	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 de Revisão Prova 3

Questão 1 Analise os seguintes fragmentos de código e responda:

- Esse código compila? Se não, porquê?
- Esse código tem alguma exceção não tratada? Se sim, qual?
- Se complia e não tem nenhuma exceção não tratada, qual a saída esperada no console?

(a)

```
public class Driver {
        public static void main(String[] args) {
                int i = 0;
                int y = 6;

                try {
                        y /= i;
                        i = 5;
                        y += 2;
                } catch (RuntimeException e) {
                        i++;
                        y = 4;
                }
                System.out.println(i + y);
        }
}
```

Resposta Código compila, não tem execptions não tratadas lançadas. A saída é 5

(b)

```
public class Driver {
        public static void main(String[] args) {
                var str = "cr";
                try {
                        str += "e";
                } catch (Exception e) {
                        str += "i";
                } finally {
                        str += "me";
                }

                System.out.println(str);
        }
}
```

Resposta Código compila, não tem execptions não tratadas lançadas. A saída é "creme"

(c)

```
public class Driver {
        public static void main(String[] args){
                var anoAtual = LocalDate.now().get(ChronoField.YEAR);

                if (anoAtual == 1998) {
                        throw new Exception("Nao estamos mais em 1998");
                } else {
                        System.out.println("Estamos no ano de: " + anoAtual);
                }
        }
}
```

Resposta Código não compila, pois, não tem a declaração da checked exception no método main

(d)

```
public class Driver {
        public static void main(String[] args) {
                var arr = new int[] { 1, 9, 5, 0 };
                for (var i : arr) {
                        if (i % 2 == 0)
                                throw new NumeroParException();
                }
                System.out.println("somente numeros impares!");
        }
        public static class NumeroParException extends RuntimeException {}
}
```

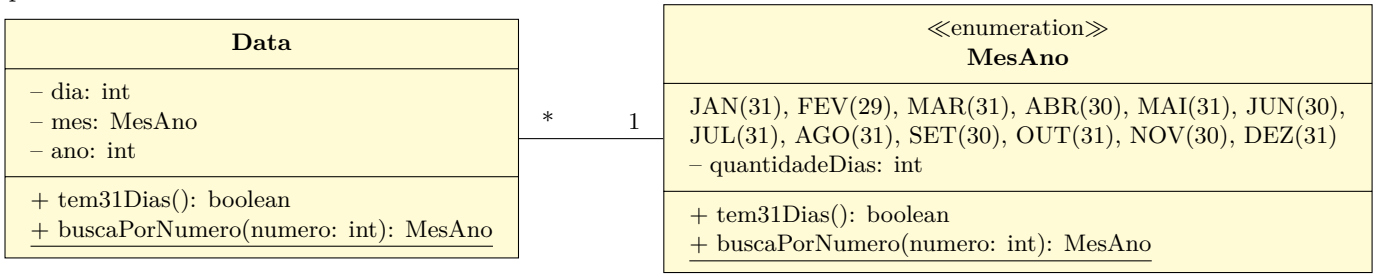
Resposta Código compila, mas lança há uma exception não tratada NumeroParException.

Questão 2 Considere o seguinte código Java:

```
public enum MesAno {
        JAN(31), FEV(29), MAR(31), ABR(30), MAI(31), JUN(30), JUL(31), AGO(31), SET(30), OUT(31), NOV(30), DEZ(31);

        private final int quantidadeDias;

        private MesAno(int quantidadeDias) {
                this.quantidadeDias = quantidadeDias;
        }
        public boolean tem31Dias() {
                return quantidadeDias == 31;
        }
        public static MesAno buscaPorNumero(int numero) {
                return MesAno.values()[numero - 1];
        }
}
public class Data {

        private int dia;

        private MesAno mes;

        private int ano;

        public Data(int dia, int mes, int ano) {
                this.dia = dia;
                this.mes = MesAno.buscaPorNumero(mes);
                this.ano = ano;
        }

        public int getDia() {
                return dia;
        }

        public void setDia(int dia) {
                this.dia = dia;
        }

        public MesAno getMes() {
                return mes;
        }

        public void setMes(MesAno mes) {
                this.mes = mes;
        }

        public int getAno() {
                return ano;
        }

        public void setAno(int ano) {
                this.ano = ano;
        }

        public boolean mesTem31Dias() {
                return mes.tem31Dias();
        }
}
```

(a) Desenhe o diagrama UML do código.

Resposta



(b) Altere o método *buscaPorNumero(int numero)* para ele lançar uma exceção quando o paramêtro *numero* não for válido. Ex: *numero = 19*

```
        public static MesAno buscaPorNumero(int numero) {
                if (numero < 1 || numero > 12) throw new IllegalArgumentException("mes invalido");
                return MesAno.values()[numero - 1];
        }
}
```

(c) Implemente um método *main* que instâncie as seguintes 3 datas:

- 22/04/1870
- 30/06/2002
- 15/-15/2038

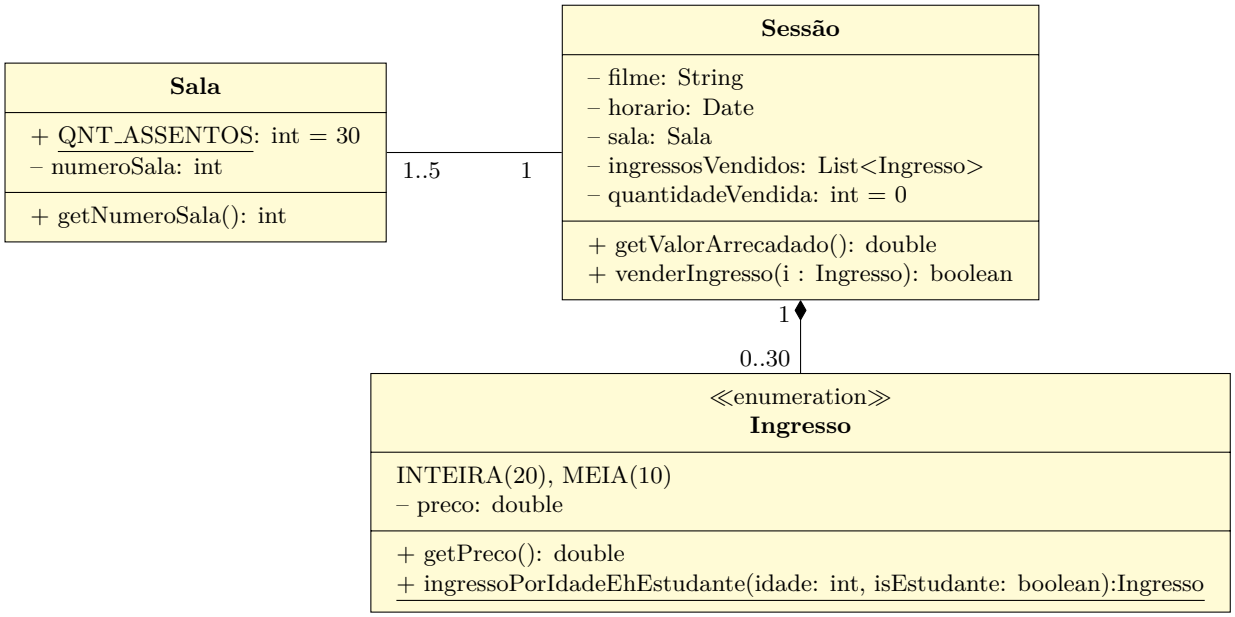
Utilizando try-catch, caso alguma data seja inválida, instâncie **qualquer** data válida como valor padrão.

Resposta

```
public class Driver {
        public static void main(String[] args) {
                var data1 = criaData(22, 4, 1870);
                var data2 = criaData(30, 6, 2002);
                var data3 = criaData(15, -15, 2038);

                System.out.println(data1);
                System.out.println(data2);
                System.out.println(data3);
        }
        private static Data criaData(int dia, int mes, int ano) {
                try {
                        return new Data(dia, mes, ano);
                } catch (IllegalArgumentException e) {
                        return new Data(1, 1, 1950);
                }
        }
}
```

Questão 3 Analise a seguinte situação proposta e o diagrama UML proposto como solução do problema :

Em um cinema, há 5 salas que possuem o número da sala e 30 assentos disponíveis cada. Em um dia de funcionamento desse cinema, as salas funcionam exibindo sessões, cada sessão possui um horário e o filme em cartaz. A bilheteria do cinema vende os ingressos das sessões ao público. Cada ingresso custa R$ 20,00 a inteira e R$ 10,00 a meia entrada que é dada para pessoas menores de 21 anos e estudantes. Além disso, para acompanhamento da saúde financeira do cinema, é necssário falar qual o valor arrecadado com os ingressos vendidos de cada sessão.



Escreva o código Java correspondente ao diagrama anterior.

Resposta

```
public class Sala {
        public static final int QNT_ASSENTOS = 30;
        private int numeroSala;

        public int getNumeroSala() { return numeroSala; }
}

public class Sessao {

        private String filme;
        private Date horario;
        private Sala sala;
        private List<Ingresso> ingressosVendidos;
        private int quantidadeVendida = 0;

        public Sessao(String filme, Date horario, Sala sala) {
                this.filme = filme;
                this.horario = horario;
                this.sala = sala;
                this.ingressosVendidos = new ArrayList<>();
        }

        public double getValorArrecadado() {
                return ingressosVendidos
                                .stream()
                                .mapToDouble(Ingresso::getPreco)
                                .sum();
        }

        public boolean venderIngresso(Ingresso i) {
                var podeVender = quantidadeVendida < 30;
                if (podeVender) {
                        ingressosVendidos.add(i);
                        quantidadeVendida++;
                }
                return podeVender;
        }
}

public enum Ingresso {
        INTEIRA(20), MEIA(10);

        private double preco;

        private Ingresso(double preco) { this.preco = preco; }

        public double getPreco() {return preco; }

        public static Ingresso ingressoPorIdadeEhEstudante(int idade, boolean isEstudante) {
                var ingresso = INTEIRA;
                if (idade < 21 || isEstudante)
                        ingresso = MEIA;
                return ingresso;
        }
}
```