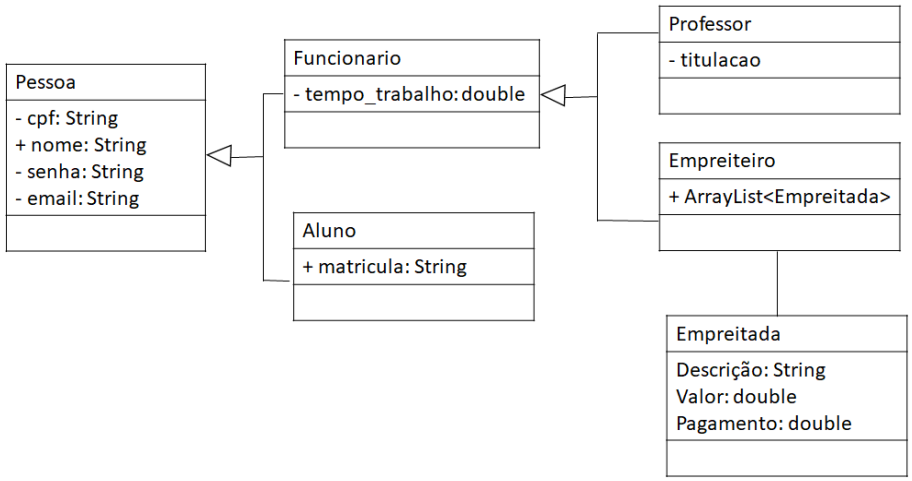
Laboratório – Herança, Polimorfismo, Classes Abstratas e Interfaces

Atividade 1: Herança e Sobrecarga de Operadores

Parte 1.1: Criação das classes usando herança

- Uma escola possui funcionários e alunos.
- Funcionários possuem cpf, nome, e-mail, senha e ano de entrada no trabalho.
- Dos alunos, é necessário armazenar nome, e-mail senha e número de matrícula.
- Professores são funcionários e sobre eles é necessário armazenar a titulação.
- Empreiteiros são funcionários que realizam ações (empreitadas) por demanda. As empreitadas possuem descrição, valor e pagamento e devem ser armazenadas no cadastro do empreiteiro.

Escreva a hierarquia de classes abaixo para representar o contexto descrito acima.



Parte 1.2: Modificadores de Acesso e Regras de Negócio

Defina os atributos mencionados abaixo como private ou protected e use métodos set para verificar as seguintes regras de negócio:

- O CPF não pode ser modificado após o cadastro.
- No construtor e ao atualizar os e-mails deve-ser verificar se os e-mails são válidos. São considerados e-mails válidos aqueles que possuem “@” e “.com”.
- Senhas são válidas se possuírem pelo menos 5 caracteres e senhas não podem ser substituídas por senhas iguais.
- O ano de entrada no trabalho deve ser maior ou igual que zero e uma vez que o objeto foi criado, este valor não pode mais ser atualizado.
- A titulação só pode receber um dos seguintes valores: “BÁSICO INCOMPLETO”, “BÁSICO”, “MÉDIO”, “GRADUAÇÃO”, “MESTRADO”, “DOUTORADO”.

Parte 1.3: Sobrescrita (Overriding) de Métodos

- O salário base dos funcionários é de R$ 1000.00 reais por ano de trabalho.

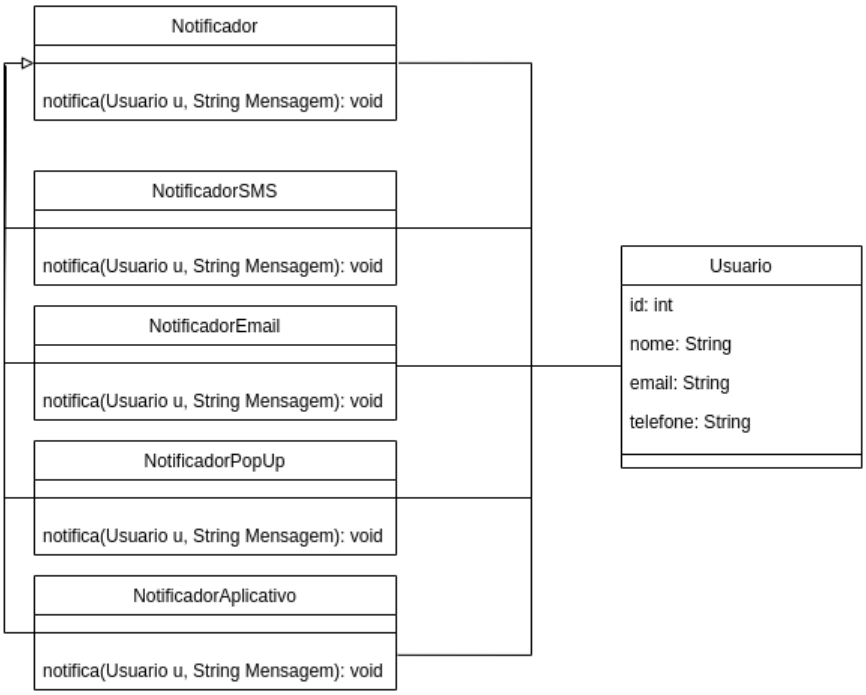
- Professores recebem além do salário base uma premiação por titulação de +1000 se tem gradução, +2000 se tem mestrado e +4000 se tem doutorado.
- Empreiteiros recebem o salário base de todos os funcionários mais os pagamentos pelas empreitadas realizadas.
- Construtores.
- toString.

Parte 1.4: Polimorfismo em Tempo de Execução

- Atribuição de Professor à Funcionário e Empreiteiro à Funcionário e demonstração da invocação de métodos.
- Criação de ArrayLists de Funcionários. Listar pagamentos.

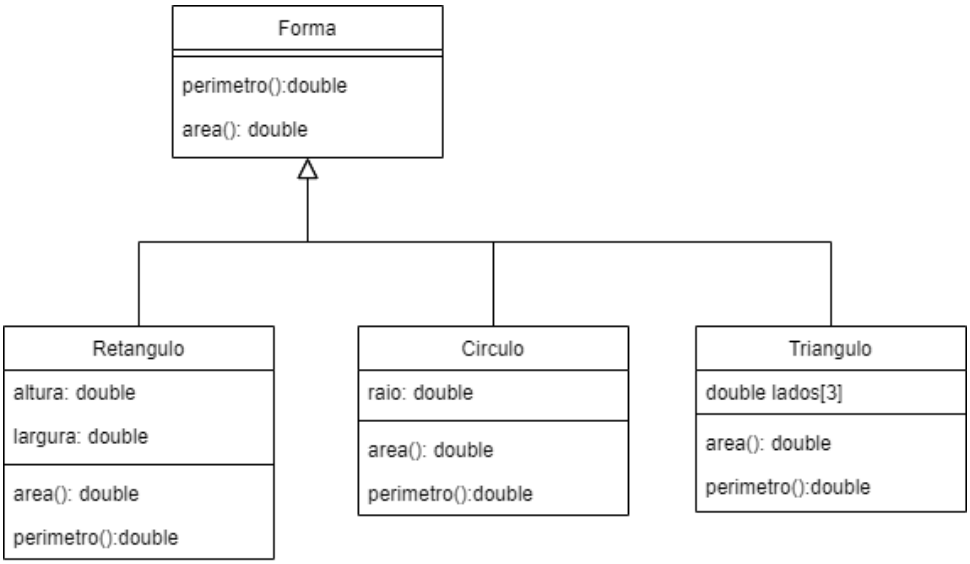
Atividade 2: Classes Abstratas e Interfaces

2) Em aplicativos modernos, ao realizar eventos no sistema, usuários são notificados por diversos meios como mensagens pop-up, e-mail, SMS e alertas no aplicativo. Implemente as classes usuário e sistema indicados abaixo, assim como a hierarquia de classes de notificação.



Atividade 3: Classes Abstratas e Interfaces

1) Polígonos possuem formas diferentes de calcular área e perímetro. Implemente a hierarquia de classes abaixo e faça um programa que crie diferentes polígonos e armazene-os em um ArrayList<Forma>. Utilizando um comando de repetição que itere sobre os elementos do array, mostre na tela as áreas e perímetros de todos os polígonos.



Para calcular a área do triângulo a partir de seus lados, use a fórmula de Heron:

Outras fórmulas para calcular a área de um triângulo

Existe outro método para calcular a área de triângulos conhecido como **fórmula de Heron**. Utilizamos essa fórmula quando conhecemos apenas a medida dos lados do triângulo, mas não conhecemos a altura. Para aplicar a fórmula de Heron do triângulo de lados a, b e c, **primeiro calculamos o semiperímetro**, ou seja, metade do perímetro do triângulo.

$$p = \frac{a+b+c}{2}$$

Conhecendo o valor do semiperímetro, basta utilizar a fórmula:

$$A = \sqrt{p(p-a)(p-b)(p-c)}$$

Figure 1. Extraída de https://mundoeducacao.uol.com.br/matematica/area-triangulo.htm

Atividade 4: Implementação de Interfaces

1) **Implementação de Interfaces**: Considere o problema de exibir os participantes de um concurso de forma ordenada pela nota.
   a. Crie uma classe Participante com os atributos nome, cpf e nota. Escreve o método toString e um construtor.
   b. Crie uma classe App com o método main. Nesta classe, crie um ArrayList para armazenar os participantes do concurso e adicione 3 participantes com notas diferentes.
   c. Para ordenar a lista usando as notas, pode ser usado o método sort existente na classe Collections. Para usá-la, primeiro importe no início do arquivo “java.util.Collections”. Se o ArrayList se chamar “participantes”, a ordenação pode ser feita usando “Collections.sort(participantes)”. Contudo, se você tentar executar o programa, será exibido um erro dizendo que o ArrayList não pode ser usado como entrada para o sort. Isto acontece porque para ordenar a lista, é necessário dizer como os Participantes devem ser comparados, ou seja, qual critério deve ser usado para ordená-los. Esta restrição é adicionada definindo que apenas classes que implementam a interface Comparable podem ser usados como entrada para o sort.

d. Atualize a classe Participante, diga que ela implementa a interface Comparable<Participante> e defina o método compareTo como indicado abaixo. Verifique que após esta mudança, o programa passa a funcionar.

```
public class Participante implements Comparable<Participante>
{
    String cpf;
    String nome;
    int nota;

    public Participante(String cpf, String nome, int nota) {
        this.cpf = cpf;
        this.nome = nome;
        this.nota = nota;
    }

    public String toString() {
        return "Pessoa(" + nome + ", cpf=" + cpf + ", nota=" +
nota + ")";
    }

    public int compareTo(Participante p) {
        return p.nota - nota;
    }
}
```