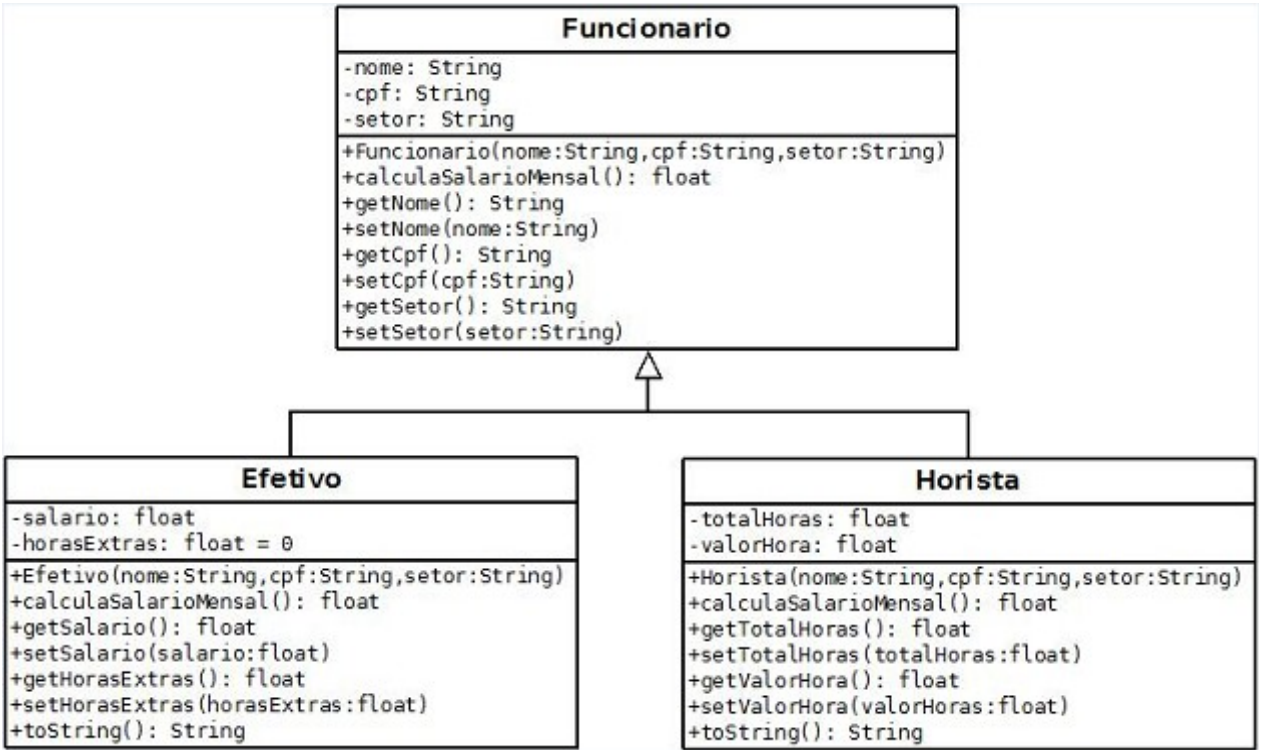
UFABC	Universidade Federal do ABC Disciplina: Programação Orientada a Objetos 2Q-2019

Atividade 5: Herança / Polimorfismo

Obs.: Para entrega: a pasta src com os arquivos .java de cada um dos projetos (os exercícios relacionados podem estar no mesmo projeto). Renomeie cada pasta de modo a identificar o exercício correspondente. Submeta uma pasta compactada na atividade do Tidia.

Exercício 1

Os funcionários de uma empresa dividem-se em duas categorias: **efetivos** e **horistas**, conforme a hierarquia de classes abaixo:



- Os funcionários efetivos possuem um salário fixo para 40 horas de trabalho semanal. Caso exceda esse número de horas, o funcionário receberá como horas extras (50% de acréscimo no valor da hora normal por hora extra).
- Os funcionários horistas recebem um valor estipulado por hora. Caso o total de horas ultrapasse 160 horas, as horas excedentes serão pagas com 40% de acréscimo.
- Todo funcionário possui nome, cpf e setor em que trabalha e um método para calcular o salário mensal.
- O método **toString()** (da superclasse Object) deve ser sobrescrito nas subclassses para retornar uma String com os dados do funcionário e o seu salário mensal.

Crie um pacote **funcionario** e implemente esta hierarquia de classes.
Escreva o programa principal (no pacote *default*) e crie um vetor com 5 funcionários. Crie os funcionários e armazene-os no vetor e imprima seus dados.

Exercício 2 (Prof. Paulo Pisani - adaptação)

Você ficou encarregado de criar um sistema para calcular a área e o perímetro de **figuras planas convexas**. Seu trabalho é:

- Criar uma hierarquia de classes que inclua **círculo**, **triângulo**, **quadrado** e **retângulo**.
- As figuras triângulo, quadrado e retângulo devem ser representadas pelo comprimento de seus lados. O círculo será representado pelo seu raio.
- Crie métodos para retornar as medidas de **área** e **perímetro** (note que para o triângulo, é possível calcular a área conhecendo apenas seus lados com a fórmula de Heron).
- Sobrecarregue o método **toString()** para retornar informação detalhada sobre cada figura (tipo, área e perímetro).

Atenção: todas as figuras devem estar no pacote **geometria**.

Crie um novo pacote chamado **criafiguras** e crie o programa principal que deve ter um vetor de figuras planas. Nesse vetor, instancie uma figura de cada tipo e imprima na tela as informações de cada figura.

Exercício 3 (Prof. Paulo Pisani - adaptação)

Crie uma hierarquia de classes para o seguinte problema (no pacote **copaamerica**):

- Um time de futebol possui vários jogadores, que podem ser: atacante, lateral ou goleiro.
- Todo jogador possui um número na camisa (definido pelo construtor) e um método para imprimir sua posição (e.g. atacante, goleiro, etc). Um lateral pode ficar na direita ou esquerda do campo (isso é definido por parâmetro no construtor).
- No time de futebol, haverá um vetor de jogadores. Esse vetor deve ser armazenado em ordem crescente de número da camisa;
- Um time de futebol possui um método para substituir jogador:
  - Este método deve fazer a substituição e manter o vetor de jogadores ordenado por número da camisa.

Para este exercício, vamos considerar que um time pode ter entre 1 e 11 jogadores.
Adicione um método para imprimir a escalação atual também.

No programa principal (que ficará no pacote **default**):

- Leia 3 jogadores e insira no Time ABC
- Leia outros 3 jogadores e insira no Time DEF
- Imprima as duas escalações
- Leia o número de um jogador (que existe em ABC) e outro que entrará (como substituição) no time
- Faça a substituição e imprima as duas escalações

Exercício extra: crie um método verificaTime(), que retorna true se o time é válido (ou seja, 11 jogadores, sendo apenas 1 goleiro).

Exercício 4 (Prof. Paulo Pisani - adaptação)

Usando as classes do exercício anterior, o objetivo deste exercício é criar uma **lista ligada** de figuras planas. Esta lista ligada deve conter um método que permite a **inserção ordenada crescente** das figuras geométricas, respeitando a seguinte ordem:

- os objetos são ordenados de acordo com o número de vértices que os compõe. Define-se o círculo como a maior figura por este critério. Quando os objetos empatam em número de vértices, usa-se como critério de desempate, o tamanho da área e em seguida o tamanho do perímetro. Isto é, se duas figuras são retângulos, então definimos a ordem por suas áreas e em caso de empate, pelo perímetro.

- Para que a lista ligada saiba comparar as figuras, implemente um método (na classe FiguraPlana) que recebe como argumento um objeto FiguraPlana e retorna -1 caso a figura receptora da chamada seja menor que a figura recebida por argumento, 0 caso sejam iguais e 1 se a receptora for maior.
- A lista ligada deve ter um método que imprima na tela as figuras na ordem que aparecem na lista.
- Implemente um programa principal para testar esta ListaLigada.