Universidade do Estado de Santa Catarina
Ciência da Computação - Programação Orientada a Objetos
Prova II – 2018 / 2

Nome: _____ **Nota:** _____

Questão 1 (4,0) Implemente uma classe Fracao com as seguintes propriedades:

a) (0,6) A classe deve ter dois elementos – numerador e denominador – que são de um tipo genérico (*generics*), mas que sejam obrigatoriamente numéricos;

b) (0,4) A classe deve ter *getters* e *setters* para o numerador e o denominador, e também deverá sobrescrever o método `toString()` de modo a mostrar um objeto fração em um estilo apropriado (ex: n / d). Crie ainda um construtor que recebe dois valores: `num` e `den`, sendo que o primeiro representa o valor do numerador e o segundo representa o valor do denominador;

c) (0,5) Ainda sobre o construtor mencionado, garanta que se o denominador da fração for zero, uma exceção do tipo `java.lang.ArithmeticException` deverá ser lançada não concluindo a criação do objeto;

d) (0,6) Crie uma nova classe de Exceção em tempo de execução denominada `InvalidSquareRootException` que possui um construtor o qual especifica a seguinte mensagem (String) de erro: `"Invalid square root. Imaginary numbers required."`;

e) (0,8) Crie um método denominado `sqrt( ) : Fracao` que realiza a operação de raiz quadrada da Fracao atual (this) e retorna um novo objeto de Fracao que contem o resultado. Caso a raiz quadrada seja negativa (isto é, ou o denominador ou o numerador eram negativos), deve-se lançar uma exceção `InvalidSquareRootException`;

f) (0,4) No método principal (*main*) crie um objeto Fração válido (ex: 11 / 7) e um objeto Fração inválido (ex: 3 / 0). Após isto, utilize o método `sqrt` da classe Fracao para ambos os objetos criados;

g) (0,7) Faça o devido tratamento de erros para evitar que o programa termine sua execução com a tentativa de criar a fração inválida ou realizar a operação de raiz quadrada de um valor inválido.

Questão 2 (6,0) Leia a descrição da problemática abaixo e construa um sistema que modele o cenário descrito utilizando os conceitos de herança, interface e polimorfismo. Realize o que se pede:

OBS: o diagrama de classes é apresentado como material suplementar de uma modelagem adequada.

Um determinado estabelecimento conhecido como "locobanca", estabelecido na região norte de Joinville, trabalha com locações e venda de alguns produtos. Basicamente, este estabelecimento trabalha com mídias de filmes, mídias de séries e revistas. Todos esses produtos possuem um código numérico utilizado na venda, um nome (texto) e a quantidade de dias desde o seu lançamento oficial ("tempo de vida"). Mais especificamente, é necessário ainda registrar em sistema a duração em minutos dos filmes, o número da temporada e o número de episódios das séries e o número de páginas e o valor sugerido (base) das revistas. Todos os atributos dos produtos devem ser privados, mas devem ter *getters* & *setters* públicos que permitem acesso e alteração.

Entre os produtos mencionados, apenas as mídias de filmes e séries podem ser alugadas por clientes da locobanca. Toda mídia que pode ser emprestada deve manter registro da sua data de empréstimo atual (que é armazenada como texto) e da quantidade de dias que ficará emprestada (valor numérico), caso a mídia esteja alocada (caso contrário, ficam sem preenchimento). Assim, ao realizar uma operação de alugar uma mídia, deve-se informar qual a quantidade de dias, e qual a data atual. Como retorno, essa operação deve devolver o valor a ser cobrado por tal empréstimo que, por padrão é calculado como:

- Se o produto tem "tempo de vida" menor do que 365 dias, então: R$ 4,00 × #dias do empréstimo
- Caso contrário: R$ 2,50 × #dias do empréstimo

Para séries, entretanto, este valor deve ser multiplicado por um ponderador β definido por:

$$\beta = 1 + \text{episódios} / 100$$ (i.e. sobrescrever o método nesta classe).

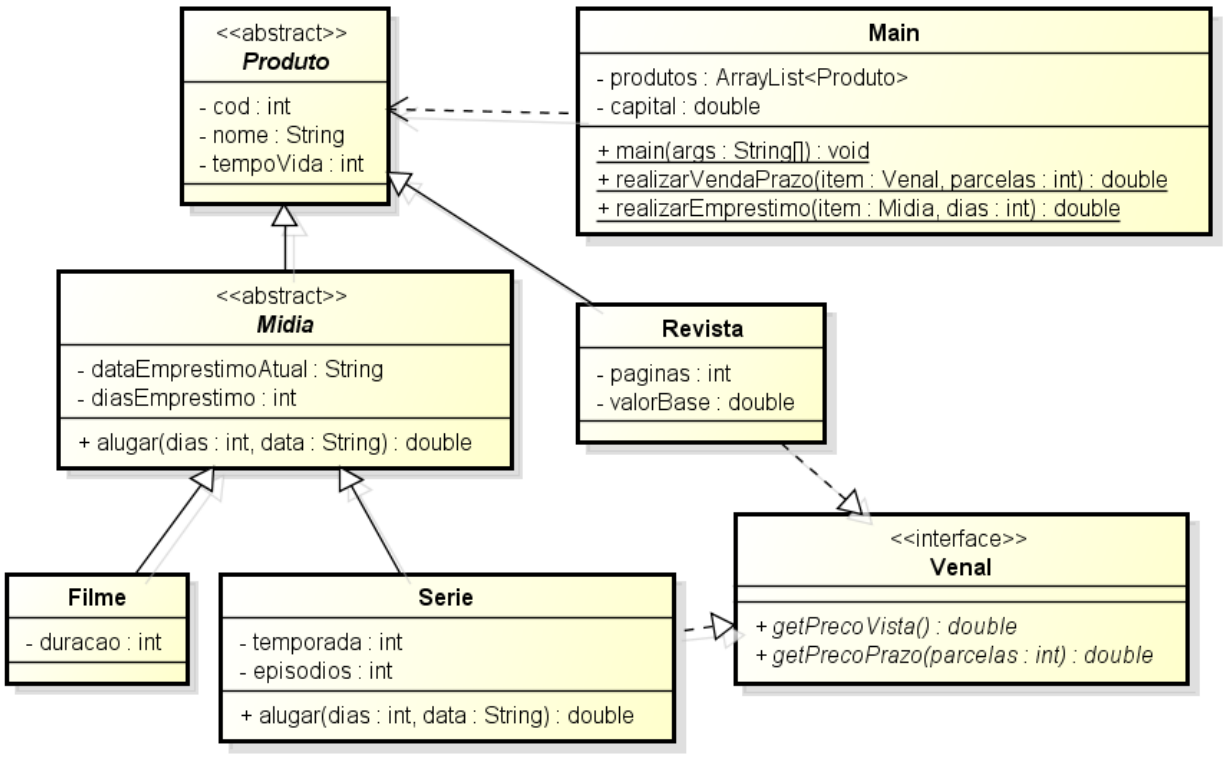
[continua no verso]

Dentre os produtos da locobanca, somente as revistas e as mídias de séries são itens Venais, isto é, comercializáveis com o propósito de venda. Note que todo item Venal, por lei, deve ter um preço calculável para pagamento a vista e um preço de parcela calculável para pagamentos a prazo. Neste sentido, as revistas vendidas à vista são comercializadas pelo seu valor base multiplicado por 150%, e o valor das parcelas ρ, em um pagamento a prazo, é calculado como: $\rho = \text{valorBase} \times (1{,}5 + \#\text{parcelas} \times 0{,}05)$. Já as mídias de séries vendidas à vista são comercializadas por preço único de R$ 25,00 ou em duas parcelas de R$ 14,00 ou em três parcelas de R$ 10,00.

Com a estrutura de modelagem pronta, implemente na classe principal os seguintes métodos estáticos:

- ⇨ Um método polimórfico denominado `realizarVendaPrazo` que recebem um objeto `Venal` e um valor inteiro indicando a quantidade de parcelas de uma compra a prazo, e retorna um valor real (double) com o valor das parcelas;
- ⇨ Um método polimórfico denominado `getPeriodoEmprestimo`, que recebe um objeto `Produto` e retorna uma String que identifica a data do empréstimo e a quantidade de dias deste empréstimo (ex: "produto emprestado em: 12/10/2018 por 3 dias."), caso o produto esteja emprestado, ou uma String que informa que o produto não foi locado, ou então, caso seja uma Revista, uma String que informa que o produto não pode ser emprestado.

Por fim, no método principal da classe principal do programa, crie uma coleção de produtos do tipo `ArrayList` e adicione cinco produtos criados com valores fixos, mesclando entre filmes, séries e revistas (pelo menos um de cada). Na sequência, peça para o usuário digitar dois números de códigos de produtos: α e β. Então itere sobre os produtos da coleção criada e, para o produto com código α, utilize o método `realizarVendaPrazo` e para o produto com código β, utilize o método `getPeriodoEmprestimo`.




```
if( you.haveStudied() ) printf("--- Boa prova! ---");
else printf("--- Boa sorte! ---");
```