**Zadanie č.3**

Úvod do počítačovej bezpečnosti

Implementácia správy používateľských hesiel

2016/2017

**Autor: Michal Drahovský, Ján Dzvoník**

Obsah

[1 Postup riešnia 3](#_Toc467076473)

[1.1 Funkcie na generovanie a aplikovanie saltu 3](#_Toc467076474)

[1.2 Funkcie na zahashovanie saltového hesla 3](#_Toc467076475)

[1.3 Funkcie na kontrolu zložitosti hesla 4](#_Toc467076476)

[1.4 Funkcie na vytvorenie časového odstupu medzi pokusmi o prihlásenie 6](#_Toc467076477)

[2 Záver 8](#_Toc467076478)

# Postup riešnia

## Funkcie na generovanie a aplikovanie saltu

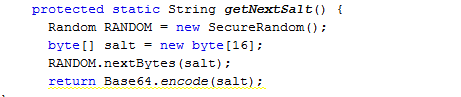
Salt je náhodne vygenerovaný reťazec ktorý sa prikladá k nezahashovanému heslu aby sa skomplikovali útoky zamerané na zlomenie hesla. Salt by mal byť pre každé nové heslo náhodný a unikátny. Vygenerovaný salt je následné potrebné si zapamätať aby si sním ďalej dalo overiť heslo.

Funkcie na generovanie saltu sme vytvorili v Classe Security. Vygenerovaný salt sa následne aplikuje pri hashovaní hesla a zapíše sa do súboru hesla.txt do stĺpca za zahashované heslo. Pre každé heslo sa generuje unikátny a náhodný salt.

Na encodovanie saltu sme využili knižnicu com.sun.org.apache.xerces.internal.impl.dv.util.Base64;

Na generovanie saltu sme využili knižnicu java.security.SecureRandom, java.util.Random.

Ukážka zdrojového kódu funkcie, ktorá generuje salt.



Ukážka zdrojového kódu, ktorý aplikuje salt a zavolá funkciu na zahashovanie hesla.



Vygenerovaný salt može vyzerať napríklad: uEtD3IBQ4XWSk1t1UfFOkA==

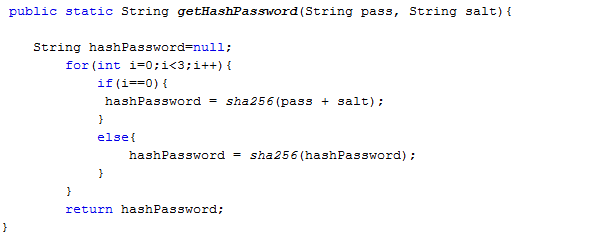
## Funkcie na zahashovanie saltového hesla

Z bezpečnostných dôvodov je potrebné jednotlivé heslá ukladať v nečitateľnej forme. Na tento problém sa používajú asymetrické hashovacie funkcie. To znamená, že sa nedajú spätne dešifrovať. V súčastnosti sa za bezpečnú hashovaciu funkciu považuje aj SHA-256. Túto funkciu sme si vybrali aj v našom zadaní.

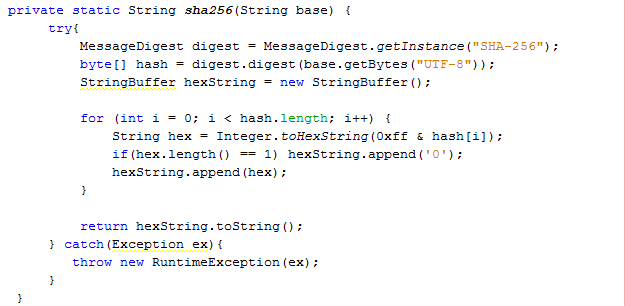
Funkcie na zahashovanie saltového hesla sme doprogramovali v Classe Security. Táto funkcia zahashuje saltové heslo funkciou SHA256 a následne tento hash znova zahashuje ešte 2 krát taktiež funkciou SHA256.

Na hashovaciu funkciu SHA256 sme využili knižnicu java.security.MessageDigest;

Ukážka kódu generovania hesla



Ukážka kódu funkcie ktorá hashuje heslo funkciou SHA256



Zahashované heslo môže byť v tvare: 105544b0cfa33a1910b32c7cf910f705b5b66b3778fe4fdf8bdef75c0860fda1

## Funkcie na kontrolu zložitosti hesla

Kontrola zložitosti hesla je dôležitá kvôli ochrane užívateľského účtu. Ak by užívateľ nebol prinútený si zvoliť bezpečné heslo niektorí lenivejší užívatelia, ktorým sa nechce pamätať si heslá by volili veľmi slabé heslá. Takéto slabé heslá sú následne ľahko prelomiteľné pomocou brute force alebo slovníkovým útokom čo predstavuje vysokú bezpečnostnú hrozbu. Následne by mohli prísť o svoj účet alebo by mohli byť zneužité informácie z účtu.

Funkcie na kontrolu zložitosti hesla sme doprogramovali v Classe Security. Užívateľ je prinútený zvoliť heslo tak aby spĺňalo zadefinované pravidlá pre heslo.

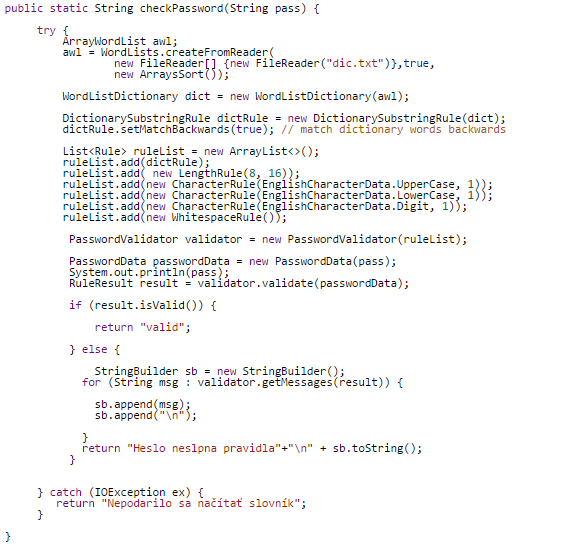
Tieto pravidlá sú

* Dĺžka hesla musí byť od 8 po 16 znakov
* Heslo musí obsahovať aspoň 1 veľké písmeno
* Heslo musí obsahovať aspoň 1 malé písmeno
* Heslo musí obsahovať aspoň 1 číslicu

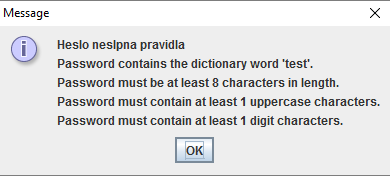
Heslo, ktoré síce spĺňa tieto požiadavky, ale takisto môže byť slabé. Preto heslo, ktoré užívateľ zvolí je ešte kontrolované slovníkom či neobsahuje často používané slová v heslách. Ak heslo spĺňa všetky požiadavky a neobsahuje slovo zo slovníka užívateľ je úspešne registrovaný.

Pre kontrolu zložitosti hesla sme použili knižnicu - passay-1.2.0

Ukážka kódu kontroly zložitosti hesla



Ak požívateľ nesplní požiadavky je upozornení, ktoré pravidlá nespĺňa.



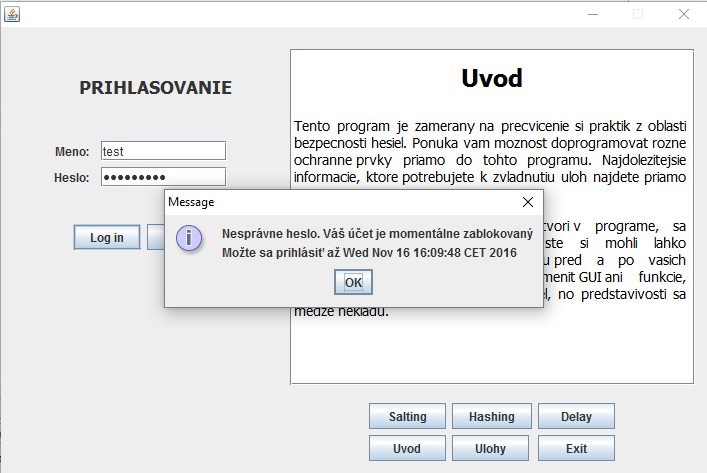
## Funkcie na vytvorenie časového odstupu medzi pokusmi o prihlásenie

Vytvorenie časového odstupu medzi jednotlivými nesprávnymi pokusmi sa používa aby sa predišlo prelomeniu hesla technikou brute force. Jednotlivé pokusy sa zaznamenávajú v programe a následne sa obmedzí prístup k aplikácií pre zadaného používateľa. Ak sa používateľ prihlasuje veľa krát so zlým heslom jeho čas dokedy sa nemôžu prihlásiť sa zväčšuje o niekoľko sekúnd až minút. Pre užívateľa niekoľko sekúnd nepredstavuje veľké množstvo času ale pri počítači riadenom útoku sa predĺži útok o podstatne viacej času.

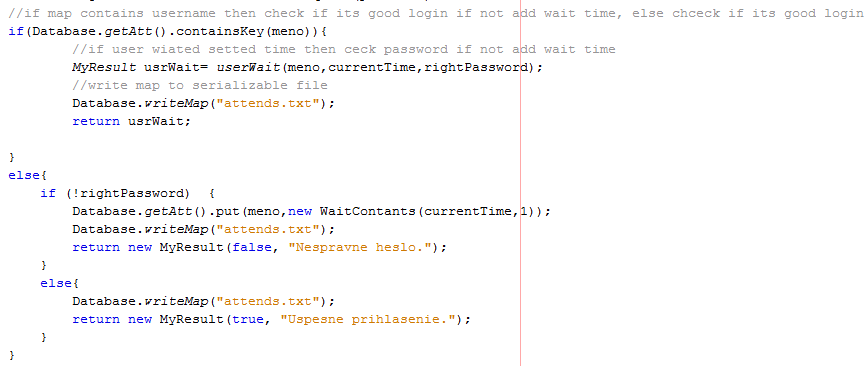
Funkcie, ktoré kontrolujú správnosť údajov a vytvárajú časový odstup medzi prihláseniami sme vytvorili v Class Login a funkcie, ktoré následne ukladajú informácie o jednotlivých uživateľoch a ich pokusoch o prihlásenie v Class Database. Ďalej sme si vytvorili pomocnú class WaitContants. Táto Classa iba uchováva čas vo formáte Date a počet sekúnd o ktoré sa zvyšuje suspendovanie užívateľského konta.

Ak sa používateľ prihlási so zlým heslom, je pridaný do mapy kde ako kľúč je login a hodnota je inštancia class WaitConstant so dátumom a časom dokedy musí čakať a počet sekúnd koľko má čakať. Ak používateľ znova zadá zlé heslo aplikácia sa pozrie do mapy či sa tam používateľ nachádza. Ak áno tak okontroluje či jeho čas, dokedy sa nemôže prihlásiť je menši ako súčasný čas. Ak sa nenachádza v mape tak sa používateľa pokúsy prihlásiť. Ak používateľova suspendácia vypršala tak sa ho pokúsi prihlásiť. Ak sa znova zle prihlási tak jeho čas sespendovania sa navýši o počet sekúnd, ktoré mu prislúchajú. Počet sekúnd koľko má čakať a čas suspendovania sa vždy zvyšuje pri každom zlom pokuse o prihlásenie alebo pri pokuse o prihlásenie pred vypršaním suspendácie. Mapa s používateľmi je vždy aktualizovaná a zapisovaná do databázy(súboru), pri každom pokuse o prihlásenie aby sa predišlo tomu, že ak používateľ vypne aplikáciu jeho suspendácia by sa stratila. Preto pri spustení aplikácie si načíta dáta zo súboru (attends.txt) a aktualizuje mapu.

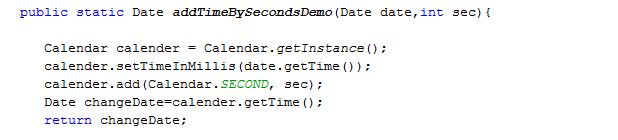
Ak sa užívateľ prihlasuje počas suspendácie je upozornení na to do kedy sa nemôže prihlásiť.



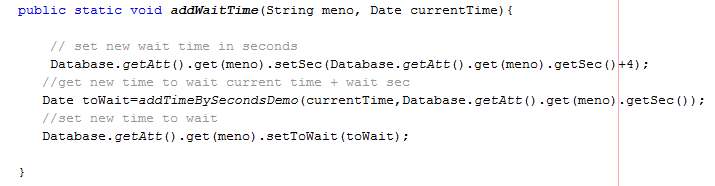
Ukážka kódu overovania prihlásenia



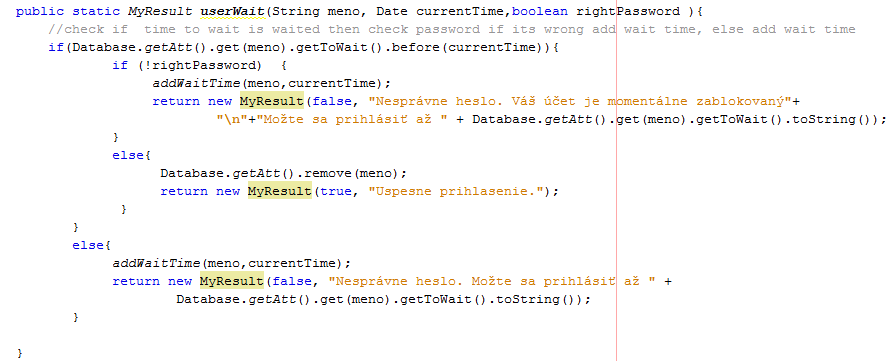
Ukážka kódu funkcie, ktorá pridá k súčasnému dátumu počet sekúnd ktoré ma užívateľ čakať



Ukážka kódu funkcie, ktorá aktualizuje čas suspendácie používateľa



Ukážka kódu funkcie, ktorá overí či používateľ čakal zadaný suspendačný čas



# Záver

Je potrebné dodržiavať určité bezpečnostné pravidla ak chceme ochrániť našu aplikáciu pre únikom informácií. Preto aj pri tvorbe našej aplikácie sme sa snažili dodržiavať zadané bezpečnostné pravidlá. Funkcie, ktoré bolo potrebné doprogramovať sme doprogramovali aby sme našu aplikáciu ochránili pred ľahkým zlomením hesla účtov používateľov hackermi.