Programa 1 (valendo nota) – P1n

1) Desenvolva uma classe chamada *ValidaCPF*, com métodos **de classe**, adaptada da classe apresentada em https://www.devmedia.com.br/validando-o-cpf-em-uma-aplicacao-java/22097

Além do método para validação e formatação de saída da classe original, no método *isCPF*, você também deve verificar se o *String* passado é 'convertível' em um CPF válido. Ou seja, se ele não é uma sequência de caracteres aleatória, ou se não é um dos formatos válidos como indicado à seguir..

Adicione um último método de classe, *toLong* que receba um *String* validado pelo método *isCPF* e devolva um *long* com o CPF.

Atenção.Você deverá prever que o usuário poderá passar o CPF de "qualquer" jeito/formato, com e sem ponto.

Formatos válidos: `12345678901, 123.456.789-01, 123.456.789/01.`

Formatos inválidos: `123 456 789 01, 123.456.789 01, 123a456b789c01, 123 456, 123#45&67O7890-1`, etc.

2) Crie uma classe *ValidaData*, também com métodos de classe, para verificar (ou seja, retorna um *boolean*) o dia, o mês e o ano.

Os métodos: *isDia*, *isMes*, *isAno*. Os argumentos de entrada desses métodos podem ser do tipo *int* ou do tipo *String*.

Para o **dia**, considerem **válidos**, 1 até 31, para o **ano**, válidos do (ano corrente -120) até o ano corrente (isso pode ser obtido pela própria classe GregorianCalendar). Assim, a aplicação nunca fica obsoleta. "0" a direita devem ser absorvidos na conversão, ou seja, "01" ou "1" são convertidos pra o *int* 1.

O mês pode ser utilizado como *int* ou *String* de 1 a 12 ("1" a "12", ou ainda "01", "02" ...). Mas também pode ser entrado por extenso, em portugues: "janeiro", "fevereiro", "marco" (não use o cedilha) "abril" ... "novembro", "dezembro". Veja o conceito de *enum*, fica muito bom. (http://152.92.236.11/javatutor/java/javaOO/enum.html)

O problema, sabemos, é como validar o dia com o mês adequadamente, Por isso, também faça o método *isDataValida* que valida a data completa (dia, mês e ano). A data tem que ser válida, ou seja, mesmo que você tenha passado individulamente pelo *isDia*, o dia 30 não vale com o mês 2. O método pode ter duas assinaturas, uma com os argumentos em *int* e outra com *String*.

3) Crie uma classe *Pessoa* com o seguinte:

Campos de instância:

- *nome*, da classe String;
- *sobreNome*, da classe String;
- *dataNasc*, da classe *GregorianCalende*r (veja a documentação), usando o método `GregorianCalendar(int year, int month, int dayOfMonth)`

  Alternativamente *dataNasc* pode ser da classe `LocalDate`. Não temos documentação desta classe em nossa imagem na máquina 11. Portanto, avaliem se vale apena usar. O método `LocalDate.of(int year, Month month, int dayOfMonth)` permite criar uma instância de objeto. Observem que este não é um construtor, mas um método de classe.

- *numCPF*, do tipo *long*.
- *peso*, do tipo *float*;
- *altura*, do tipo *float*.

Métodos de instância:

- Um construtor com parâmetros de entrada recebendo *nome*, *sobreNome*, dia, mês e ano de nascimento.
- Um construtor com os parâmetros do construtor anterior, mais o CPF, peso e altura.
- Métodos *get* e set *apropriados*.
- *toString* que devolva as informações do objeto com formatação.

Um último requisito para Pessoa:

A cada objeto da classe Pessoa instanciado, o número de objetos deve ser atualizado. Ou seja, você vai ter que "bolar" uma forma de se contar quantas instâncias foram criadas dentro desta classe. Um método que pode ser chamado mesmo sem um objeto criado (interprete isso) que devolva quantos objetos foram criados, `int numPessoas()`. Não deve ser implementado solto na main. Dá o seu jeito!

4) Desenvolva duas classes, *Mulher* e *Homem*, que herdam de *Pessoa*. Tecnicamente, quando se fala herança, ou que "estendam", deve ser interpretado como herança de classe.

Crie os métodos construtores e o método *toString*. No método *toString* deve aparecer ainformação do gênero (homem ou mulher) e **a idade**. O programa não deve perguntar a idade da pessoa. Deve ser exibida a idade de acordo com a data de nascimento.

Observação: Você TEM que usar adequadamente a modularidade, encapsulamento, reuso e herança.

5) Faça um programa, ou seja, a classe principal, P1nX, com pelo menos – o mínimo, o seguinte:

Um campo com um *array* para armazenar objetos da classe Mulher e Homem;

Ao ser iniciado o programa deve ler como parâmetros de linha de comando as informações para criar um objeto da classe Mulher ou Homem.

Erros nestes parâmetros devem ser detectados, expostos e NÃO podem abortar o programa. Se o usuário errou algum parâmetro, ou ordem que o seu prorgama espera, aponte o erro e exiba um "help" com a sintaxe correta.

```
java P1nX <genero> <nome> <sobre> <dia> <mes> ano> <CPF> <peso> <altura>
```

Aqui precisa dizer para o usuário, se ele não souber, como é para especificar o genero.

Se os parâmetros estiverem corretos, deve-se criar o objeto de acordo com os parâmetros de linha de comando e exibir as informações deste objeto. Esse objeto não precisa ser colocar no array abaixo.

Se o usuário acertou os parâmetros ou não, isto é, mesmo que o primeiro objeto não tenha sido criado, na sequência, pergunte ao usuário quantos elementos adicionais ele quer criar, usando as classes deentrada de dados (Scanner, InputStreamReader/BufferedReader) pelo teclado.

Crie a instância do *array neste* momento, não antes, e com base nesta informação.

Desenvolva uma rotina para ler os dados, criar o objeto e armazenar no *array*. Você precisa perguntar se o usuário quer entrar com dados de uma Mulher ou de um Homem.

Controle o número de elementos adequadamente. **Atenção:** o usuário pode decidir inserir menos elementos do que o número definido antes. Controle isso também. Por exemplo, se o usuário pressionar ENTER o loop de leitura para. Ou seja, não é para chamar um erro.

Quando a rotina de entrada terminar, exiba todos os elementos do *array*. Se você fez uso das técnicas recomendadas, isso é simples. Use o método que devolve o número de objetos criados a seu favor, para verificar ou iterar pelo *array* corretamente.

Depois disso, exiba o número de objetos de cada gênero e o programa termina. Se você contar isso numa variável separada, vai estar errado neste exercício! Não use uma variável que é incrementada quando há a criação de um novo objeto. Use o operador ***instanceof*** para fazer esta contagem, mas somente nesta rotina final.

Atenção: Todas as entradas devem ser criticadas. Não podem haver dados faltantes ou inconsistentes. Exemplos: tamanho de *array* negativo, 12345 ser o nome da pessoa, dia 45, mês 17, ano 2074, etc. O programa não pode abortar com erros de entrada do usuário em nenhum caso, em nenhum ponto da rotina. Seu programa tem que verificar essas entradas antes de criar as instâncias, etc.

Entradas inválidas devem disparar uma rotina onde o programa avisa do erro (ou seja, o erro foi detectado) e pede uma correção (ou seja o erro é "tratado"). O bom senso deve prevalecer.

Para esta 1ª experiência, o programa principal pode ser codificado todo no método *main* (sabendo que isso não é incentivado no futuro).