Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática – CEEI
Unidade Acadêmica de Sistemas e Computação – UASC
Disciplina: Laboratório de Programação 2

Laboratório 02

Como usar esse guia:

- Leia atentamente cada etapa
- Referências bibliográficas incluem:
  - material de referência desenvolvido por professores de p2/lp2 em semestres anteriores (ONLINE)
  - o livro Use a cabeça, Java (LIVRO-UseCabecaJava)
  - o livro Java para Iniciantes (Livro-JavaIniciantes)
- Quadros com dicas tem leitura opcional, use-os conforme achar necessário
- Preste atenção nos trechos marcados como importante (ou com uma exclamação)!

Sumário

Acompanhe o seu aprendizado

Conteúdo sendo exercitado

- Classes básicas
- Uso de objetos
- Uso do toString
- Introdução a documentação com javadoc

Objetivos de aprendizagem

- Iniciar o entendimento sobre classes e objetos. Observar que uma classe encapsula atributos e métodos e os objetos são instâncias dessa classe. Um programa OO é constituído de objetos que realizam tarefas a partir das “mensagens” enviadas a eles (chamada de métodos públicos) e podem se comunicar entre si.

Perguntas que você deveria saber responder após este lab

- O que é uma classe?
- O que é um objeto?
- Qual a diferença entre encapsulamento e ocultação da informação?
- O que significa dizer que cada objeto possui sua identidade?
- Quais os componentes essenciais de uma classe?
- Objetos são acessados por meio de variáveis de referência. O que isso significa?
- Dê exemplos de métodos acessadores e modificadores.
- O que é o método toString() e em quais situações devemos sobrescrevê-lo?

Material para consulta

- Referências bibliográficas incluem:
    - material de referência desenvolvido por professores de p2/lp2 em semestres anteriores (ONLINE)
    - o livro Use a cabeça, Java (LIVRO-UseCabecaJava)
    - o livro Java para Iniciantes (Livro-JavaIniciantes)
- Tutorial oficial Java
- Javadoc (tutorial oficial)
- Entendendo a diferença entre classes e objetos

Introdução

A disciplina de Programação 2 tem como foco o design e a construção de programas funcionalidades mais completas que necessitam de mais design do código. Nesse sentido, este e os próximos laboratórios trabalharão de forma diferente: cada laboratório terá uma especificação das funcionalidades a serem implementadas. Os critérios de avaliação dos laboratórios terão como base essa especificação.

É importante também que, a partir de agora, você comece a trabalhar com alguma IDE como, por exemplo, o Eclipse. Uma IDE é um ambiente integrado de desenvolvimento que oferece muitas facilidades durante a codificação. Recomendamos **fortemente** que neste momento você utilize o **Eclipse,** pois ele auxiliará no seu processo de aprendizagem. Ao longo dos labs daremos algumas dicas de como realizar certas ações no Eclipse, e não temos como dar todas as dicas para outras IDEs.

Como exemplo de classe e uso de objetos, considere o exemplo do controle acadêmico abaixo. Observe como o Aluno foi definido na classe Aluno (Aluno.java) e como os objetos dessa classe foram utilizados no ControleAcademico.

```
public class Aluno {

	private String nome;
	private int anoNascimento;

	public Aluno(String nome, int anoNascimento) {
		this.nome = nome;
		this.anoNascimento = anoNascimento;
	}

	public int getIdade() {
		return 2018 - anoNascimento;
	}

	public String toString() {
		return "Aluno - "  + this.nome;
	}

}
```

```
public class ControleAcademico {
	public static void main(String[] args) {
		Aluno a1 = new Aluno("JOAO", 1990);
		Aluno a2 = new Aluno("MARIA", 2000);
		System.out.println(a1);
		System.out.println(a2);
	}
}
```

Importante!

A partir de agora você deve documentar seu código! Para isso utilizaremos o modelo javadoc de documentação. O javadoc é um modelo de comentário de código que pode gerar, automaticamente, documentação como essa:

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

O Javadoc costuma ser o primeiro contato do desenvolvedor com um programa, e é importante que ele seja bem escrito. Este bloco de comentários deve ser inserido anteriormente à entidade que irá documentar. Veja os exemplos de documentação para diversas entidades de Java:

classe	`/**`
	`* Representação de um estudante, especificamente de computação, matriculado da * UFCG. Todo aluno precisa ter uma matrícula e é identificado unicamente`
	`* por esta matrícula.`
	`*`
	`* @author Matheus Gaudencio`
	`*/`
	`public class Aluno {`
	`...`

atributo	`/**`
	`* Matrícula do aluno. No formato 2XXXYYZZZ onde XX é o ano e`
	`* semestre de entrada, YY é o código do curso e ZZ é um identificador * do aluno no curso.`
	`*/`
	`private String matricula;`

construtor	`/**`
	`* Constrói um aluno a partir de sua matrícula e nome.`
	`* Todo aluno começa com o campo CRA como nulo.`
	`*`
	`* @param matricula a matrícula do aluno, no formato "0000000000"`
	`* @param nome o nome do aluno`
	`*/`
	`public Aluno(String matricula, String nome) {`
	`...`

método	`/**`
	`* Retorna a String que representa o aluno. A representação segue o`
	`* formato "MATRICULA - Nome do Aluno".`
	`*`
	`* @return a representação em String de um aluno.`
	`*/`
	`public String toString() {`
	`...`

Métodos com parâmetros usam a notação @param para indicar o que significa cada parâmetro.	

No Eclipse, no menu Project, opção "Generate Javadoc..." o Eclipse vai organizar o javadoc escrito por você em páginas HTML semelhantes às da API de java.

Controle Institucional da Situação Acadêmica (COISA)

Neste projeto, você deve desenvolver um sistema capaz de gerenciar o uso dos laboratórios de Ciência da Computação (LCC's) e sua vida acadêmica. O COISA se trata de um sistema complexo, logo a separação de responsabilidades através da criação de classes é de **extrema** importância, pois tem como objetivo estruturar, coerentemente, o seu programa.

A vida do aluno pode ser organizada em quatro atividades básicas: (1) organizar suas contas nos LCCs, (2) estudar para as disciplinas, (3) organizar suas contas nas cantinas e (4) acompanhar sua saúde física e mental. Para permitir o controle dessas 4 atividades, você irá desenvolver um sistema que permite avaliar a quantidade de espaço que você tem usado em um laboratório, a quantidade de horas que você tem estudado, como está sua conta (e o quanto você tem comido) nas cantinas da universidade e, por fim, como está sua saúde física e mental (e por consequência, sua saúde geral).

Assim, para cada uma das atividades, é descrito um conjunto de valores referentes a cada atividade e de ações que podem ser feitas para aquela atividade.

Atividade	**Estado (dados)**	**Ações**
Conta de Laboratório	Nome do laboratório Espaço ocupado Cota (limite de espaço)	Adicionar dados Remover dados Verificar uso de espaço Imprimir estado
Disciplina	Nome da disciplina Horas de estudo Notas 1, 2, 3 e 4	Cadastrar horas de estudo Cadastrar uma nota Calcular média Verificar se foi aprovado Imprimir estado
Conta da Cantina	Nome da cantina Débito em conta	Adicionar lanche (quantidade de alimentos e valor) Pagar conta Verificar o débito em conta Imprimir estado
Saúde	Saúde física Saúde mental	Definir estado de saúde mental Definir estado de saúde física Retorna estado geral de saúde

Para facilitar a sua vida, já existe uma classe pronta que irá ser executada para testar o funcionamento do seu programa. Por este programa é possível ver as classes que devem ser implementadas (dica: uma classe para cada atividade) bem como os métodos públicos. Você deve copiar essa classe no seu programa e fazê-la funcionar de acordo com as especificações que virão a seguir. Execute sempre esta classe para garantir que você está desenvolvendo corretamente cada atividade.

```
package lab2;

public class Coisa {

    public static void main(String[] args) {
```

```
        ContaLaboratorio contaLCC2 = new ContaLaboratorio("LCC2");
        contaLCC2.consomeEspaco(1999);
        System.out.println(contaLCC2.atingiuCota());
        contaLCC2.consomeEspaco(2);
        System.out.println(contaLCC2.atingiuCota());
        contaLCC2.liberaEspaco(1);
        System.out.println(contaLCC2.atingiuCota());
        contaLCC2.liberaEspaco(1);
        System.out.println(contaLCC2.atingiuCota());
        System.out.println(contaLCC2.toString());

        Disciplina prog2 = new Disciplina("PROGRAMACAO 2");
        prog2.cadastraHoras(4);
        prog2.cadastraNota(1, 5.0);
        prog2.cadastraNota(2, 6.0);
        prog2.cadastraNota(3, 7.0);
        System.out.println(prog2.aprovado());

        prog2.cadastraNota(4, 10.0);
        System.out.println(prog2.aprovado());
        System.out.println(prog2.toString());

        ContaCantina cantinaSeuMatias = new ContaCantina("Seu Matias");
        cantinaSeuMatias.cadastraLanche(2, 500);
        cantinaSeuMatias.cadastraLanche(1, 500);
        cantinaSeuMatias.pagaConta(200);
        System.out.println(cantinaSeuMatias.getFaltaPagar());
        System.out.println(cantinaSeuMatias.toString());

        Saude saude = new Saude();
        System.out.println(saude.getStatusGeral());
        saude.defineSaudeMental("boa");
        saude.defineSaudeFisica("boa");
        System.out.println(saude.getStatusGeral());

        saude.defineSaudeMental("fraca");
        saude.defineSaudeFisica("fraca");
        System.out.println(saude.getStatusGeral());

        saude.defineSaudeMental("boa");
        saude.defineSaudeFisica("fraca");
        System.out.println(saude.getStatusGeral());
    }
}
```

Essa classe ao ser executada deve produzir a seguinte saída:

```
false
true
true
false
LCC2 1999/2000
false
true
PROGRAMACAO 2 4 7.0 [5.0, 6.0, 7.0, 10.0]
800
Seu Matias 3 1000
boa
```

```
boa
fraca
ok
```

A seguir apresentaremos em detalhe cada funcionalidade, buscando facilitar o seu entendimento do que deve ser feito.

1. Conta de um Laboratório

A conta de laboratório deve ser responsável por manter o registro da quantidade de espaço utilizado, pelo aluno, em determinado laboratório. Para cada laboratório, seria criado um objeto para controle desse estado (espaço ocupado).

Cada laboratório também tem uma COTA própria. A COTA determina o limite de espaço disponível no sistema de armazenamento. Considere que a cota de todo laboratório é, por padrão, de 2000mb (aproximadamente 2gb). É possível passar uma cota como parâmetro na construção do objeto que representa a conta de um laboratório. O *toString* deste objeto deve gerar uma String que o representa contendo: o nome do laboratório, o espaço ocupado e a cota.

Ao atingir a cota, o método de verificação de cota deve indicar que a conta do usuário atingiu a cota (através de um booleano). Mesmo com a cota cheia, isto é, limite de espaço disponível no sistema de armazenamento atingido, o usuário ainda pode adicionar dados, ou seja, ele pode ocupar mais espaço do que o limite indicado na cota.

Para implementar esta funcionalidade, você deve criar uma classe **ContaLaboratorio** com os construtores: **ContaLaboratorio (String nomeLaboratorio)** e **ContaLaboratorio (String nomeLaboratorio, int cota)**.

É importante também implementar os métodos abaixo:

- **void consomeEspaco(int mbytes)**
- **void liberaEspaco(int mbytes)**
- **boolean atingiuCota()**
- **String toString()**

2. Disciplina

Por padrão toda disciplina tem 4 notas. Calcula-se a média da disciplina, fazendo a média aritmética dessas 4 notas (sem arredondamento). Todo aluno é considerado aprovado quando atinge a média igual ou acima de 7.0. Caso alguma das notas não seja cadastrada, ela é considerada como zero. Se a alguma nota (nota 1, nota 2, nota 3 ou nota 4) for cadastrada mais de uma vez, consideramos a última nota cadastrada (ou seja, é possível repor notas nesse sistema). É possível cadastrar horas de estudo para determinada disciplina. As horas de estudo são cumulativas, ou seja, a nova quantidade cadastrada soma-se às previamente cadastradas.

O *toString()* deve gerar uma representação da disciplina que contenha o nome da disciplina, o número de horas de estudo, a média do aluno e as notas de cada prova.

Cada objeto que representa uma disciplina começa sem horas de estudo cadastradas. Para desenvolver esta funcionalidade, você deve implementar a classe **Disciplina** que tem o construtor **Disciplina(String nomeDisciplina)**. Implemente, também, os métodos:

- **void cadastraHoras(int horas)**
- **void cadastraNota(int nota, double valorNota)** // notas possíveis: 1, 2, 3 e 4
- **boolean aprovado( )**
- **String toString( )**

3. Conta na Cantina

O aluno pode manter contas ("penduras") em várias cantinas da universidade. Para controlar os gastos em uma cantina, os lanches realizados são adicionados à conta de uma determinada cantina. Para cada lanche são indicados o número de itens que foram consumidos naquele momento e o valor gasto (em centavos). Ao pagar a conta o usuário indica o valor que será pago em centavos. Este valor pode ser menor, mas nunca superior ao valor devido. A qualquer momento, o aluno pode consultar o valor que ainda deve na cantina.

Por fim, o usuário pode gerar uma string representando o histórico de sua conta. O método *toString(),* comunica: o nome da cantina, a quantidade de itens e o valor dos lanches cadastrados até o momento. Observe que este método apresenta o valor dos gastos independentemente dos valores pagos ou não.

Para controlar os gastos de uma cantina, você deve criar a classe **ContaCantina** com o construtor **ContaCantina(String nomeDaCantina)**. Esta classe deve ter os métodos:

- **void cadastraLanche(int qtdItens, int valorCentavos)**
- **void pagaConta(int valorCentavos)**
- **int getFaltaPagar( )**
- **String toString( )**

4. Sua saúde física e mental

Por fim, é importante acompanhar o estado de saúde geral do aluno. Todo aluno tem uma saúde física e mental. Cada uma delas pode estar boa ou fraca. Caso ambas estejam fracas, a saúde geral do aluno é “fraca”. Se ambas estiverem boas, a saúde geral do aluno é “boa”. Se uma delas estiver fraca, mas a outra estiver boa, a saúde geral do aluno é considerada “ok”.
Faça uma classe **Saude** que tenha um construtor inicial sem parâmetros e que irá definir a saúde mental e a saúde física do aluno como “boa”. Essa classe precisa ter 3 métodos:

- **void defineSaudeMental(String valor)**
- **void defineSaudeFisica(String valor)**
- **String getStatusGeral()**

5. Bônus!

Ao terminar as atividades propostas anteriormente, você pode tentar completar as tarefas descritas a seguir.

5.1 - Mais Notas na Disciplina

Na classe Disciplina, crie um construtor adicional que recebe também o número de notas da disciplina. Além desse construtor, crie outro construtor que recebe o número de notas e um array de inteiros com os pesos de cada uma das notas, para o cálculo de uma média ponderada.

Por exemplo, uma disciplina de 2 notas e pesos [6, 4] tem sua média calculada como (6 * Nota_1 + 4 * Nota_2) / 10. Caso o array de pesos não seja passado, considere cada nota com mesmo peso (ou seja, a média é uma média aritmética: (Nota_1 + Nota_2)/2).

5.2 - Registro Detalhado de Lanches

Na conta da cantina, além de registrar o número de itens, você pode registrar um pequeno texto descrevendo o seu lanche. Por exemplo, “Almoço com a família”, ou, “Café de 2 litros para a aula”. Este registro é opcional. Além disso você deve criar um método que retorne os últimos 5 detalhes cadastrados.

Ou seja, você deve implementar, além dos métodos básicos de ContaCantina, os métodos:

- **void cadastraLanche(int qtdItens, int valorCentavos, String detalhes)**
- **String listarDetalhes( )** // deve retornar uma string com os últimos 5 detalhes, um em cada linha

5.3 - Nível de Saúde

Além da saúde mental e física, o aluno pode cadastrar um emoji que descreva sua última sensação em geral. Por exemplo, há certos dias que ele pode estar “**:(**”, “*_*”, “:o)”,“<(^_^<)”, “¯_(ツ)_/¯” ...

O emoji, não tem relação com o cálculo da saúde geral do aluno, que é baseado na saúde física e mental. Ele expressa o sentimento do aluno no momento.

Caso seja registrado um emoji, ele deve ser retornado como adicional ao estado de saúde geral do aluno. No entanto, caso o estado geral de saúde mude isto é, haja alguma alteração na saúde física ou mental, a mensagem de estado geral de saúde deve voltar a ter a saída padrão (sem o emoji).

Assim, além da alteração no método **getStatusGeral()**, é preciso adicionar um novo método:

- **void definirEmoji(String valor)**

5.4 - Múltiplos Alunos

O sistema deve permitir agora múltiplos alunos. Cada aluno deve ter um conjunto de disciplinas, conjunto de contas de laboratório e de cantina e um objeto que represente sua saúde. Para isso, crie uma classe **Aluno** que tenha atributos representando esse estado. O Aluno deve ter os seguintes métodos:

- **void cadastraLaboratorio(String nomeLaboratorio)**
- **void cadastraLaboratorio(String nomeLaboratorio, int cota)**
- **void consomeEspaco(String nomeLaboratorio, int mbytes)**
- **void liberaEspaco(String nomeLaboratorio, int mbytes)**

- **boolean atingiuCota(String nomeLaboratorio)**
- **String laboratorioToString(String nomeLaboratorio)**
- **void cadastraDisciplina(String nomeDisciplina)**
- **void cadastraHoras(String nomeDisciplina, int horas)**
- **void cadastraNota(String nomeDisciplina, int nota, double valorNota)**
- **boolean aprovado(String nomeDisciplina)**
- **String disciplinaToString(String nomeDisciplina)**
- **void cadastraCantina(String nomeCantina)**
- **void cadastraLanche(String nomeCantina, int qtdItens, int valorCentavos)**
- **void pagarConta(String nomeCantina, int valorCentavos)**
- **int getFaltaPagar(String nomeCantina)**
- **String cantinaToString(String nomeCantina)**
- **void defineSaudeMental(String valor)**
- **void defineSaudeFisica(String valor)**
- **String getStatusGeral()**

5.5 - Linha de Comando

Faça com que seu sistema receba comandos da entrada e que traduzam tais comandos em ações nesse sistema (exemplo: CONSOME LCC1 1000; LIBERA LCC1 50; ...). Não é necessário criar um "Menu" para o seu sistema. É suficiente receber esses comandos na entrada padrão (na linha de comando).

Entrega

No Eclipse, faça um projeto que contenha as classes necessárias para fazer funcionar o programa passado inicialmente. É opcional fazer as funcionalidades bônus.

É importante que todo o código esteja devidamente documentado com javadoc. Isto significa que classes, atributos e métodos devem ser descritos usando o formato de Javadoc apresentado.

Para a entrega, faça um **zip** da pasta do seu projeto. Coloque o nome do projeto para: LAB2_SEU_NOME e o nome do zip para LAB2_SEU_NOME.ZIP. Exemplo de projeto: LAB2_MATHEUS_GAUDENCIO.ZIP. Este zip deve ser submetido pelo Canvas.

Seu programa será avaliado pela corretude e, principalmente, pelo DESIGN do sistema. É importante:

- Usar nomes adequados de variáveis, classes, métodos e parâmetros.
- Fazer um design simples, legível e que funciona. É importante saber, apenas olhando o nome das classes e o nome dos métodos existentes, identificar quem faz o que no código.