GABARITO - ATIVIDADE 03 - POO
André Luiz Takayama Oliveira

Questão 01

```java
import java.time.LocalDate;

public class Main {
    public static void main(String[] args) throws Exception {
        LocalDate dataNascimentoEstudante01 = LocalDate.of(2000, 5,
10);
        LocalDate dataInsercaoEstudante01 = LocalDate.of(2023, 8, 27);
        LocalDate dataMatriculaEstudante01 = LocalDate.of(2023, 1, 02);

        Aluno estudante01 = new Aluno(01, "André", "Rua Fulaninha",
"909826-9797", dataNascimentoEstudante01, "04267", "04587890448-32",
dataInsercaoEstudante01, "14121f", dataMatriculaEstudante01);
        estudante01.aluno();

        LocalDate dataNascimentoProfessor01 = LocalDate.of(2000, 5,
10);
        LocalDate dataInsercaoProfessor01 = LocalDate.of(2023, 8, 27);
        LocalDate dataContratacaoProfessor01 = LocalDate.of(2023, 1,
02);

        Professor docente01 = new Professor(01, "Takayama", "Rua Nikas
Pantmun", "901226-9797", dataNascimentoProfessor01, "04267",
"04587890448-32", dataInsercaoProfessor01, "14121f",
dataContratacaoProfessor01);
        docente01.getNome();
        docente01.professor();
    }
}
```

```java
import java.time.LocalDate;

public class BasePessoa {
    private int codigo;
    private String nome;
    private String endereco;
    private String telefone;
```

```
    private LocalDate dataNascimento;
    private String rg;
    private String cpf;
    private LocalDate dataInsercao;

    public int getCodigo() {
        return codigo;
    }

    public String getNome() {
        return nome;
    }

    public String getEndereco() {
        return endereco;
    }

    public String getTelefone() {
        return telefone;
    }

    public LocalDate getDataNascimento() {
        return dataNascimento;
    }

    public String getRg() {
        return rg;
    }

    public String getCpf() {
        return cpf;
    }

    public LocalDate getDataInsercao() {
        return dataInsercao;
    }

    public BasePessoa(int codigo, String nome, String endereco, String
telefone,
    LocalDate dataNascimento, String rg, String cpf, LocalDate
dataInsercao) {

        this.codigo = codigo;
```

```
        this.nome = nome;
        this.endereco = endereco;
        this.telefone = telefone;
        this.dataNascimento = dataNascimento;
        this.rg = rg;
        this.cpf = cpf;
        this.dataInsercao = dataInsercao;
    }

}
```

```
import java.time.LocalDate;

public class Aluno extends BasePessoa {
    String matricula;
    LocalDate dataMatricula;

    Aluno(int codigo, String nome, String endereco, String telefone,
LocalDate dataNascimento, String rg,
    String cpf, LocalDate dataInsercao, String matricula, LocalDate
dataMatricula) {

        super(codigo, nome, endereco, telefone, dataNascimento, rg,
cpf, dataInsercao);

        this.matricula = matricula;
        this.dataMatricula = dataMatricula;
    }

    public String getMatricula() {
        return matricula;
    }

    public LocalDate getDataMatricula() {
        return dataMatricula;
    }

    public void aluno (){
        System.out.println("Código: " + getCodigo());
        System.out.println("Nome: " + getNome());
        System.out.println("Endereço: " + getEndereco());
        System.out.println("Telefone: " + getTelefone());
```

```
        System.out.println("Data de Nascimento: " +
getDataNascimento());
        System.out.println("RG: " + getRg());
        System.out.println("CPF: " + getCpf());
        System.out.println("Matrícula: " + getMatricula());
        System.out.println("Data de Nascimento: " +
getDataNascimento());
        System.out.println("Data de Inserção: " + getDataInsercao());
        System.out.println("Data de Matrícula: " + getDataMatricula());
    }


}
```

```
import java.time.LocalDate;

public class Professor extends BasePessoa {
        String registro;
        LocalDate dataContratacao;

        Professor(int codigo, String nome, String endereco, String
telefone, LocalDate dataNascimento,
            String rg, String cpf, LocalDate dataInsercao, String
registro, LocalDate dataContratacao) {

            super(codigo, nome, endereco, telefone, dataNascimento, rg,
cpf, dataInsercao);

            this.registro = registro;
            this.dataContratacao = dataContratacao;
        }

        public String getRegistro() {
            return registro;
        }

        public LocalDate getDataContratacao() {
            return dataContratacao;
        }

        public void professor (){
            System.out.println("Código: " + getCodigo());
```

```
            System.out.println("Nome: " + getNome());
            System.out.println("Endereço: " + getEndereco());
            System.out.println("Telefone: " + getTelefone());
            System.out.println("Data de Nascimento: " +
getDataNascimento());
            System.out.println("RG: " + getRg());
            System.out.println("CPF: " + getCpf());
            System.out.println("Matrícula: " + getRegistro());
            System.out.println("Data de Nascimento: " +
getDataNascimento());
            System.out.println("Data de Inserção: " +
getDataInsercao());
            System.out.println("Data de Matrícula: " +
getDataContratacao());
        }
}
```

Código copiado diretamente do meu VSCode, mas está também no github, (link anexado no corpo do email)

Questão 02.

Alternativa correta é a letra B. Pois o código já está sem o “setId”.

Questão 03.

A) Se necessário, o programador pode criar uma instância dessa classe, facilitando assim sua operação, bastando adicionar a outra classe a linha de código:

Calculadora calc = new Calculadora()

Professor, essa foi a afirmativa da alternativa, mas eu fiquei na dúvida se a falta do ponto e vírgula (;) ao fim da linha foi um erro de digitação ou uma pegadinha, pois o programador poderia ir na classe main e criar uma instância da classe calculadora com a linha do exemplo, mas precisaria de um ponto e vírgula no final. Se foi erro de digitação, a alternativa é verdadeira, se foi pegadinha, é falsa.

B) Falso. Influencia sim, visto que classes abstratas não podem ser influenciadas, apenas herdadas. A palavra “abstract” não faz falta, se estivesse no código, mudaria completamente o comportamento do mesmo.

C) Falso. O código está absolutamente certo, não tem nenhum erro.

D) Falso. O uso da palavra “abstract” não tem a ver como ele ter a possibilidade de ser instanciado diretamente.

E) Falso. A classe está escrita corretamente.

Questão 04.

Alternativa C, pois um objeto é uma instância de uma classe que possui atributos, métodos e estado. (No curso de POO do Guanabara, que o senhor nos indicou, ele ensina o macete AME. (A = ATRIBUTO, M = MÉTODO, E = ESTADO.

Questão 05.

Alternativa B, as classes serem potencialmente reutilizáveis é o que faz a programação orientada a objetos ser vantajosa para alguns aspectos, tendo em vista que facilita e otimiza o trabalho do desenvolvedor, pois não precisa ficar reescrevendo partes do código do absoluto zero.