	PUC-MG: Pontifícia Universidade Católica de Minas Gerais
	Curso: Engenharia de Software
	Disciplina: Programação Modular
	Professor(a): Pedro Pongelupe Lopes
	Semestre: 2024.1
	Aluno: | **Matrícula:**

Exercício

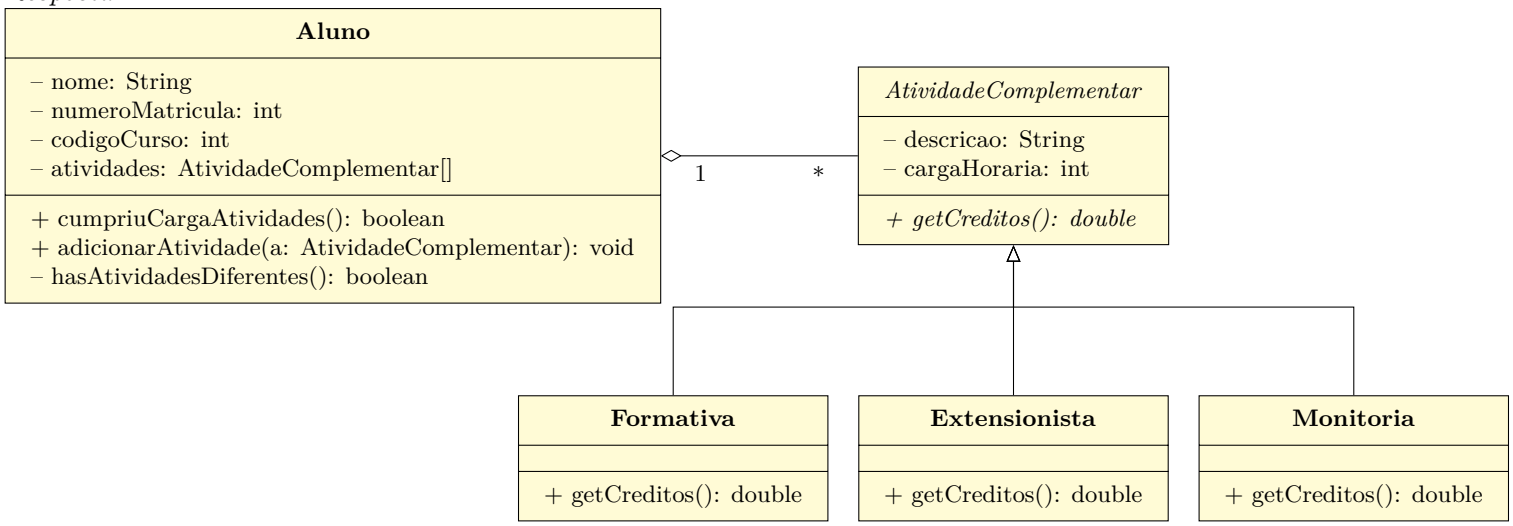
Questão 1 Em uma Universidade, um aluno é cadastrado no sistema acadêmico com nome, número de matrícula e código do curso que está realizando. Para se formar, além de cursar todas as disciplinas do currículo, o aluno precisa cumprir uma carga de atividades complementares.

Uma atividade complementar pode ser de três tipos: formativa, extensionista ou monitoria. Cada atividade tem uma descrição e uma carga horária. As atividades geram 1 crédito de acordo com a carga horária e seu tipo: as formativas geram 1 crédito a cada 30h; as extensionistas, a cada 15h e as de monitoria a cada 20h. Os créditos podem ter valores fracionários. Para se formar, o aluno precisa acumular pelo menos 4 créditos e ter participado de pelo menos 2 tipos diferentes de atividade.

Como o sistema acadêmico já tem implementado e funcionando a parte de matrículas, disciplinas, notas e aprovação, cabe a você planejar esta parte das atividades complementares.

(a) Utilizando todos os conceitos vistos até hoje na disciplina, **modele um diagrama de classes UML** para o problema proposto. O modelo deve incluir **classes, relacionamentos, atributos e métodos** necessários para resolver completamente o problema. **Não é necessário incluir** construtores ou métodos get/set, mas indique as visibilidades de métodos e atributos.

Resposta



(b) Considerando seu modelo em (a), escreva o código dos métodos envolvidos na tarefa de **calcular quantos créditos foram gerados por uma atividade complementar**. Note que, dependendo do seu modelo, pode ser necessário escrever o código de mais de um método nesta questão.

Resposta

```
// Formativa
@Override
public double getCreditos() { return getCargaHoraria() / 30; }

// Extensionista
@Override
public double getCreditos() { return getCargaHoraria() / 15; }

// Monitoria
@Override
public double getCreditos() { return getCargaHoraria() / 20; }
```

(c) Considerando seu modelo em (a), escreva o código dos métodos envolvidos na tarefa de **verificar se um aluno já cumpriu os requisitos de atividades complementares para poder se formar**. Note que, dependendo do seu modelo, pode ser necessário escrever o código de mais de um método nesta questão.

Resposta

```
public boolean cumpriuCargaAtividades() {
  double creditos = 0;
  for (AtividadeComplementar a : atividades)
    creditos += a.getCreditos();
  return creditos >= 4d && hasAtividadesDiferentes();
}

private boolean hasAtividadesDiferentes() {
  boolean[] arr = new boolean[3]; // 0 - formativa; 1 - extensionista; 3 - monitoria
  for (AtividadeComplementar a : atividades) {
    if (a instaceof Formativa) arr[0] = true;
    if (a instaceof Extensionista) arr[1] = true;
    if (a instaceof Monitoria) arr[2] = true;
  }
  var quantidade = 0;
  for (int i = 0; i < arr.length; i++) {
    if (arr[i]) quantidade++;
  }
  return quantidade > 1;
}
```

(d) Utilizando sintaxe JUnit, escreva o **código de testes unitários** para seu(s) método(s) da resposta (c).

Resposta

```
@Test
void testAlunoCumpriuCarga() {
  var a = new Aluno();
  a.adicionarAtividade(new Monitoria("monitoria de calculo 2", 60));
  a.adicionarAtividade(new Formativa("palestra sobre LaTeX", 60));

  assertTrue(a.cumpriuCargaAtividades());
}

@Test
void testAlunoNaoCumpriuCargaPorCreditos() {
  var a = new Aluno();
  a.adicionarAtividade(new Monitoria("monitoria de calculo 2", 60));
  a.adicionarAtividade(new Extensionista("TIS IV", 15));

  assertFalse(a.cumpriuCargaAtividades());
}
@Test
void testAlunoNaoCumpriuCargaPorTipoAtividade() {
  var a = new Aluno();
  a.adicionarAtividade(new Monitoria("monitoria de calculo 2", 60));
  a.adicionarAtividade(new Monitoria("monitoria de AED I", 60));
  a.adicionarAtividade(new Monitoria("monitoria de Desenvolvimento Web", 60));

  assertFalse(a.cumpriuCargaAtividades());
}
```