UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Aula 10: Programação orientada a objetos [6] (Laboratório 4)

Prof. Dennis Giovani Balreira
(Material adaptado do Prof. Thiago L. T. da Silveira)

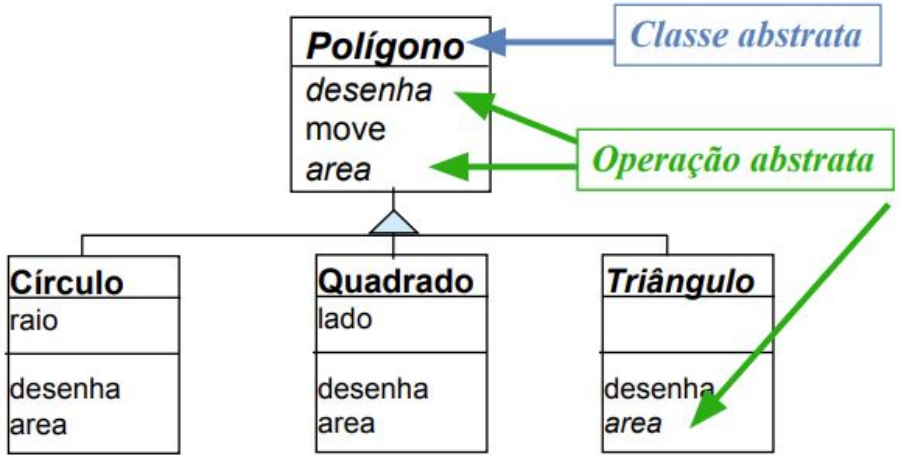
INF01120 - Técnicas de Construção de Programas

Classes abstratas e interfaces no diagrama de classes

Classes abstratas no diagrama de classes

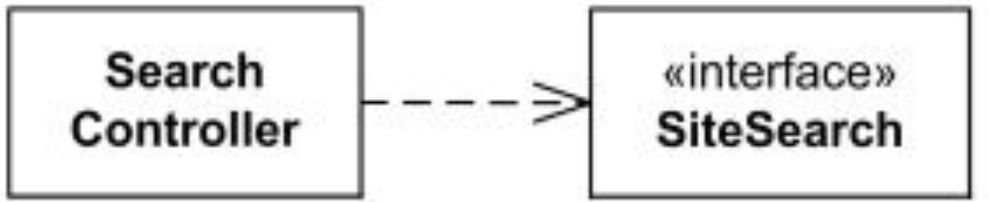
- Devem ser destacadas em *itálico*, ou colocadas com as tags {A} ou {Abstract} ao lado do nome (mesma coisa para métodos abstratos)
- Devem estar associadas a uma relação de generalização/especialização

- Ex:



Interfaces no diagrama de classes

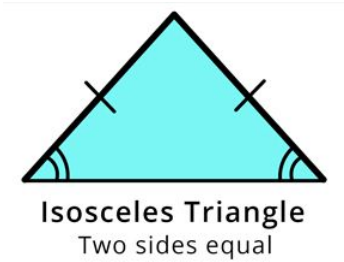
- Devem ser destacadas com o estereótipo `<<interface>>` em cima do nome
- Devem estar associadas a uma relação de dependência
- Como todos seus métodos são abstratos, não há necessidade de indicar esta informação explicitamente
- Ex:

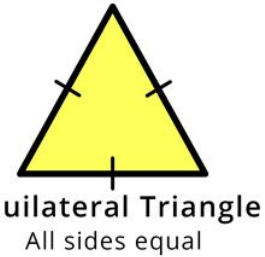


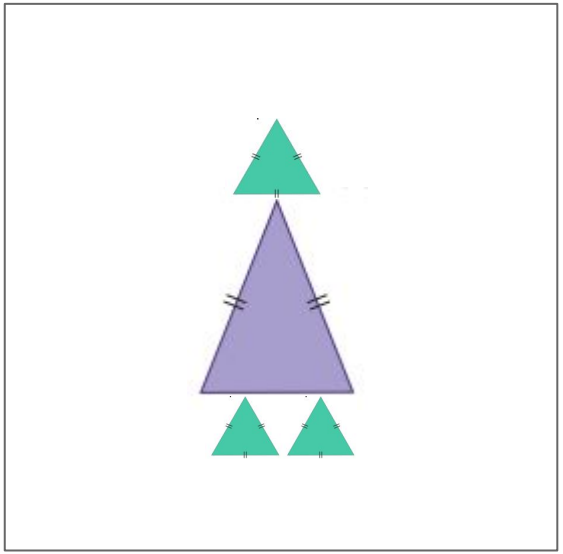
Laboratório 4

Exercício 1 - Parte 0

- Um aplicativo de matemática deseja modelar diferentes tipos de triângulos de acordo com seus lados para serem usados em um aplicativo simples para manipulação de figuras bidimensionais





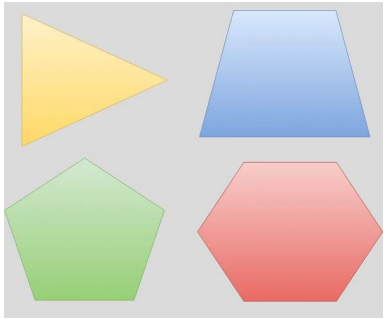


Exercício 1 - Parte 1

- O projeto deve ter uma interface `Poligono` e as seguintes classes:
  - `Triangulo`
  - `TrianguloEquilatero`
  - `TrianguloIsosceles`
  - `AplicacaoTestaPoli`

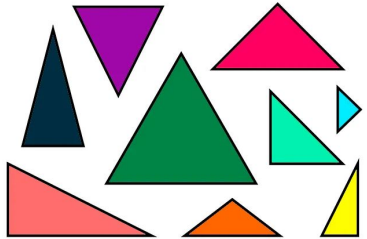
- A interface `Poligono` e as classes `Triangulo`, `TrianguloEquilatero` e `TrianguloIsosceles` devem estar em um mesmo pacote chamado "`poligonos`", cada uma em um arquivo .java
- A classe `AplicacaoTestaPoli` deve estar no pacote "`teste`"
- Comece pensando em como seria um diagrama de classes para o problema (não precisa implementar/enviar) e só depois vá para a implementação

Exercício 1 - Parte 2



- Crie uma implementação em Java, conforme instruções:
  - A interface `Polígono` deve possuir o atributo tamanho do canvas (área quadrada do desenho) definido como 100 e os métodos `calculaArea()` e `imprimeTipoPoligono()`

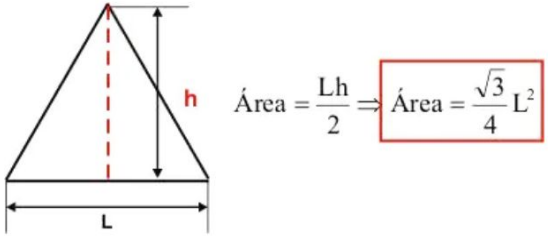
  - A classe `Triangulo` deve ser abstrata e implementar a interface `Polígono`. Implemente dois métodos chamados `calculaArea()`: um método de classe (static) que recebe como parâmetros os valores de base e altura e calcula a área desse triângulo e outro método de instância (não static) que calcula a área conforme seus atributos base e altura. O método `imprimeTipoPoligono()` deve ser definido como abstrato. A classe deve possuir os três lados como atributos privados, além de base e altura. Utilize polimorfismo estático (sobrecarga) para criar dois métodos construtores: um que recebe valores de base e altura como parâmetros e outro que recebe três valores correspondendo às medidas dos três lados como parâmetros. Implemente também os *getters* e *setters* para cada um dos atributos

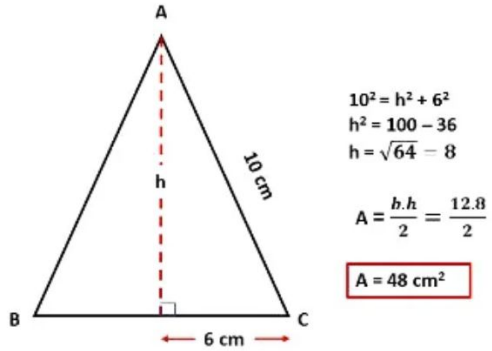


Exercício 1 - Parte 2

- Crie uma implementação em Java, conforme instruções:
  - A classe `TrianguloEquilatero` deve ser declarada como uma subclasse de `Triangulo`. Seu construtor deve chamar o construtor da superclasse que recebe os lados como parâmetros. Utilize polimorfismo dinâmico (sobrescrita/*override*) para calcular a área a partir de seus lados e para imprimir o nome do tipo de polígono

  - A classe `TrianguloIsosceles` deve ser declarada como uma subclasse de `Triangulo`. Seu construtor deve chamar o construtor da superclasse que recebe os lados como parâmetros. Utilize polimorfismo dinâmico (sobrescrita/*override*) para calcular a área e para imprimir o nome do tipo de polígono. Defina um novo método privado chamado `calculaAltura()`, que retorna a altura de um triângulo isósceles a partir dos três lados setados como atributos. Use este método como auxiliar para descobrir a altura do triângulo e assim poder setar os atributos (base e altura) e chamar o método `calculaArea()` da superclasse que devolve a área





Exercício 1 - Parte 2

- Crie uma implementação em Java, conforme instruções:
  - Observações:
    - A única classe que deve ter atributos (sem contar a constante da Interface) é a `Triangulo`, e todos devem ser privados
    - Não é necessário fazer consistência dos valores (se realmente são triângulos, se os valores são positivos, se realmente é um equilátero ou isósceles, etc.). Assuma que eles serão fornecidos corretamente
    - É permitido alterar a forma de passar os lados dos triângulos para um ArrayList ou array se desejado (em vez de 3 floats separados - um para cada lado do triângulo). Neste caso faça as alterações no main para que fique compatível com seu código, mas lembre-se de usar os mesmos valores
    - Lembre-se de utilizar a notação @Override para polimorfismo dinâmico (sobrescrita). A notação também deve ser utilizada na hora de sobrescrever os métodos abstratos da Interface

Exercício 1 - Parte 3

- A classe `AplicacaoTestaPoli` deve possuir o código `”AplicacaoTestaPoli.java”` disponível no Moodle

Exercício 1 - Entrega

- Envie um único arquivo no formato **`ZIP`** contendo a implementação do **Laboratório 4 - Exercício 1** (`Poligono.java`, `Triangulo.java`, `TrianguloEquilatero.java`, `TrianguloIsosceles.java`, `AplicacaoTestaPoli.java`)

- O nome do arquivo deve seguir o formato:

  "l4-<primeiro_nome_do_aluno>-<último_nome_do_aluno>.zip"

- Atente ao prazo de entrega do trabalho especificado no Moodle!

- Bom trabalho! Qualquer dúvida contate o professor

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Obrigado pela atenção!
Dúvidas?

Prof. Dennis Giovani Balreira
(Material adaptado dos Profs. Marcelo Pimenta e Thiago L. T. da Silveira)