Programação Orientada a Objetos

TEMA 4 - Considerações Sobre Classes e Objetos

EXERCÍCIOS:

1. O que é uma ***exceção***?

2. Qual o objetivo de se rotular um método com o modificador de acesso **public**?

3. Por qual motivo usamos o modificador **private** para rotular variáveis de instância e alguns métodos da classe?

4. Qual o propósito da palavra-chave **this**?

5. O que é *composição*?

6. Crie uma classe chamada Racional para realizar aritmética com frações. Escreva um programa para testar sua classe. Use variáveis de inteiros para representar as variáveis de instância private da classe — o numerador e o denominador. Forneça um construtor que permita que um objeto dessa classe seja inicializado quando ele for declarado. O construtor deve armazenar a fração em uma forma reduzida. A fração

   2/4

   é equivalente a 1/2 e seria armazenada no objeto como 1 no numerador e 2 no denominador. Forneça um construtor sem argumento com valores padrão caso nenhum inicializador seja fornecido. Forneça métodos public que realizam cada uma das operações a seguir:
   a) Somar dois números Racional: o resultado da adição deve ser armazenado na forma reduzida. Implemente isso como um método static.
   b) Subtrair dois números Racional: o resultado da subtração deve ser armazenado na forma reduzida. Implemente isso como um método static.
   c) Multiplicar dois números Racional: o resultado da multiplicação deve ser armazenado na forma reduzida. Implemente isso como um método static.
   d) Dividir dois números Racional: o resultado da divisão deve ser armazenado na forma reduzida. Implemente isso como um método static.
   e) Retorne uma representação String de um número Racional na forma a/b, onde a é o numerador e b é o denominador.
   f) Retorne uma representação String de um número Racional no formato de ponto flutuante. (Considere a possibilidade de fornecer capacidades de formatação que permitam que o usuário da classe especifique o número de dígitos de precisão à direita do ponto de fração decimal.)

7. (*Classe **Huge Integer***) Crie uma classe HugeInteger que utiliza um array de 40 elementos de dígitos para armazenar inteiros com até 40 dígitos. Forneça os métodos parse, toString, add e subtract. O método parse deve receber uma String, extrair cada dígito usando o método charAt e colocar o valor inteiro equivalente de cada dígito no array de inteiros. Para comparar objetos HugeInteger, forneça os métodos a seguir: isEqualTo, isNotEqualTo, isGreaterThan, isLessThan, isGreaterThanOrEqualTo e isLessThanOrEqualTo. Cada um destes é um método predicado que retorna true se o relacionamento estiver contido entre os dois objetos HugeInteger e retorna false se o relacionamento não estiver contido. Forneça um método predicado isZero. Se você se sentir ambicioso, forneça também os métodos multiply, divide e remainder. [*Observação:* valores boolean primitivos podem ser gerados como as palavras “true” ou “false” com o especificador de formato %b.]