Architektúra systému Musipher k dátumu 29.5. 2020.

Obsah

1 Funkcionality jednotlivých tried

1.1 Adder

Trieda s viacero funkcionalitami, ktoré sú používané pri komponovaní a dekomponovaní hudby. Pri komponovaní na základe vstupu generuje tóninu, progresiu akordov, akordické a rámcové tóny. Pri dekomponovaní parsuje JFugue formát označenia nôt na numerický. Extrahuje z melódie tóninu, jednotlivé akordy a progresiu akordov.

1.2 AESCypher

Testovacia trieda, nie je volaná.

1.3 CompositionModule

Modul zodpovedný za volanie jednotlivých tried, ktoré spracovávajú vstup pri komponovaní a dekomponovaní hudby. Pri hudobnom výstupe vygeneruje MIDI skladbu uloženú v zdrojovom priečinku s názvom „miusik.mid“.

1.4 Cryptomodule

Modul zodpovedný za šifrovacie a dešifrovacie operácie. **Šifrovacie:** Vstup najprv prevedie z OT na Base32 štandard. Potom prázdnym znakom „a“ vyplní minimálnu požadovanú veľkosť biov pre blok AES 128. Bity z OT prevedie na **byte array,** ktorý je zašifrovaný pomocou AES 128 na základe kľúča. **Dešifrovacie (chybné):** Vstup doplní o „0“ bity, tak aby bola veľkosť deliteľná 8 a vstup sa dal previesť na **byte array.** Vsup prevedie na byte array, a dešifruje pomocou kľúča.

1.5 Main

Volá jednolivé moduly, vypisuje vstup a výstup.

1.6 MelodyAutomaton

Archivovaná trieda nepoužívaná.

1.7 MelodyAutomaton_2

Zo vstupu skomponuje melódiu, vstupuje binárny reťazec typu **String** vystupuje melódia vo formáte JFugue.

1.8 MelodyDecoder

Dekomponuje melódiu, na vstupe je hudba reprezentovaná numericky, na výstupe binárny reťazec typu **String.**

1.9 Midi_Sequencer

Archivovaná trieda, nepoužitá.

1.10 MyPattern

Archivovaná trieda, nepoužitá.

1.11 RhytmAutomaton

Archivovaná trieda, nepoužitá.

1.12 RhytmAutomaton_2

Úlohou triedy je vytvoriť jednotlivé rytmické skupiny pre periódu a rozložiť ich na konkrétne miesta. Vstupom je binárny reťazec, výstupom názvoslovie dĺžok tónov vo formu JFugue.

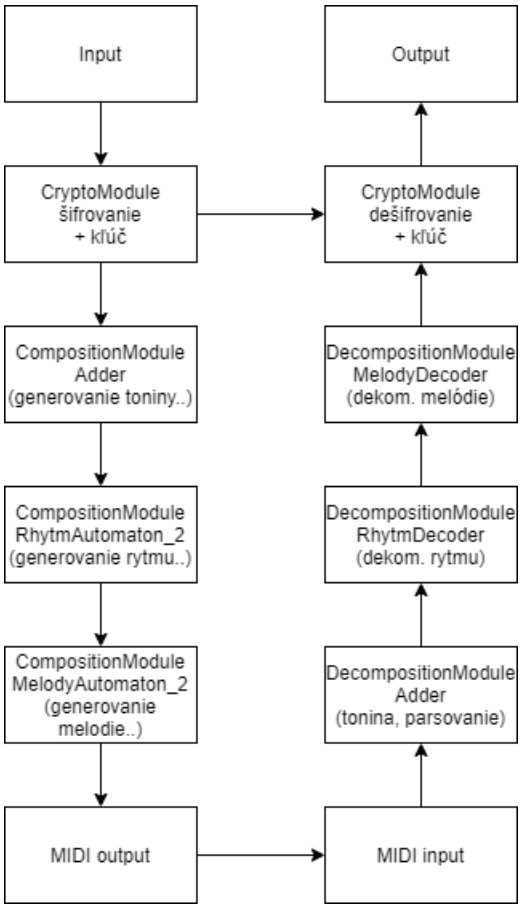
1.13 RhytmDecoder

Pri dekomponovaní z melódie získa rytmické označenia a pre jednotlivé rytmické skupiny vráti binárny výstup.

2 Použitie systému

2.1 UML Diagram

V ideálnom prípade je Input rovný Outputu. Medzi volaním CryptoModule je prenesený kľúč.



2.2 Main funkcia

Input je zadaný v premennej **String input.** Hudobný výstup je vygenerovaný v priečinku, v ktorom sa nachádza projekt. Zmysluplný presun utajeného kľúča nie je implementovaný, posúva sa iba v zdrojovom kóde. V Maine je implementovaná funkcia, ktorá porovná reťazce **a** a **b** či sa zhodujú. Reťazec **a** je binárny reťazec, pred tým ako je z neho vytvorená hudba. Reťazec **b** je binárny reťazec vytvorený na základe dekomponovania hudby.

```java
public static void main(String[] args) {

    CryptoModule cryptoModule = new CryptoModule();
    CompositionModule compositionModule = new CompositionModule();
    String input = "pavolsobota";
    String a;
    String b;
    try {
        String Strinput = cryptoModule.encryptData(input);
        SecretKey secretKey = cryptoModule.getSecretKey();
        System.out.println("toto je secretKey "+secretKey.toString());
        a= Strinput;
        compositionModule.compositionModule(Strinput);
        Strinput=compositionModule.decompositionModule();
        b=Strinput;
        for(int i = 0; i < b.length();i++){
            if(a.toCharArray()[i]==b.toCharArray()[i]){
                System.out.print("\u001B[30m"+a.toCharArray()[i]);
            }else{
                System.out.print("\u001B[31m"+b.toCharArray()[i]);
            }

        }
        System.out.println();

        String openText = cryptoModule.decryptData(Strinput,secretKey);
        System.out.println("Vysledny output: ");
        System.out.println(openText);
    }catch(Exception e){
        System.out.println(e.getLocalizedMessage());
    }
```

3 Errory a nedostatky

3.1 Správny output

Bezchybnú funkcionalitu registrujeme v algoritmoch pre šifrovanie, kompozíciu hudby a dekompozíciu. Teda, tie bity, ktoré v skladbe uložíme, z nej aj všetky extrahujeme. Problém nastáva pri dešifrovaní. Pri komponovaní a dekomponovaní hudby sa manipuluje

```
69q 64i 62s 60s 72q 74i 76s 77s 65i 67s 65s 60i. Rs 69i Rs 67s 69i 64s 65s 74q 69i 67s 65s 77q 79i 81s 82s 72i 70s 74s 76i Rs 70s 72i.
11000011001110011011010011111 0110001
10000011100001100111001101101001111101100011001100001110101001100001110001001110101001101001110110011011001111110100100101000101111 1
Vysledny output:
pavolsobotaaaaaaaaaaaaaacmamctfhskd47zjp6rnkn67lafxa5nbi2mhztuadeklk3xvpiygm4dvufdjq7goqamrjnlo6v5daztqowqungd4z2absffvn32xumdg
```

input — prázdne znaky — znaky vytvorené kvôli tomu, aby sa z krátkeho inputu dala vytvoriť celá skladba

```
Process finished with exit code 0
```

s veľkosťami blokov zašifrovaného textu. Ak pri manipulácii nedôjde k zmene veľkosti prvého bloku a veľkosť hudbou zašifrovaných dát presahuje 127 bitov, tak je output správny, pretože prvý blok ostal neporušený a môže byť dešifrovaný pomocou kľúča.

3.2 Nesprávny output

Ak ale veľkosť hudbou zašifrovaných dát je menšia ako 128, tak prvý blok zašifrovaný pomocou kľúča sa poruší a výsledný output je nekorektný.

```
59q 62q 59i 57i 52q 68i. Rs 66i. 68s 59i Rs 61s 56i 61s Rs 52q 56q 52i 50i 45q 52i 56i 61i. 59s 57q 56q 57i. 61s 59q 54i. 59s 61i
00000101111110010100111100011100011
11001100000010111111001010011110001110000110011000000111000100010011111000101110000101100001111010011010010001110000001001
Vysledny output:
gsxkgzpw7vr5r2eo43r5hopzid25r5ybs5k4eqd6dy2jzuxbgukplwhxaglvlqsapypdjhgs4e2rj5oy64azovocib7b4ne42lqtkfhv3d3qdf2vyjah4hruttjocniu

Process finished with exit code 0
```