/*
ANÁLISE DO CODE(linguagem natural)
*/

A classe **CheckStrength**, **é uma classe onde irá me informar, a partir de uma string(senha), se a mesma corresponde**
a um tipo de "senha" fraca, normal, forte, muito forte ou extremamente forte!

Inicialmente temos uma classe pública enumerável, onde temos a respectiva característica da senha como forte ou fraca por exemplo.

É criado um dicionário com palavras comuns de senhas, onde, se o usuário colocar alguma senha que esteja empregada nesse dicionário,
a senha será considerada fraca.

Uma das rotinas configura em analisar caractere por caractere de modo a retornar se aquele caractere é um número ou uma letras maiúscula
ou minúscula ou se é um caractere que não corresponde a nenhum dos anteriores, aqui, mencionados.

Para que eu tenha um bom *feedback* de uma senha forte, existe uma rotina que contabiliza o n° de caracteres dentro da *String*(senha), até porque, eu preciso de no mínimo 8 caracteres para eu conseguir fazer uma senha. Não se pode haver uma senha com menos de 8 caracteres

A rotina checkPasswordStrength irá verificar a força da senha(da *String* entrada).
A primeiro a rotina se volta a classe StringUtil, onde irá, nesse caso(StringUtils.equalsNull), receber uma *String*(minha senha)
e verificar se existe algo lá dentro como apenas espaços em bracos, vazio ou nulo e diz que a mesma está vazia.
Após o primeiro if, ele cria um contador de níveis para adicionar pontos a determinado caractere (dependendo do que exista na *String* ele vai adicionando ou removendo sua pontuação), visando mensurar uma pontuação que será retornada ao final da função,
para que depois, quando obtiver toda a pontuação, faça um calculo e determine a força da senha, inclusive. Ou seja, todas as regras referentes à força da senha serão tratados dentro dessa rotina.

Na rotina **getPasswordLevel** utiliza-se a rotina **checkPasswordStrength ,** para que eu possa ter um valor e desse valor eu analiso por intermédio de um *switch* e dependendo do valor da variável *level*, eu retorno o nível atribuído como segurança para a senha apresentada em particular.

Na classe **StringUtils, tenho métodos que tratam casos especiais em uma *String*. Calculando o tamanho de um inteiro, verificando se uma *String* é composta apenas por determinado**

caractere ou dígito ou mesmo se está contém apenas espaços em branco, se é vazia ou nula. Essa classe, juntamente com seus métodos, são de formidável importância para a classe **CheckStrength que visa tratar casos específicos, segundo suas regras, e nivelar determinada String de modo a obter o nível de segurança desta, getPasswordLevel.**

/
MUDANÇAS FEITAS NO CODE(qualidade do code)
*/
Nome de variáveis estão com parâmetros inconsistentes, sem boa assimilação. Foram renomeados para melhor compreensão do algoritmo, tanto na classe CheckStrength como na Main e StringUtils.
A rotina **checkPasswordStrength** tornou-se *protected*, uma vez que ela só me retorna o nível da senha e esse valor é tratado na rotina **getPasswordLevel,** e nos meus casos de teste**,** que, por se tratar de uma get, irá retornar a força da minha senha na main que por acaso também foi modificado, pois, estava chamando a rotina **checkPasswordStrength** que só me retornava um valor inteiro que não faz o menor sentido.

Na classe StringUtils, todos os métodos que serão utilizados pela classe CheckStrength, foram alteradas para protected para que apenas as classes e subclasses do pacote pertencente a este, possam utilizá-lo.

Foram adicionados novos comentários e modificados alguns existentes. Fora feito indentação do código. Diminuído o tamanho de linhas do método **checkPasswordStrength** , criando dois novos métodos(**checkDateAndDictionary, breakString**) que contém, basicamente, as partes de código retiradas do método anterior.