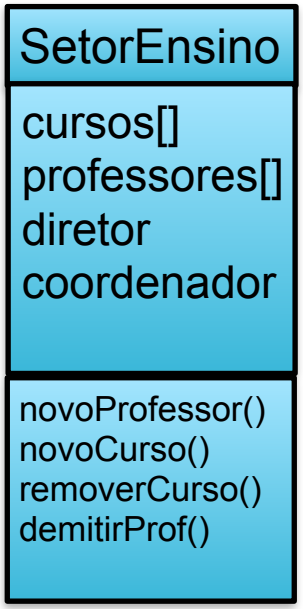
Classe SetorEnsino




```
public class SetorEnsino {
    //Atributos
    Curso cursos[];
    Professor professores[];
    String diretor;
    String coordenador;
    //Métodos
    boolean novoProfessor(String nome, long siape) {
        for ( int i = 0; i < professores.length; i++) {
            if (professores[i] != null) {
                professores[i] = new Professor();
                professores[i].nome = nome;
                professores[i].siape = siape;
                return true;
            }
        }
        return false;
    }
```

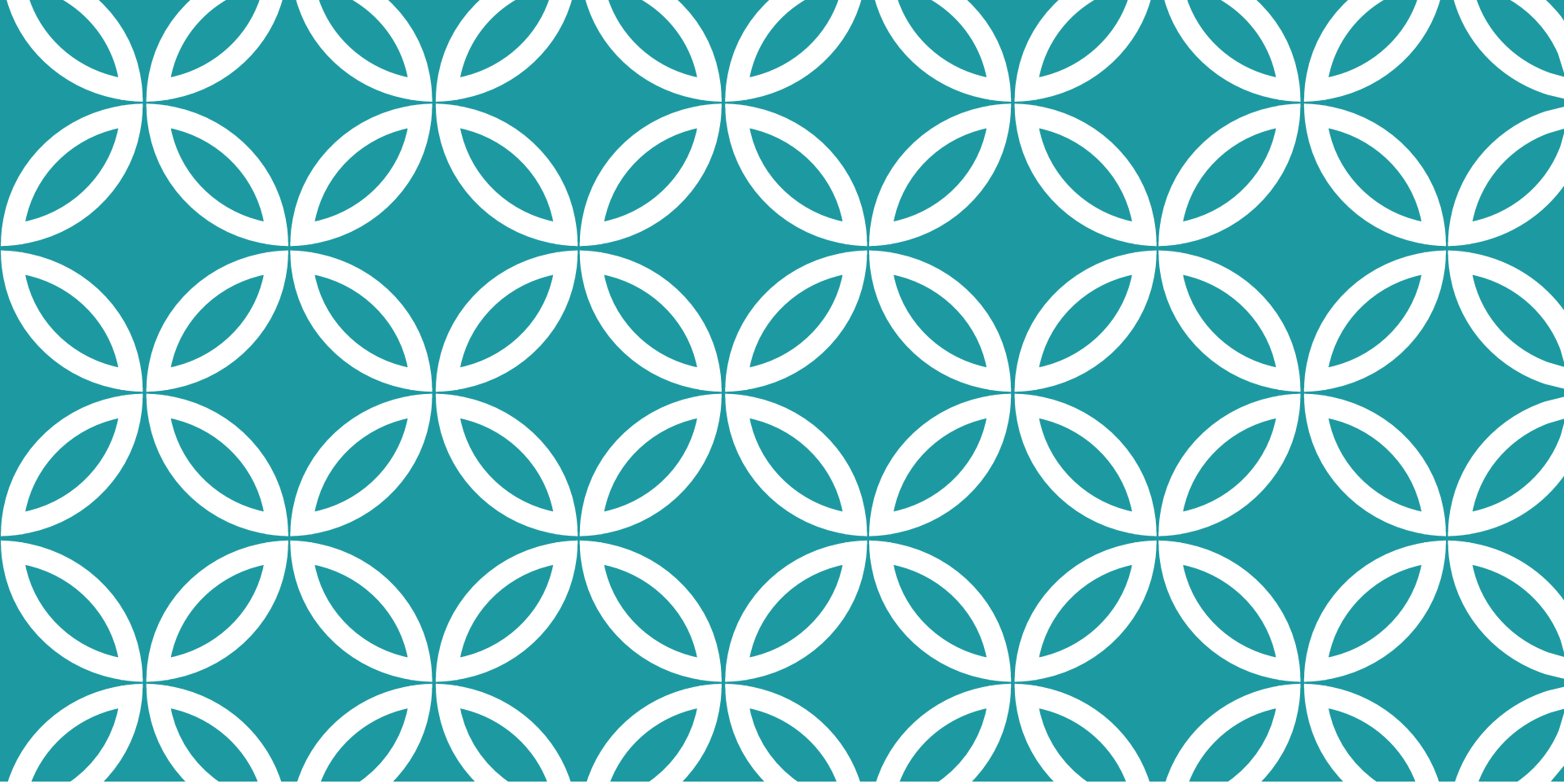
```
boolean demitirProfessor(long siape) {
    for ( int i = 0; i < professores.length; i++) {
        if (professores[i] != null
        && professores[i].siape == siape) {
            professores[i] = null;
            return true;
        }
    }
    return false;
}
```

Classe SetorEnsino




```
public class SetorEnsino {
    //Atributos
    Curso cursos[];
    Professor professores[];
    String diretor;
    String coordenador;
    //Métodos
    boolean novoCurso(String nome, String ppc) {
        for ( int i = 0; i < cursos.length; i++) {
            if (cursos[i] != null) {
                cursos[i] = new Curso();
                cursos[i].nome = nome;
                cursos[i].ppc = ppc;
                return true;
            }
        }
        return false;
    }
```

```
    boolean removerCurso(String nome) {
        for ( int i = 0; i < cursos.length; i++) {
            if (cursos[i] != null &&
            cursos[i].nome.equals(nome)) {
                cursos[i] = null;
                return true;
            }
        }
        return false;
    }
}
```



1. Contrutores e Getters/Setters

Obs: Modificadores

Modifier	Class	Package	Subclass	Global
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

private - acessado apenas pela própria classe

protected - acessado pela classe, classes do pacote e subclasses

[default] - acessado pela classe, classes do pacote

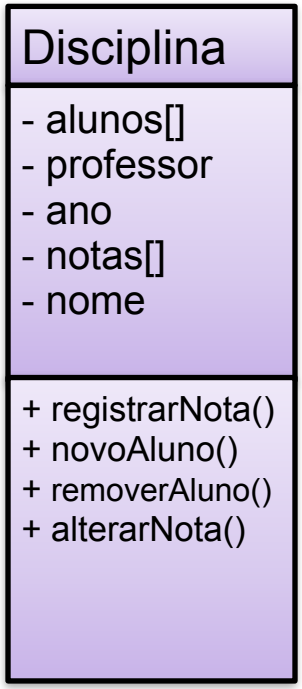
public - acessado por todos

Melhorando as Classes

1. Torne os atributos privados (private) para que somente sejam acessados em sua classe.

2. Torne os seus métodos públicos (public) para que possam ser chamados fora da classe a partir de objetos dessas classes.

Classe Disciplina

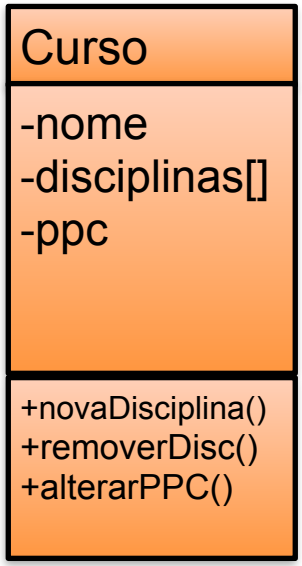



```
public class Disciplina {
    private String nome;
    private Aluno alunos[];
    private Professor professor;
    private int ano;
    private float notas[];

    public boolean registrarNota(float nota, String nome) {
        for ( int i = 0; i < notas.length; i++) {
            if (alunos[i] != null &&
            alunos[i].nome.equals(nome)) {
                notas[i] = nota;
                return true;
            }
        }
        return false;
    }
```

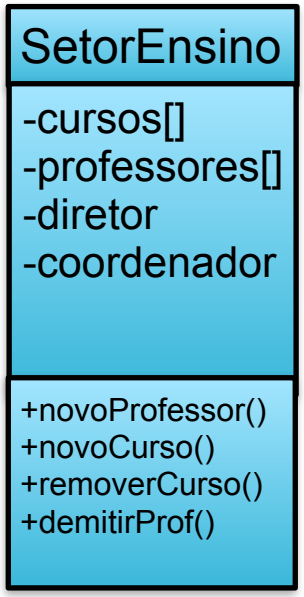
```
    public boolean registrarAluno(String nome) {
        for ( int i = 0; i < alunos.length; i++) {
            if (alunos[i]!=null) {
                alunos[i].nome = nome;
                return true;
            }
        }
        return false;
    }
```

Classe Curso

```
public class Curso {

    private String nome;

    private String ppc;

    private Disciplina disciplinas[];


    public void alterarPPC(String ppc) {

        this.ppc = ppc;

    }
```



Classe SetorEnsino




```
public class SetorEnsino {
    private Curso cursos[];
    private Professor professores[];
    private String diretor;
    private String coordenador;

    public boolean novoProfessor(String nome, long siape) {
        for ( int i = 0; i < professores.length; i++) {
            if (professores[i] != null) {
                professores[i] = new Professor();
                professores[i].nome = nome;
                professores[i].siape = siape;
                return true;
            }
        }
        return false;
    }
```

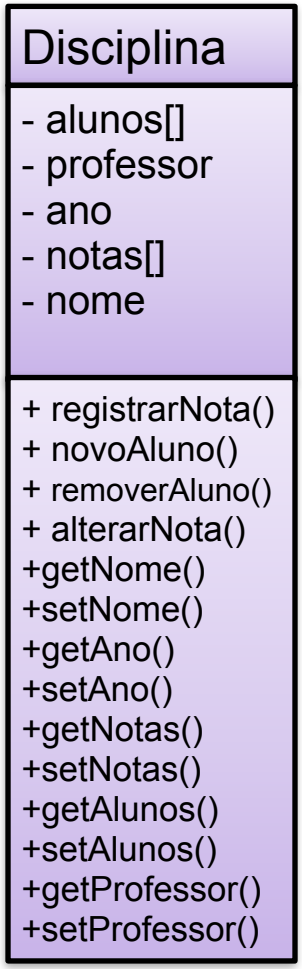
```
public boolean demitirProfessor(long siape) {
    for ( int i = 0; i < professores.length; i++) {
        if (professores[i] != null &&
        professores[i].siape == siape) {
            professores[i] = null;
            return true;
        }
    }
    return false;
```

Melhorando as Classes

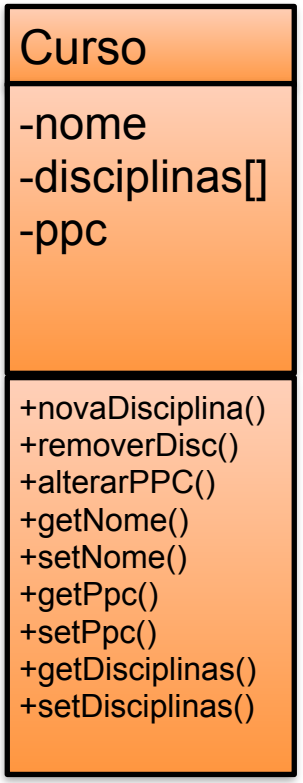
3. Crie pares de métodos públicos para modificar (set) e acessar (get) para cada atributo das classes.

Classe Disciplina

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

    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public int getAno() {
        return ano;
    }
    public void setAno(int ano) {
        this.ano = ano;
    }
    public float[] getNotas() {
        return notas;
    }
    public void setNotas(float[] notas) {
        this.notas = notas;
    }
    public Aluno[] getAlunos() {
        return alunos;
    }
    public void setAlunos(Aluno[] alunos) {
        this.alunos = alunos;
    }
    public Professor getProfessor() {
        return professor;
    }
    public void setProfessor(Professor professor) {
        this.professor = professor;
    }
```



Classe Curso

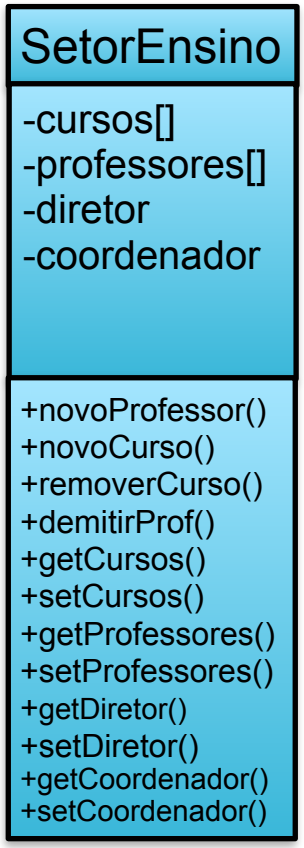



```
public class Curso {
    private String nome;
    private String ppc;
    private Disciplina disciplinas[];

    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getPpc() {
        return ppc;
    }
    public void setPpc(String ppc) {
        this.ppc = ppc;
    }
    public Disciplina[] getDisciplinas() {
        return disciplinas;
    }
    public void setDisciplinas(Disciplina[] disciplinas) {
        this.disciplinas = disciplinas;
    }
```

Classe SetorEnsino

```
public class SetorEnsino {

    private Curso cursos[];
    private Professor professores[];
    private String diretor;
    private String coordenador;


    public Curso[] getCursos() {
        return cursos;
    }
    public void setCursos(Curso[] curs
        this.cursos = cursos;
    }
    public Professor[] getProfessores()
        return professores;
    }
    public void setProfessores(Professor[] professores) {
        this.professores = professores;
    }
```

```
public String getDiretor() {
    return diretor;
}
public void setDiretor(String diretor) {
    this.diretor = diretor;
}
public String getCoordenador() {
    return coordenador;
}
public void setCoordenador(String coordenador) {
    this.coordenador = coordenador;
}
```



Melhorando as Classes

4. Crie construtores para as classes.

- Construtores são os métodos chamados no momento que se cria um objeto novo (mesmo nome da classe). Ex: Aluno x = **new Aluno()**;

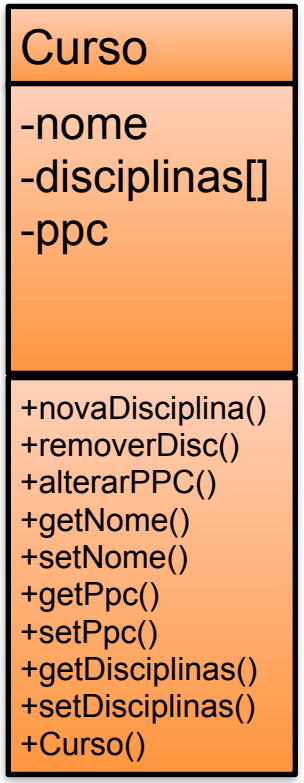
- É o único método que não tem retorno (nem mesmo void).

- É possível ter múltiplos construtores para mesma classe (mesmo nome).

- Caso não seja implementado um construtor, usa-se o método padrão de java que não recebe nenhum argumento.

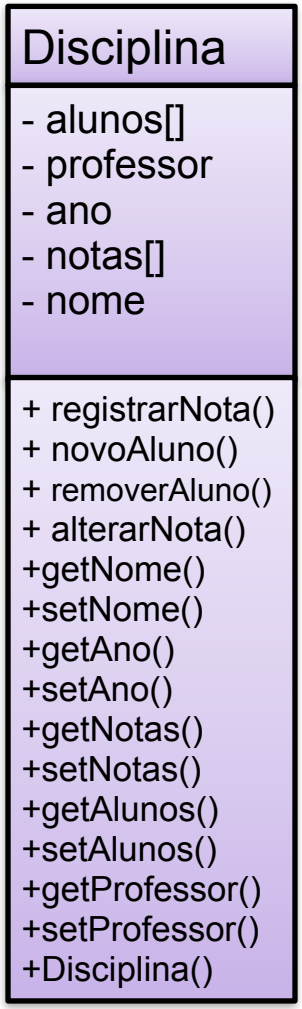
Classe Curso

```
public class Curso {

    private String nome;

    private String ppc;

    private Disciplina disciplinas[];


        public Curso(String nome, String ppc, Disciplina[] disciplinas) {

            this.nome = nome;

            this.ppc = ppc;

            this.disciplinas = disciplinas;

        }


        public Curso(String nome, String ppc) {

            this.nome = nome;

            this.ppc = ppc;

        }
```



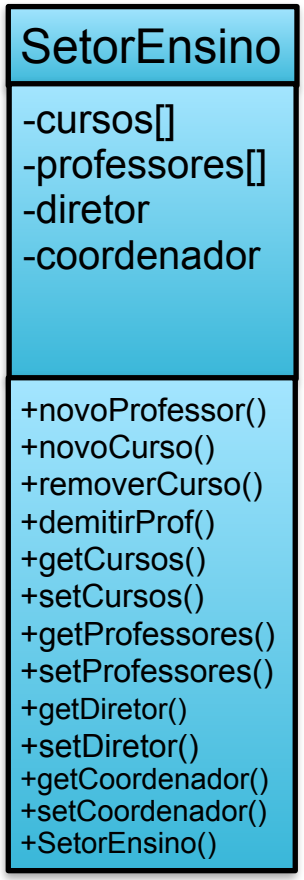
Classe Disciplina

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


         public Disciplina(Professor professor, String nome, int ano) {
                this.professor = professor;
                this.nome = nome;
                this.ano = ano;
                this.alunos = new Aluno[30];
                this.notas = new float[30];
           }
           public Disciplina(int quantAlunos, Professor professor, String nome, int ano) {
                this.alunos = new Aluno[quantAlunos];
                this.notas = new float[quantAlunos];
                this.professor = professor;
                this.nome = nome;
                this.ano = ano;
           }
```



Classe Aluno

```
public class SetorEnsino {
    private Curso cursos[];
    private Professor professores[];
    private String diretor;
    private String coordenador;

    public SetorEnsino(Curso[] cursos,
      Professor[] professores,
      String diretor,
      String coordenador) {
          this.cursos = cursos;
          this.professores = professores;
          this.diretor = diretor;
          this.coordenador = coordenador;
      }
```



E o que muda após criar os get/set e os construtores?

```
public class SetorEnsino {
    public boolean novoProfessor(String nome, long siape) {
        for ( int i = 0; i < professores.length; i++) {
            if (professores[i] != null) {
                professores[i] = new Professor();
                professores[i].nome = nome;
                professores[i].siape = siape;
                return true;
            }
        }
        return false;
    }
```

```
public class SetorEnsino {
    public boolean novoProfessor(String nome, long siape) {
        for ( int i = 0; i < professores.length; i++) {
            if (professores[i] != null) {
                professores[i] = new Professor(nome, siape);
                return true;
            }
        }
        return false;
    }
```

E o que muda após criar os get/set e os construtores?

```
public class SetorEnsino {
    public boolean demitirProfessor(long siape) {
        for ( int i = 0; i < professores.length; i++) {
            if (professores[i] != null &&
            professores[i].siape == siape) {
                professores[i] = null;
                return true;
            }
        }
        return false;
    }
}
```

```
public class SetorEnsino {
    public boolean demitirProfessor(long siape) {
        for ( int i = 0; i < professores.length; i++) {
            if (professores[i] != null &&
            professores[i].getSiape() == siape) {
                professores[i] = null;
                return true;
            }
        }
        return false;
    }
}
```

E o que muda após criar os get/set e os construtores?

```
public class SetorEnsino {
    boolean novoCurso(String nome, String ppc) {
        for ( int i = 0; i < cursos.length; i++) {
            if (cursos[i] != null) {
                cursos[i] = new Curso();
                cursos[i].nome = nome;
                cursos[i].ppc = ppc;
                return true;
            }
        }
        return false;
    }
}
```

```
public class SetorEnsino {
    boolean novoCurso(String nome, String ppc) {
        for ( int i = 0; i < cursos.length; i++) {
            if (cursos[i] != null) {
                cursos[i] = new Curso(nome, ppc);
                return true;
            }
        }
        return false;
    }
}
```

E o que muda após criar os get/set e os construtores?

```
public class SetorEnsino {
    boolean removerCurso(String nome) {
        for ( int i = 0; i < cursos.length; i++) {
            if (cursos[i] != null &&
            cursos[i].nome.equals(nome)) {
                cursos[i] = null;
                return true;
            }
        }
        return false;
    }
}
```

```
public class SetorEnsino {
    boolean removerCurso(String nome) {
        for ( int i = 0; i < cursos.length; i++) {
            if (cursos[i] != null &&
            cursos[i].getNome().equals(nome)) {
                cursos[i] = null;
                return true;
            }
        }
        return false;
    }
}
```

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.
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.