VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra 456

Webová kytara
Web guitar

2010

Tomáš Buriánek

Prohlášení:

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

Abstrakt

Náplní této práce je tvorba aplikace, spustitelné přímo z internetu použitím webového prohlížeče, která je určená pro vytváření hudebních skladeb pomocí zápisu not do notové osnovy. Tuto osnovu je pak možné dále upravovat změnou pozice not, či editací vlastností jednotlivých not. Tato aplikace pak dále poskytuje zapisování not pomocí druhého způsobu, tedy pomocí zápisu kytarových tabů do tabové osnovy, které jsou typické pro kytarové nástroje. Výsledkem zadávání not či tabů pak může být hudební skladba, kterou lze přehrát, uložit ji do hudebního formátu, či uložit zápis not do souboru pro příští použití. Práce se pak zaobírá teorií základních hudebních pojmů, dále použitými technologiemi potřebných pro funkčnost aplikace a nakonec samtonou implementací aplikace v programovacím jazyce.

Klíčová slova

Java Web Start, Java Sound API, MIDI, Java Swing, nota, tab, zvuková banka.

Abstract

Content of this work is making application executable directly from the internet by web browser and which is for creating music using writing notes to the note stave. This stave is possible to editing by changing position of notes or by changing properties of these notes. This application offers writing notes with another way by writing tabs to tab stave, which is typical for guitar instrumenst. Result of writing notes or tabs to staves maybe music track, that can be played, saved to music format or save note stave to file for next use. Another content of this work deals with teory about basic music terms and using technologies useful for functionality of application and in the end implementation of application in programming language.

Key words

Java Web Start, Java Sound API, MIDI, Java Swing, note, tab, Soundbank.

Obsah:

1. Úvod

Tato aplikace by měla sloužit pro usnadnění tvorby hudebních skladeb, ale také jako jednoduchý nástroj pro učení se základních hudebních znalostí, především zápisu not do notové osnovy, ale i jiných důležitých pojmů.Tento program by měl tedy sloužit ať už pokročilejším hudebníkům, ale i lidem s menšímy znalostmi v oblasti hudby.

Důvod proč jsem tento program vytvořil byl ten, že i když v dnešní době jsou takovéto obdobné aplikace a to i na mnohem profesionálnější úrovni dostupné, nejsou většinou ale volně šiřitelné a i jednoduchý program na zápis not je velmi finančně nákladný. Dalším důvodem je také to, že tyto profesionální programy jsou, pro člověka s menšímy či žádnými znalostmi v oboru hudby, zbytečně složité na pochopení a zorientování se.

Cílem je, aby uživatel mohl spustit program z internetu ve webovém prohlížeči, v tomto programu pak ***zadat do notové osnovy noty*** s příslušnými vlastnostmi, tyto noty dále pak upravovat, tedy ***přemisťovat*** je na jinou pozici***, editovat*** jejich vlastnosti a případně i ***mazat*** z notového zápisu.

Tento program je tedy vhodný pro hudebníka, který hraje na jakýkoliv hudební nástroj, jenž je v tónovém rozsahu tohoto programu.

Protože ale mojí představou bylo to, že tento program bude sloužit hlavně pro hráče na kytarový nástroj, bude v tomto programu další možnost zadávání not do notové osnovy a to pomocí ***tabů*****.** Tento způsob zápisu not je typický hlavně pro hráče na kytaru nebo na baskytaru, či jiný kytarový nástroj. Výhodou zápisu not pomocí tabů je to, že není potřeba tolik znalostí jako pro zápis not do notové osnovy. V tomto zápise by se opět mělo dát taby **vkládat, přemisťovat, editovat** či **mazat.** Mezi těmito oběma zápisy by se měly noty a taby navzájem převádět. Tím tedy uživatel zadávající noty do osnovy bude vědět, jak se tato skladba hraje na kytaru a naopak, pokud bude uživatel zadávat taby do zápisu tabu, tak i laik v hudebním oboru bude vědět jak se jím zadané taby příslušně zadávají do notové osnovy.

Jednou z nejdůležitějších funkcí této aplikace bude pak ,po zadání ať už not do notové osnovy či do tabového zápisu, ***přehrání hudby*** z těchto zápisů. Tím tedy uživatel uslyší, výsledek jeho tvorby. Pro podporu učení hry na kytaru bude také při přehrávání hudby se uživateli postupně ***zvýrazňovat právě přehrávané noty/ taby***, takže bude vědět jaká nota/tab v danou chvíli hraje, což usnadňuje pochopení způsobu hry. Jednou z dalších funkcí programu je vypisování názvů konkrétních tónů, nad každou zadanou notou. Uživatel tak ví i po formální stránce, jak se příslušné tóny jmenují a tím se opět více zdokonaluje v hudbební stránce.

V práci se nejdříve v kapitole 2. Hudební teorie řeší problematika ohledně hudby, tedy důležitých pojmů hudební tématiky jako jsou noty a jejich zápis do notové osnovy a další pojmy, které jsou nutné pro pochopení a práci s programem. V kapitole 3.Zápis not pomocí tabů je popsán další způsob jakým se dají zapisovat noty a to pomocí speciálního zápisu tabů, určený především pro kytarové nástroje. V kapitole 4. Popis technologie MIDI a Java Sound API jsou popsány technologie a prostředky, které jsou využity pro implementaci samotného programu. Kapitola **5. Programátorská dokumentace k vlastní implementaci programu** popisuje již samotný program, tedy implementační jazyk, ve kterém je napsán a dále UML třídní diagram znázorňující celou strukturu programu s jednotlivými třídami a podčástmi programu. Tato struktura je pak popsána v dalších podkapitolkách, kde se nastiňují všechny vytvořené třídy a jejich nejdůležitější úkoly v programu a další důležité vlastnosti. Důraz je kladen hlavně na grafické komponenty sloužící pro zápis not a tabů, dále na přehrávání hudby pomocí vlákna, celkové ovládání programu a různé akce s tím spojené a také využití technologie Java Web Start pro spuštění aplikace z internetu. V kapitole 6.Uživatelská dokumentace se pak popisuje celý program z pohledu uživatele, tedy popis hlávních částí progamu a práce s tímto programem.

2. Hudební teorie

Než dojde k popisu a jádru celého programu je nutné uvést základní teorii ohledně tónů a dalších pojmů, které jsou potřebné pro pochopení některých funkcí programu. Následující materiál je čerpán z literárního zdroje **[1]**.

2.1. Zvuk a tón

Zvuky jsou vše co slyšíme. Vznikají chvěním hmoty, tj. chvěním různých těles, kapalin nebo plynů. Chvějící se těleso rozechvívá okolní vzduch a ten rozechvívá sluchové ústrojí v našem uchu; toto chvění vnímáme jako zvuk.

Zvuky dělíme na *tóny* a *hluky.* Tóny vznikají pravidelným chvěním, hluky nepravidelným chvěním hmoty.

Tóny mají svoji určitou přesnou výšku, kterou můžeme napodobit zpěven nebo hudebnímy nástroji. Tóny zahrané na jednotlivých nástrojích nebo zazpívané tóny se budou sice od sebe lišit barvou ale výška bude stále stejná.

Hluky jsou všechny ostatní zvuky, kterých nemůžeme přesně určit jakou mají výšku. Jsou to například rány, šumy, praskání aj.

V hudbě používáme především tóny. Jsou to tóny hudebních nástrojů a lidského hlasu (zpěvu). V hudbě ale využíváme i některé hluky, například zvuk bubínku, činelů nebo trianglu.

2.2. Vlastnosti tónů

Máme čtyři základní vlastnosti tónů:

Délka- podle toho jak dlouho tóny znějí rozlišujeme jejich různou délku. Tóny mohou být krátké, dlouhé, velmi krátké, stejně dlouhé, kratší delší.

Síla- podle toho jak silně tóny znějí rozlišujeme jejich různou sílu. Jsou to tóny slabé, silné, velmi silné. Apod.

Barva- tóny se vzájemně liší i svojí barvou. Tato barva se dána především nástrojem, který vytváří dané tóny.

Výška- podle výšky tóny rozlišujeme na hluboké a vysoké. Výšku tělesa určujeme přesně počtem kmitů tělesa za vteřinu (kmitočet, frekvence). Nejhlubší tóny mají kmitočet kolem 20 kmitů, vysoké i několik tisíc za vteřinu. Kmitočet tónů ve střední poloze se pohybuje okolo 500 kmitů za vteřinu.

2.3.Tónová soustava

Je to přehledné uspořádání všech tónů podle jejich výšek. Tónová soustava se zaobírá jen jednou vlastností tónů a to pouze jejich výšky. Ostatních tří vlastností, délka, barva, síla, si nevšímá. Základem naší tónové soustavy je sedm tónů: c, d, e, f, g, a, h. Těchto sedm tónů se v tónové soustavě opakuje několikrát v různých výškových polohách. Souhrnně se nazývají ***základní tónová řada*.** Od výchozího tónu **c** k nejbližšímu opakovanému **c** nacházíme osm stupňů: 1. c, 2. d, 3. e, 4. f, 5. g, 6. a, 7. h, 8. c. Vzdálenost mezi těmito dvěma c (nižším a nejblíže vyšším c) se tedy nazývá *oktáva*.

Tónová soustava obsahuje celkem devět oktáv. Každá má své jméno. Subkontra oktáva, kontra, velká, malá, jednočárkovaná ař pětičárkovaná oktáva. Podle toho se i tóny jmenují tak jak patří do jednotlivých oktáv. Např: kontra A, malé d, jednočárkované g apod.

2.4. Tóny zvýšené a snížené

Každý tón naší tónové soustavy můžeme jednou nebo i dvakrát zvýšit nebo snížit. Jednoduché zvýšení tónu označujeme jeho názvem s příponou *-is*, dvojité *-isis*. Takto můžeme obdobně snížit tóny a to příponou -es jednodudché a -eses dvojité.

Máme tedy tóny ***základní*** – c, d, e , f, g, a, h, a dále pak tóny ***zvýšené*** či ***snížené***, které se nazývají

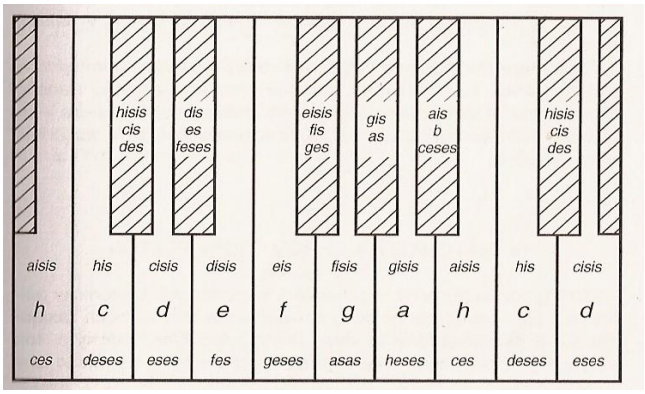
odvozené(alterované) od těchto základních tónů. Snižování a zvyšování tónů se pak nazývá alterace. Tyto odvozené tóny jsou pak dobře vidět v přehledné tabulce *tabulka 1*.

Tóny dvakrát zvýšené	cisis	disis	eisis	fisis	gisis	aisis	hisis
Tóny jednou zvýšené	cisis	disis	eisis	fisis	gisis	aisis	hisis
Tóny základní	c	d	e	f	g	a	h
Tóny jednou snížené	ces	des	es	fes	ges	aes	hes

Tabulka 1: Tabulka základních a odvozených tónů

2.5. Celý tón a půltón, tóny enharmonické

Jak je vidět z tabulky, základních a odvozených tónů je dohromady 35 v rámci jedné oktávy. My však v oktávě rozlišujeme pouze 12 různých tónových výšek. Je tedy jasné, že některé tóny s rozdílnými jmény mají stejnou výšku, neboli tóny stejné výšky se mohou jmenovat různě. Názorně je to vidět na obrázku s klaviaturou klávesového nástroje *obrázek 1*.



Obrázek 1: klaviatura

Klaviatura má bílé a černé klávesy a to v rozmezí jedné oktávy sedm bílých a 5 černých. Bílé klávesy značí základní tóny c, d, e, f, g, a, h, a černé klávesy značí tóny odvozené a střídají se po celé klaviatuře vždy dvě a tři.

Nejmenší vzdálenost ve výšce dvou tónů, která se v naší hudbě používá je ***půtón.*** Nachází se vždy mezi dvěma sousednímu klávesami na klaviatuře a to buď mezi dvěma bílými (e-f) nebo mezi bílou a černou(c – cis). U kytary je to vzdálenost dvou sousedních pražců na hmatníku. Oktáva se tedy skládá z 12ti půltónů.

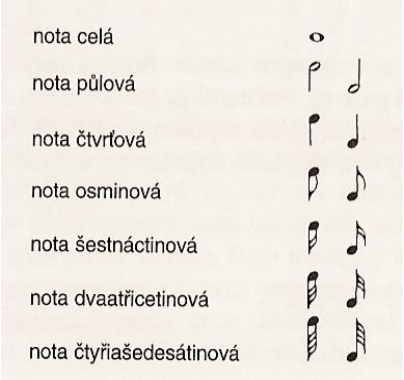
Dva půltóny tvoří celý tón. Je vidět že řada základních tónů z jedné oktávy se skládá ze dvou půtónů (e-f, h-c) a z pěti celých tónů (c-d, d-e, f-g, g-a, a-h). Je tedy patrné že vzdálenosti mezi jednotlivými stupni nejsou stejné. Jednoduchým zvýšením nebo snížením posunujeme výšku tónu o půltón. Například c je vzdáleno od svého jednoduchého zvýšení cis o půltón. Dálepak d je vzdáleno od svého jendoduchého snížení des také o půltón. Takto můžeme odvodit všechny dané snížení a zvýšení tónů a z toho je také patrné že některé odvozené tóny se kryjí s jinými odvozenými nebo základnímu tóny. Z toho vyplývá, že máme několik tónů, které mají stejnou výšku ale jiný název. Tyto tóny se nazývají tóny enharmonické a jsou umístěné na stejné klávese, nebo pražci. Například tóny cis a des jsou k sobě navzájem enharmonické.

2.6. Hodnoty a tvary not, notová osnova

Tóny zapisujeme notami. Zde již přichází na řadu další vlastnost tónů a to ***délka tónu*****.** Tu

označujeme tvarem noty. Výšku tónu zase značíme umístěním noty v notové osnově, tedy v poloze vertikální. Barva tónů se značí použitým nástrojem, ostatní vlastnosti jako síla tónu, rychlosti hry a způsobu přednesu se značí zvláštním označením.

Základní hodnoty a tvary not jsou vyznačeny na obrázku *Obrázek 2*.



Obrázek 2: Základním hodnoty a tvary not

Z obrázku *Obrázek 2* je patrné, že nota celá se rovná dvěma notám půlovým, čtyřem čvrťovým, osmi osminovým atd. Je tedy jasné že nejbližší menší nota má vždy poloviční hodnotu a nejbližší větší nota má vždy hodnotu dvounásobnou než nota předcházející. Podle toho se tedy například nota půlová rovná čtyřem osminovým. Dále je dobré uvést, že v praxi se používají také zkrácené názvy a to: půlka, čtvrtka, osminka, šestnáctinka. Tyto výrazy jsou ale spíše hovorové.

Z grafického vzhledu mají noty tyto tři základní části: ***hlavničku (vyplněnou či nevyplněnou)***, ***nožku***, ***praporec (jednoduchý, dvojitý, trojitý, nebo i čtyřnásobný).*** Nota ***celá*** se skládá pouze z nevyplněné hlavičky. Ostatní noty již mají nožičku, která se píše buďto z levé strany dolů, či z pravé strany nahoru. Nota ***půlová*** má také hlavičku nevyplněnou, ale oproti notě celé má tedy i nožičku. Nota ***čtvrťová*** má oproti notě půlové vyplněnou hlavičku. Nota ***osminová*** má navíc oproti notě čtvrťové praporec. Tento praporec se zapisuje vždy směrem v pravo, ať už je nožička vlevo či vpravo. U dalších not se již jen zvyšuje počet praporců. To vše je dobře vidět na předcházejícím obrázku. Pro inspiraci si můžeme prohlédnout jakékoliv tištěné či elektronické podoby notových zápisů. V rukopisu se pak pro zjednodušení píší jako šikmé čáry.

Pokud v notovém zápisu následují za sebou několik not osminových či menších, můžeme jejich praporce nahradit společným trámcem, který je také jednoduchý, dvojitý, trojitý atd. Pokud je to možné dáváme tyto noty do skupin spojených trámcem, protože to pak zjednodušuje čtení notového zápisu.

Noty pak zapisujeme do ***notové osnovy.*** Tato osnova má pět linek a mezi nimy čtyři mezery. Výšku tónu se pak zvyšuje odspodu nahoru. Dále pak může mít notová osnova další ***pomocné linky*** a to pod notovou osnovou i nad notovou osnovou.

Noty pak můžeme psát na linky i do mezer mezi linkami. Totéž platí i pro pomocné linky.

U not různé výšky spojených trámci se trámec směřuje šikmo. Pokud je to možné, dodržujeme při psaní nožek u not pravidlo, že notám, které se nachází na třetí lince a vyšší, píšeme nožku směrem dolů, kdežto u nižších not směrem nahoru.

Můžeme také spojit trámce i u not různých hodnot tedy osminové se šestnáctinovými aj.

2.7. Jména not a klíče

Jména not v osnově určuje znaménko zvané ***klíč***, který se vždy píše na začátku osnovy na každém řádku. Klíč udává umístění jedné noty a podle ní se řídí i pojmenování not ostatních. Uplatňuje se zde vždy základní tónová řada c d e f g a h, jenže v každém klíči je její postavení jiné. V dnešní době se

užívá čtyř klíčů:

1. G klíč (houslový klíč), který určuje postavení noty g' na druhé lince
2. F klíč (basový klíč), který určuje postavení noty f (malé f) na čtvrté lince
3. C klíč altový, který udává postavní noty c' na třetí lince
4. C klíč tenorový, který určuje postavení noty c' na čtvrté lince

Basový klíč F je především vhodný pro zápis hlubokých tónů, zatímco houslový G klíč je vhodný pro notaci tónů výsokých. Tyto dva klíče jsou také nejvíce používané. Altový a tenorový klíč postihují především střední polohu tónového rozsahu a dost se překrývají s F a G klíčem. Altový klíč se používá především pro violu, protože má nejhlubší strunu c. V tenorovém se pak někdy píše ve vyšší poloze pro violoncello, fagot či pozoun.

Dále je dobré uvést, že nota *c'* je umístěna v houslovém klíči na prvni pomocné lince pod osnovou a v basovém klíči je umístěna na první pomocné lince nad osnovou. Velmi hluboké a velmi vysoké tóny zapisujeme o oktávu výše či níže pomocí zvláštních značek.

2.8. Pomlky

Další důležitou věcí kromě tónů a jiných zvuků mají v hudbě velký význam i přestávky či odmlčení. Tyto odmlčení se označují značkami, tedy *pomlkami*. Můžeme je také nazývat *pauzy* nebo *pomlčky*. Pomlky mají stejné hodnoty jako noty.

Celá pomlka je vodorovná silná čára a *visí* na čtvrté lince, půlová pomlka je pak stejná čárka, ta ale *leží* na třetí lince. Osminová pomlka a další pomlky mají stejný tvar, kdy počet háčků odpovídá počtu praporců. Čtvrťová pomlka pak bývá na způsob obrácené pomlky osminové.

Celá pomlka pak značí pomlku na celý takt a to jakéhokoliv druhu. Proto se také nazývá celotaktová.

2.9. Posuvky

Z předchozí kaptiolce o odvozených tónech vyplývá značení těchto odvozených tónů v notovém zápisu a to pomocí *posuvek*. Jednoduché zvýšení se označuje křížkem **#**, dvojité zvýšení dvojkřížkem **##.** Jednoduché snížení značí bé ***b*** , dvojité snížení pomocí dvojitého bé ***bb.*** Pro nás v našem programu má největší význam **#.**

2.10. Takt

Hudebná skladby členíme na krátké úseky, které se nazývají **takty**. Tyto takty pak vyznačujeme kolmými čarami. Takty se pak dále dělí na doby, což jsou stejně dlouhé časové úseky. Takty mohou mít dvě, tři, nebo čtyři doby. Podle toho jsou odvozeny i základní takty: dvoudobý a třídobý. Ve dvoudobém taktu se střídají dvě doby. Dvoudobý takt se může napsat v notách různých hodnot, tedy takt dvoupůlový, dvoučtvrťový či dvouosminový. V našem případě nám stačí takt **dvoupůlový.** V tomto taktu se tedy nachází v součtu délek not vždy jedna celá hodnota. Tento takt tedy může být složen z jedné noty celé, dvou not půlových, čtyř not čtvrťových a dalších různých kombinací.

2.11 Tempo

Tempo znamená rychlost, jakou se skladba hraje. Tempo udá přesně metronom, přístroj, který udává počet úderů za minutu. Časový úsek mezi dvěma údery metronomu pak představuje půlovou, čtvrťovou nebo osminovou notu. Číselná hodnota tempa pak udává, kolik se provede úderů metronomu za minutu. Jednotlivá tempa pak mají svoje latinské názvy a dělí se na velmi pomalá tempa přes, mírná a rychlá, až po velmi rýchlá tempa. Tyto tempa mají svoje konkrétní hodnoty. Nejnižším tempem je tempo grave s hodnotou 40. Nejvyšším je pak prestissimo s hodnotou 208.

3. Zápis not pomocí tabů

Dalším způsobem jakým je možné také zapsat noty je pomocí speciláního zápisu tabů. Taby jsou v současné době velmi rozšířené a populární. Slouží především pro hráče na kytarové nástroje, ale můžeme se setkat i se zápisem tabu i k jiným nástrojům, například i k bicím.
Taby jsou oproti notám více intuitivní. Nezapisují se do nich konkrétní noty, jako do notového zápisu, ale přímo způsob jakým se hraje na kytaru, aby pak tato vydávala požadované tóny. Z toho vyplývá, že i člověk s menšímy znalostmi hudby, po zorientování se v zápisu tabů, může z tohoto zápisu pohodlně číst a současně i se učit tuto skladbu, kdežto v notovém zápisu je nutné vědět jaký tón se nachází na které lince v zápisu a pak také vědět, kde tento tón může nalézt na kytaře.
Člověk se tak může snadněji naučit a zahrát třeba písničku od své oblíbené kapely, protože mnoho tabů k písničkám, a to hlavně známých kapel, jsou k dispozici volně na internetu.
Zápis do tabu a také čtení z něj, je vcelku jednoduchý. Základem je 6 linek, které představují všech 6 strun na kytaře, které jsou podle daného lazení vyznačeny písmeny, které označují daný tón prázdné struny (např. Lazení EADGBE). Čísla, která jsou pak vyznačena na daných linkách označují, kolikátý pražec na dané struně musí být chycen, aby kytara vydala příslušný tón. Tyto tóny se tedy hrají postupně zleva doprava. Tóny které jsou umístěny nad sebou pak značí souzvuk více tónu v jednu chvíli. Na obrázku *Obrázek 3* je znázorněn tab části skladby "How Many More Times" od skupiny Led Zeppelin ze zdroje **[2].**

```
   Q  E E  Q  Q    Q  Q  Q   Q    Q  E E  Q  Q    Q  Q  Q  Q
e|---------------|--------------|---------------|--------------|
B|---------------|--------------|---------------|--------------|
G|---------------|--------------|---------------|--------------|
D|---------------|--------------|---------------|--------------|
A|----7-5-----5--|----7--5------|----7-5-----5--|----7--5------|
E|-0-------7-----|-0---------7--|-0-------7-----|-0--------7---|
```

Obrázek 3: Zápis části písně "How many more times" od skupiny Led Zeppelin

Protože v taby nemají svůj tvar, tak jako noty, musíme délku jednotlivých tabů znázornit pomocnými značkami, které jsou znázorněny na *Obrázku 4*.

```
W - whole = nota celá
H - half = půlová
Q - quarter = čtvrťová
E - 8th = osminová
S - 16th = šestnáctinová
```

Obrázek 4: Zápis délky tabů

V zápise tabu se pak objevují i další značky, jako jsou skluzy, vibrato, příklepy, odtržení a jiné značky. Nám však pro pochopení stačí tento základní zápis.
Tento způsob zápisu not do tabů tedy lépe podporuje učení se na kytarový nástroj, hlavně pro začátečníky, kteří se těžce orientují v notovém zápise a přesně ani nevědí kde a jakou strunu mají držet aby vydali tón, který chtějí.

4. Popis technologií MIDI a Java Sound API

Před popisem vlastní implementace programu je vhodné také uvést, jakých technologií využívá tento program pro přehrávání a práci s hudbou. Program je implementován v programovacím jazyce Java. Tato Java platforma nabízí aplikační rozhranní Java Sound API, které je určeno právě pro práci se

zvukem v Javě, čehož se tedy v programu využívá. V této API se konkrétně využívá zvuků ve formátu MIDI. Následující text o technologii MIDI je čerpán ze zdroje **[3].** Texto o aplikačním rozhranní je čerpán ze zdroje **[4].**

4.1 MIDI

Zkratka anglického názvu "Musical Instrument Digital Interface". MIDI je mezinárodní hardwarový a softwarový standard, který vzniknul v roce 1983jako elektronický komunikační protokol pro komunikaci mezi hudebnímy zařízenímy jako jsou: syntetizátory, samplery a počítače. Informace, které jsou vyměňovány mezi jednotlivými zařízenímy mohou obsahovat noty, změny programů, hlasitost či další prvky. Základní MIDI protokol poskytuje 16 nezávislých kanálů informací. MIDI tedy nepřenáší audio signál, čí jiná média, jako je tomu u samplovaného zvuku. Přenáší pouze signalizační zprávy, jako jsou intenzita hudebních not pro přehrávání, kontrolní signály pro parametry jako jsou hlasitost, vibrato a snímání, pokyny a hodinový signál pro nastavení tempa, či jiné parametry. MIDI jako elektronický protokol je velmi významný a rozšířený a to hlavně skrze hudební průmysl. MIDI je tak obsaženo ve spoustě počítačových aplikací, které toho využívají k emulování různých zvuků a tím se tedy i obohacuje funkčnost těchto aplikací. MIDI je také využito v Java Sound API, což je pro náš program podstatné.

4.2 Java Sound API

Jak je uvedeno v na stránkách Java Sun ze zdroje [4], Firma Sun popisuje tuto svoji Java Sound API takto:

"Java Sound API je nízkoúrovňová API pro poskytnutí a kontrolování vstupu a výstupu zvukového média. To poskytuje jednoznačnou kontrolu nad schopnostmi obvykle požadovanými pro zvukový vstup a výstup v systému, který podporuje rozšiřitelnost a flexibilitu".

Z toho tedy vyplývá, že základím účelem této API je vypomáhat nám při psaní programů, které budou ve výsledku vytvářet zvukové vlny.

Dále Sun zdůrazňuje že:

"Java Sound poskytuje nejnižší úroveň zvkové podpory na Java platformě. To poskytuje vysoký stupeň kontroly nad zvukově-specifickou funkcionalitu. Neobsahuje tedy sofistikované zvukové editory a grafické nástroje. Spíše poskytuje sadu schopností na kterých můžou některé aplikace stavět. To zdůrazňuje nizko-úrovňovou kontrolu, obvykle požadovanou od koncového uživatele, který těží z vysokůrovňových rozhranních postavenýc na vrcholku Java Sound"

Naším úkolem jakožto programátora je tedy využít Sound API k vytvoření vysokoůrovňového uživatelského rozhranní postaveném na vrcholku Java Sound.

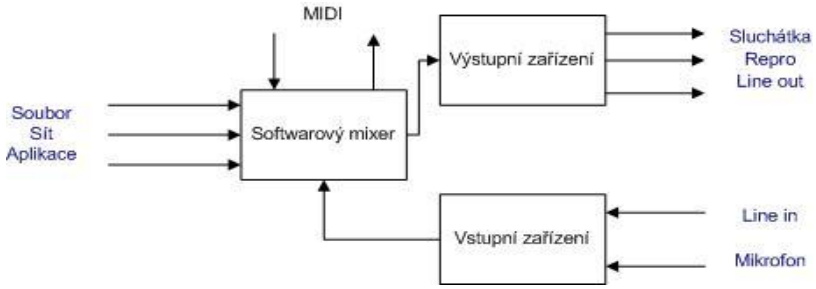
Architektura Java Sound se dělí na dvě části a to: Samplované audio a MIDI audio.

Samplované audio může být chápáno jako série digitálních hodnot, které representují amplitudu, či intensitu zvukových vln. Tento typ audio dat a práce s nimi je obasaženo v Java balíčku javax.sound.sampled, který poskytuje rozhranní pro zaznamenávání, mixování a přehrávání digitálního (samplovaného) zvuku.

MIDI audio může být chápáno jako zvuk "vytvořený podle šablony". Tento typ zvukových dat a práce s nimy je obsažena v Java balíčku javax.sound.midi.

Je zde ještě mnoho pomocných a implementačních tříd, které se zabývají zvukem a to v balíčku com.sun.media. Ostatní zvukové třídy, většinou odkazující na starší implementace, jsou umístěny v balíčku sun.audio.

Obrázek *Obrázek 5* vystihuje některé z důležitých prvků Java Sound API.

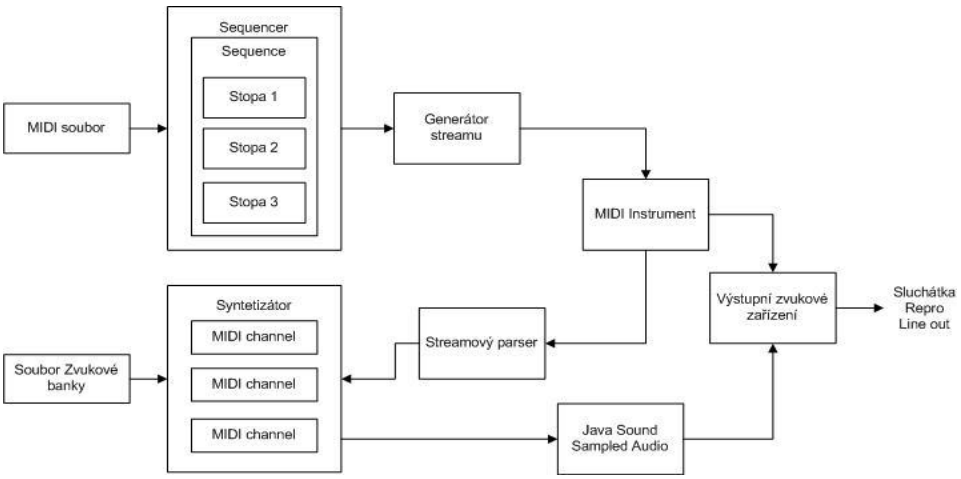


Obrázek 5: Blokový popis vnitřní struktury prvků Java Sound API

Klíčovým prvkem je Softwarový mixer. Jak je vidět, tak do tohoto Softwarového mixeru vede několik různých vstupů, ať už ze MIDI zařízení, zvukového souboru, sítě, či aplikace, ale i ze vstupního zařízení, do kterého může vést zvuk ze vstupu Line-in či mikrfonu. Tento mixer pak smísí všechny vstupy a posílá je na výstupní zařízení, které může být napojené na repro, sluchátka, nebo jiný line-out výstup.

4.2.1 MIDI v Javě

Třída javax.sound.midi.MidiSystem poskytuje vstupní bod do systémových MIDI zdrojů. Použitím statických metod MidiSystem můžeme získat objekty Sequencer nebo Synthesizer, můžeme také získat informace o systémových zařízeních, či zjistit, jaké typy zařízení a souborů jsou podporovány.Na obrázku *Obrázek 6* je zobrazeno, jak jsou spojeny jednotlivé části podsystému MIDI.



Obrázek 6: Blokový popis vnitří struktury prvků java MIDI System

MIDI soubor je kolekce stop, z nichž každá stopa má sadu not. Nota má svoji časovou známku, hodnotu noty, příkazy potřebné pro zapnutí a vypnutí noty a další potřebná data pro práci s touto notou. Tento MIDI soubor je načten jako Sequence do Sequenceru pomocí metody ***setSequence***. Načtené stopy z MIDI souboru mohou být pak dále upravovány pomocí programovacího rozhranní Java Sound. Z tohoto Sequenceru mohou být pak MIDI stopy seřazeny či spojeny dohromady v časovém pořadí. Tento vzniklý stream – proud MIDI stop pak může být zaslán skrze Generátor

Streamu do MIDI Instrumentu. Tento MIDI isntrument může syntetizovat MIDI stream do analogového zvuku, který již konečně může být slyšet ze sluchátek či, jiného audio výstupu. MIDI Sequence může být také poslána do Syntetizátoru. V tomto Syntetizátoru jsou MIDI stopy načteny do kanálů. Tyto kanály jsou spojeny se zvuky instrumentů, které jsou načteny ze zvukové banky. Stopy dohromady s instrumenty pak mohou být syntetizovány do vzorkovaného zvuku – Sampled audio. Java Sound pak vezme toto Sampled audio a pošle jej do zvukové karty počítače, která přehraje tento zvuk opět skrze sluchátka, repro či jiný audio výstup.

Soubor Zvukové banky je načten do Syntetizátoru pomocí metody ***loadAllInstruments***. Kvalita MIDI přehrávaného zvuku ze sluchátek pak tedy závisí na instrumentu zvukové banky. Základní zvuková banka obsahuje vzorky instrumentů s nízkou kvalitou zvuku. Pokud bychom chtěli zvukové banky s vyšší kvalitou, tak tyto jsou dostupné na stránkách Sun Java Sound.

Výhodou programovacího rozhranní Java Sound MIDI je to, že MIDI podsystém může být různě řetězen a měněn. Toto je pro nás tedy velmi výhodné, protože můžeme tvořit na tomto MIDI systému programy, které různě generují a využívají hudbu.

Pro účely našeho programu využijeme již předem daných Zvukových bank, ze kterých se pak pomocí Syntetizátoru vytvoří požadovaný zvuk.

4.2.2 Syntetizátor

Syntetizátor je v javě reprezentován pomocí třídy javax.sound.midi.Synthesizer. Objekt této třídy pak musíme získat z MIDI systému pomocí metody MidiSystem.getSynthesizer(), ale pouze pokud je MIDI dostupné. Abychom mohli pomocí syntetizátoru vytvářet zvuk, potřebujeme nejdříve Zvukovou banku. Tato banka se už může v Syntetizátoru standartně nacházet. Avšak v některých Java produktech se tato zvuková banka nenachází a následkem toho může být nepřehrání požadovaného zvuku. Tento problém se dá vyřešit, pokud nechceme tedy přeinstalovávat náš Java produkt, dvěma způsoby. Pro oba způsoby však budeme potřebovat zvukovou banku. Na stránkách Java Sun na http://java.sun.com/products/java-media/sound/soundbanks.html, jsou k dispozici tři zvukové banky, s různou kvalitou zvuku, zadarmo ke stažení. Tyto banky mají pak příponu .**gm**. Pokud bychom tedy chceme jinou zvukovou banku s dalšímy jinými nástroji obsaženými uvnitř, musíme hledat na internetu soubory s touto příponou. Jiné zvukové banky Java nepodporuje.

Prvním způsobem, jak vyřešit problém zvukové banky neobsažené v Java produktu, je ta, že staženou zvukovou banku nahrajeme do tohoto produktu. To se provede tak, že si otevřeme instalační složku produktu, vybereme produkt Javy, který je v našem počítači využíván pro spouštění aplikací a do složky **lib** vytvoříme složku **audio**, pokud se tu tedy tato složka nenachází. Budeme mít tedy otevřenou složku např.: **C:\ProgramFiles\Java\jre\lib\audio .** Do této složky pak zkopírujeme staženou zvukovou banku a pojmenujeme ji **soundbank.gm**. Po tomto nakopírování by pak měly MIDI zvuky z této zvukové banky v našich aplikacích fungovat.

Druhým způsobem je načíst zvukovou banku přímo do naší aplikace, čímž načteme do syntetizátoru místo standartní zvukové banky z Java produktu, banku ze souboru. To můžeme v naší aplikaci provést pomocí příkazu loadAllInstruments, ve kterém se definuje cesta k této zvukové bance.

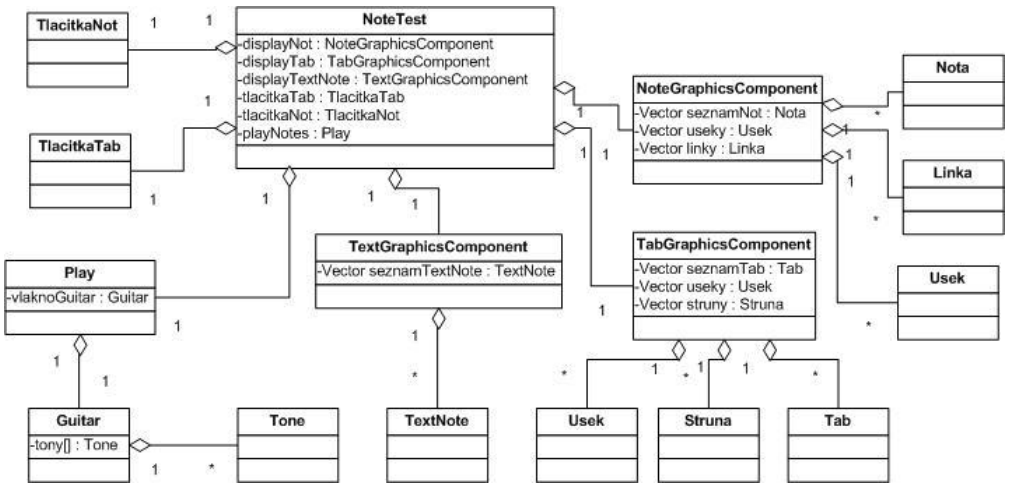
Nyní když je zvuková banka dostupná, můžeme otevřít syntetizátor a získat z něj dostupné MIDI kanály. MIDI obsahuje jednu kolekci všech 16ti kanálů, kterí jsou definovány specifikacemi MIDI 1.0. Z těchto kanálů se pak vybere pouze jeden. Tento vybraný kanál pak zpracovává kanálové zprávy, které jsou také definovány MIDI specifikacemi. Syntetizátor tedy pak vytváří zvuk, jakmile kanál obdrží zprávu **noteOn**. V této zprávě je obsaženo číslo noty, která se má přehrát a síla této noty. Tato nota pak zní, dokud kanál neobdrží zprávu **noteOff**, opět s parametry této noty. Tím tedy můžeme určovat, jak dlouho bude daná nota znít, a to tím, že mezi zprávami noteOn a noteOff vytvoříme časové zpoždění, například uspáním vlákna na požadovanou dobu. Všechny tyto zvuky vytvořené

Syntetizátorem se tedy pak pošlou na výstup zvukové karty a tím přehrání hudby z reproduktorů, či jiného zvukového zařízení.

5. Programátorská dokumentace k vlastní implementaci programu

Tento program je naimplementovám v programovacím jazyce Java. Pro spuštění a práci s aplikací je tedy potřebné mít naistalovanou java platformu od firmy Sun Microsystems. Tuto platforma je dostupná zadarmo ke stažení na stránkách Javy ze zdroje **[5]**Při implementaci progamu je využito mnoha vestavěných tříd, které usnadňují tvorbu celkové funkčnosti programu.

Základem celého programu je třída **NoteTest**, která obaluje všechny objekty z ostatních tříd. V této třídě se tedy sestavují všechny grafické komponenty do jednoho celku a výsledkem je jedno grafické rozhranní, pomocí kterého uživatel ovládá celý program. Dále se v této třídě nachází další objekty tříd, které jsou potřebné pro chod celého programu a pro zobrazování či jiného distribuování výsledků uživateli. Provázanost jednotlivých tříd je vidět na obrázku *Obrázek 7* UML třídního diagramu



Obrázek 7: UML třídní diagram

Jsou zde vidět vazby mezi třídami, tedy to, které objekty tříd jsou obsaženy v jiných objektech tříd, jejich násobnost a důležité atributy.

5.1. Třída NoteTest

Jak již bylo uvedeno, tak tato třída je základem celého programu. Je rozšířena jako **JApplet**, což je třída z balíčku **javax.swing**. Applet je softwarová komponenta, která běží v rámci jiného programu, což může být webový prohlížeč, či panel grafického prostředí. V našem případě tento JApplet vkládáme jako obsah okna JFrame, což se provádí v hlavní spouštěcí metodě **main**. Tím tedy po spuštění programu se nám zobrazí vytvořený **JFrame** i s jeho obsaženým JAppletem **NoteTest** a tím se tedy zobrazí i obsah tohoto appletu.

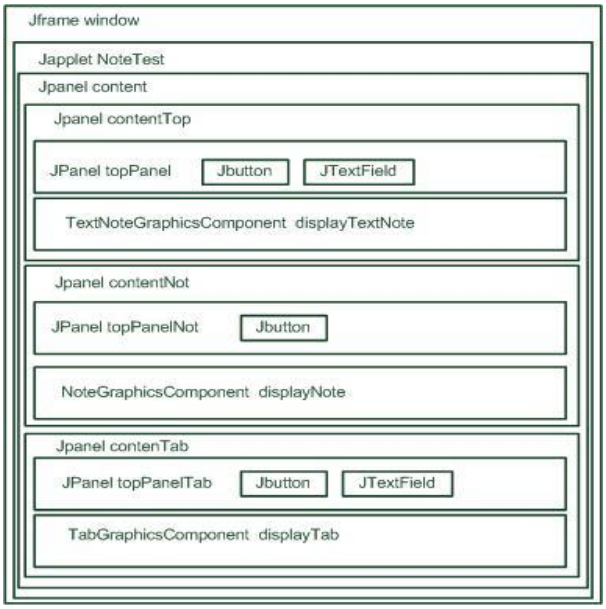
5.1.1. Obsah třídy

Tomuto JAppletu je předáván určitý obsah, který se bude ve výsledku zobrazovat. Aby byl tento obsah správně zobrazen, je zde vytvořena menší hierarchická strunktura panelů. Je zde tedy vytvořen

hlavní ***JPanel* content.** Tento panel bude obsahovat další panely a to **contentTop**, **contentTab** a **contentNot.** Tyto panely jsou pak do hlavního panelu přidány a rozmístěny pomocí ***BorderLayout*** na příslušné pozice. To je vidět na v ukázce příkazů ze zdrojového kódu. Panel **contentTop** pak obsahuje dvě podčásti. První je panel **topPanel**, který obsahuje komponenty jako tlačítka **JButton** a další ovládací prvky jako jsou třeba **JTextField**. Tyto komponenty slouží především pro práci s celým programem, tedy přehrávání notového zápisu a jiných akcí s celým programem. Druhou podčástí contentTop je přímo grafická komponenta **displayTextNote** ze třídy ***TextNoteGraphicsComponent*.** Tato komponenta zobrazuje textový zápis zadaných not. Tyto dvě podčásti jsou tedy přidány do *contentTop* opět pomocí *BorderLayout*.

Panel **contentNot** obsahuje také dvě podčásti. První je panel **topPanelNot**, který obsahuje tlačítka, která jsou určena pro práci přímo s grafickou komponentou. Druhá část je tedy grafická komponenta **displayNot** ze třídy ***NoteGraphicsComponent***. Tato komponenta zobrazuje notový zápis, se kterým pak dále pracujeme.

Posledním panelem je panel **contentTab**, který obsahuje opět dvě podčásti a to panel **topPanelTab**, obsahující pořebná tlačítka pro práci s příslušnou grafickou komponentou, a přímo grafickou komponentu **displayTab** ze třídy ***TabGraphicsComponent***. Opět seřazené pomocí *BorderLayout*. Tímto tedy máme všechny panely a grafické komponenty hierarchicky rozmístěné v jednom jediném panelu content, který nastavíme jako obsahový našemu appletu pomocí příkazu **this.setContentPane(content);** Výsledná hierarchie všech komponent je dobře vidět na znázorňujícím obrázku *Obrázek 8.*



Obrázek 8: Blokové shéma hierarchie panelů

Tlačítkům a ostatním komponentám z top panelů jednotlivých částí jsou pak v **konstruktoru** appletu NoteTest předány příslušné **Action Listenery**, tedy objekty ze tříd, které jsou určeny pro práci s těmito komponentami. Kompentám z topPanel předáváme objekt ze třídy **Play**. Komponentám z topPanelNot předáváme objekt ze třídy **TlacitkaNot**. Komponentám z toPanelTab předáváme objekt ze třídy **TlacitkaTab.**

Grafickým komponentám **displayNot** a **displayTab** pak předáme **Mouse Listener** a **Mouse Motion Listener**, a to pro zaznamenávání akcí spojených s myší – kliknutí, tažení a pohyb myši.

5.1.2 Práce s tlačítky a dalšímy ovládacími prvky

Pro tyto účely slouží tři třídy, které jsou obsaženy v této NoteTest hlavní třídě. Jsou to třídy **TlacitkaNot**, **TlacitkaTab** a **Play**.

Třída **TlacitkaNot** slouží pro práci s ovládacímy prvky nacházející se v panelu **topPanelNot** a jsou určené pro komponentu NoteGraphicsComponent. Tato třída implementuje rozhranní ActionListener. Proto se v této třídě nachází metoda **public void actionPerformed(ActionEvent e)**, která se spouští při každé akci, která se provede s tlačítky či jinými prvky, kterým byl nastaven Action Listener jako objekt této třídy. V této metodě se pak testuje, se kterým prvkem se pracuje a podle toho se provádějí další příslušné akce, které jsou nutné pro ovládání chodu programu. V této třídě tedy jde o nastavování parametrů teprve vkládaných not a také k editaci již vložených not. Také se zde rozhoduje jestli se budou vkládat, či přidávat nové noty.

Třída **TlacitkaTab** slouží opět pro práci s ovládacímy prvky, které se ale nacházejí v panelu **topPanelTab** a jsou určené pro komponentu TabGraphicsComponent. Tato třída opět implementuje rozhranní ActionListener a tedy obsahuje metodu **actionPerformed**. V této metodě se tedy pomocí ovládacích prvků nastavují parametry nově vkládaných tabů a také se provádí editace již vložených prvků.

Třída **Play** slouží také pro práci s ovládacímy prvky. Ty se nachází v panelu **topPanel**. Třídě je implementováno opět rozhranní ActionListener. Nachází se zde tlačítka pro vytvoření, spouštění a zastavování vlákna **Guitar**, které slouží pro přehrávání notového zápisu a k dalším úkonům spojeným s přehráváním, jako je stopnutí tohoto přehrávání, ale také i záznam hudby. Současně s tímto vláknem se spouští další dvě vlákna **not** a **tab** vytvořená ze tříd NoteGraphicsComponent a TabGraphicsComponent. Ty pak slouží pro zvýrazňování přehrávaných not či tabů. Ve třídě Play se pak ještě nachází další tři tlačítka, která slouží pro uložení obrázků jednotlivých zápisů, pro uložení notového zápisu do souboru a tlačítko pro načtení not z uloženého zápisu.

5.2. Třída NoteGraphicsComponent

Jednou z nejdůležitějších částí tohoto programu je třída **NoteGraphicsComponent**, jejíž objekt **displayNot** se nachází v hlavní třídě NoteTest, kde je tento objekt přidán jako jedna z částí obsahu tohoto appletu.

Tato grafická komponenta slouží pro grafické znázornění notového zápisu, ke vkládání nových not do tohoto zápisu a pro další akce s tím spojené. Data vložená uživatelem do objektu této třídy jsou pak dále zpracována ve třídě ***NoteTest*** a využita pro další účely, jako je převedení notového zápisu na tab a také na přehrání výsledného notového zápisu.

Nejdůležitějšími objekty v této komponentě jsou tři ***Vector*** seznamy. První seznam je ***Vector*** **seznamNot**, který slouží k uložení všech not ze třídy ***Nota***. Další je ***Vector*** **linky**, sloužící pro uložení všech linek ze třídy ***Linka***. Posledním je ***Vector*** **useky**, který slouží k uložení všech úseků ze třídy **Usek**. Dalšímy proměnnými jsou proměnné pro nastavení rozměrů grafického rozhranní a dále různé pomocné proměnné pro práci s komponentou. Všechny tyto proměnné jsou uloženy jako třídní. Součástí této třídy jsou tedy další tři třídy, které reprezentují tyto seznamy objektů.

5.2.1 Třída Line

Tato třída slouží k reprezentaci jedné linky notové osnovy. Nejdůležitějším parametrem je parametr **linka**, který určuje pořadové číslo linky. Linky jsou pak podle tohoto parametru seřazeny od spodu nahoru. Z hodnoty parametru linka se pak odvodí parametr Y, který společně s parametrem X slouží jako souřadnice pro vykreslení této linky v kreslící ploše grafické komponenty. To se provádí pomocí metody **public void vykresli(Graphics g)**.

5.2.2 Třída Usek

Slouží pro reprezentaci jednoho časového úseku v notovém zápisu. V této třídě je základním parametrem **usek**, což pořadové číslo úseku. Tyto úseky jsou pak seřazeny podle tohoto čísla zleva doprava. Podle čísla úseku se pak odvodí paramter X, který společně s parametrem Y určuje souřadnice tohoto konkrétního úseku v kreslící ploše komponenty. Vykreslení tohoto obdélníku se provádí opět pomocí metody **public void vykresli(Graphics g)**. Tento úsek se vykreslí jako obdélník, který však je v barvě pozadí a tudíž není vidět. Úsek je viditelný jen v tom případě, že do něj najedem myší.

5.2.3 Třída Nota.

Tato třída je jedna z nejdůležitějších, protože představuje vždy jednu konkrétní vykreslovanou notu. Ta má svoji pozici ve vykreslované ploše a ta je odvozena od toho na jaké lince a ve kterém úseku se nachází. Podle toho na jaké lince se nota nachází má nota svoji výšku tónu. Úsek nám pak značí v jakém pořadí se jednotlivé noty budou přehrávat. Nota má také svůj tvar a to podle toho jak dlouho zní v jednom taktu, tedy jestli je to nota celá, půlová, čtvrťová.

Třída obsahuje několik základních proměnných, které určují vlastnosti Noty. Jsou to tedy **linka**, **usek**, na kterých se nota nachází a podle nich se určí souřadnice X a Y. Dalšímy parametry jsou **ton** a **okt**, které určují tón a oktávu v které se tento tón nachází. Tyto hodnoty se získají z čísla linky na které se nachází. Parametr **ND** určuje délku noty v taktu, jsou to hodnoty 1, 0.5, 0.25 ..z toho tedy vyplývá název a tvar noty (nota celá, půlová...). Parametr **pulton** určuje jestli bude nota zvýšena o půltón, či nikoliv a parametr **zvys** určuje, jestli bude tón zvýšen o oktávu.

Pro vykreslení noty je tedy potřeba znát její souřadnice X a Y, které jsou určené z úseku a linky a dále pak tvar noty, který je dán parametrem ND. Nota se tedy vykreslí opět metodou **public void vykresli(Graphics g)**, na příslušných souřadnicích a s příslušným tvarem podle hodnoty ND.

5.2.4 Vytvoření všech komponent obsažených v hlavní komponentě

Všechny linky ze třídy **Linka** a úseky ze třídy **Usek** jsou vytvořeny již v konstruktoru hlavní grafické komponenty, ve které se nacházejí, takže uživatel již nemůže tyto komponenty měnit. Avšak noty už musí uživatel zadat sám. To může uživatel provést třemi způsoby - Pomocí myši je vložit kliknutím do určitého úseku na určité lince notové osnovy, načtením not ze souboru, zadáním tabů do tabové osnovy a následné převedení tabů do not.

5.2.4.1. Vložení not ze třídy Nota pomocí myši

Pro práci s myší jsou naimplementovány do této komponenty metody ze třídy ***MouseListener*** a ***MouseMotionListener***. Tím se nám vytvoří několik metod, které tyto třídy obsahují. My však využijeme jen několik základních a to **public void mousePressed(MouseEvent e), public void mouseMoved(MouseEvent e), public void mouseDragged(MouseEvent e), public void mouseReleased(MouseEvent e).**

Metoda **public void mousePressed(MouseEvent e)** se volá vždy, když uživatel klikne tlačítkem myši do kreslící plochy. Pokud klikne do prostoru vymezeným úseky a linkami a je přepnuto na tlačítko s popiskem "**Vkládání**", volá se metoda **public void pridejNotu(e)** a pokud je to možné přidá se do seznamu not nová nota. Pokud se však klikne na již přidanou notu, tak tato not se nalezne v **seznamNot** označí se pomocí metody **nota.oznac(true)**. Pokud však je přepnuté na tlačítko s popiskem "**Odstraňování**", tak tímto již přidané noty se mažou z notového zápisu a to metodou **seznamNot.remove(aktualni)**, kde **aktualni** značí index noty na kterou bylo kliknuto.V této metodě se ještě také volá další metoda **fixOrder()**, která slouží pro správné seřazení not v notovém zápisu, protože se může stát, že novou notu přidáme v notovém zápise mezi další dvě, ale do **seznamNot** se přidá na konec. To by mohlo způsobit špatné pořadí přehrávaných not. Tímto se tedy noty seřadí

pomocí jednoduchého třídícího algoritmu **Bubble Sort**. Poslední volanou metodou je zde metoda **takt()**, která slouží k vyznačení taktu v notovém zápise pomocí dvou svislých čar a to vždy po sekvenci not, kdy součet jejich hodnot **ND**, tedy délky těchto not v taktu je rovna hodnotě 1. Pokud tento součet však je větší jako 1, jsou noty do notové osnovy zadány špatně a tím se tento takt zvýrazní červeně, s tím i poslední nota přidaná do tohoto taktu.

Metoda **public void mouseMoved(MouseEvent e)** se volá pokaždé, když se pohne myší v notovém zápise. V této metodě se tedy testuje jestli se kurzor myši nenachází zrovna nad nějakým úsekem, linkou či notou a volají se metody **testujLine(e), testujNote(e)**, **testujUsek(e)** a podle toho se tyto objekty také vykreslují, zvýrazňují se tedy například noty a linky modrou barvou, pokud se kursor nachází přímo nad konkrétní notou, či linkou.Tyto metody se pak dále využívají i na jiné účely. Dále se zde nastaví index sdíleného objektu **uvnitr** na hodnotu 1. Toho se využívá ve třídě **NoteTest** k zjištění, ve které grafické komponentě se nachází kursor myši.

Metoda **public void mouseDragged(MouseEvent e)** se spustí pokaždé, pokud se klikne a táhne myší ve vymezené kreslící ploše. Tato metoda se využívá na přemisťování not v notovém zápisu. Volá se zde metodu **pohniNota(e)**, čímž se přemístí vybraná nota na nové místo v notovém zápise, pokud se tedy na tomto místě již nenachází jiná not. Zde se také volají další dvě metody: fixOrder() takt().

5.2.4.2. Vložení not ze souboru

Druhým způspobem, jak může uživatel vložit noty do notového zápisu, je načíst je ze souboru. K tomu slouží metoda **public void ctiZeSouboru(String path)**, která se volá po stisknutí příslušného tlačítka ze třídy **Play**, kdy se vybere cesta k souboru a při volání metody se tato cesta předá. V metodě ctiZeSouboru se pak smaže celý obsah seznamu not a z **BufferedReader** se přečtou postupně všechny řádky vybraného souboru. Tyto řádky se mezitím rozparsují na jednotlivé části pomocí **StringTokenizer**. Tyto části pak představují jednotlivé parametry not. Každý řádek představuje jednu notu. Po získání těchto parametrů z jednoho řádku se pak vytvoří vždy nová nota s těmito parametry. Tato nota se pak přidá do seznamu not s novými načtenými hodnotami. Toto se provádí, dokud BufferedReader nedojde na konec souboru. Tím tedy máme načtený celý nový notový zápis ze souboru.

5.2.4.3. Vložení tabů do tabového zápisu a následný převod na noty

Posledním způsobem, kterým se dají zadat noty do notového zápisu je vložením tabů do komponenty displayTab ze třídy TabGraphicsComponent. Funkce této komponenty je popsána dále, avšak práce s ní je obdobná jako s NoteGraphicsComponent. Zadáním vždy nového tabu do zápisu, se volá automaticky metoda pro převod TabToNote(), ve které se pomocí specifických operací převedou taby na noty a tím se i automaticky mění notový zápis, který zápisu v tabu musí odpovídat.

5.2.5 Vykreslení všech komponent

Nyní, když se v komponentě NoteGraphicsComponent nachází všechny komponenty potřebné pro práci, je možně je vykreslit do vykreslovací plochy této komponenty. To se provádí pomocí *Override* metody ze třídy JComponent – **public void paintComponent(Graphics g)**. Vykreslení se pak provádí ve třech cyklech, kdy se nejdříve postupně vykreslí všechny **úseky,** dále všechny **linky** a nakonec všechny **noty** z příslušných vektorových seznamů. Tato metoda se volá hned po inicializaci objektu této třídy a pak pokaždé, když se zavolá metoda **repaint()**, ve které se nejdříve celá plocha smaže a nadále pak zavolá **paintComponent**, aby se celá plocha vykreslila znovu. Metoda repaint se volá vždy když dochází ke změně obsahu komponenty, což nastáví při akcíh uživatele, jako vkládání nových not, jejich přemisťování, editace, mazání, označování a i jiných akcí.

5.3 Třída TabGraphicsComponent

Další velmi důležitou částí programu je objekt **displayTab** ze třídy **TabGraphicsComponent**, který se také nachází v hlavní třídě NoteTest a je nastaven jako jedna ze součástí grafického rozhranní

Tato komponenta slouží ke grafickému znázornění tabů přidávaných do tabového zápisu a k další práci s tímto zápisem. Data obsažená v objektu této třídy, která vložil uživatel, či samotný program při převodu mezi jednotlivými zápisy, jsou pak dále zpracována v nadřazeném objektu třídy **NoteTest,** kde se provádí další akce s tím spojené, jako je již zmíněný převod mezi oběma zápisy a nebo přehrávání výsledného zápisu.

Tato třída obsahuje jako nejdůležitější třídní proměnné tři ***Vector*** seznamy objektů. Prvním seznamem je **seznamTab**, který obsahuje všechny taby ze třídy ***Tab*** vložené do zápisu. Druhým seznamem je seznam **struny**, který obsahuje všech 6 strun ze třídy ***Struna***, které se nachází v grafickém prostředí této komponenty. Posledním je seznam **useky**, ve kterém jsou všechny úseky ze třídy ***Usek***. Dalšímy proměnnými jsou proměnné pro nastavení rozměrů grafického rozhranní a další různé pomocné proměnné pro práci s komponentou. Všechny tyto proměnné jsou uloženy jako třídní. Součástí této třídy jsou tedy další tři třídy, které reprezentují tyto seznamy objektů.

5.3.1. Třída Struna

Tato třída reprezentuje jednu konkrétní strunu. Základním parametrem, kterým se jednotlivé struny odlišují je třídní proměnná **struna**, která představuje číslo struny od 1 do 6ti. Podle čísla struny pak odvodíme hodnotu souřadnice Y. Za hodnotu proměnné X se pak dosadí hodnota odsazení strun od levého okraje. Konkrétní strunu pak vykreslíme do plochy pomocí metody **public void vykresli(Graphics g)**. To se provede vykreslením čáry se souřadnicemi počátečního a koncového bodu, které určíme z hodnot X, Y a rozměru grafického prostředí.

5.3.2 Třída Usek

Slouží k reprezentaci jednoho úseku v zápisu tabu. V této třídě je základním parametrem, kterým rozlišujeme jednotlivé úseky od sebe, třídní proměnná **usek**, jejíž hodnota nabývající čísla od 1 do N, určuje pořadí úseku v komponentě, seřazené zleva doprava. Podle čísla úseku se pak odvodí hodnota souřadnice X. Hodnota Y pak znamená odsazení úseku od horního okraje vykreslované plochy. Tento úsek se pak vykresluje opět pomocí metody **public void vykresli(Graphics g)**, jako obdélník. Tento obdélník je však viditelný pouze pokud do tohoto konkrétního úseku najedem myší, tím se zvýrazní šedou barvou. Jinak je úsek vykreslován bílou barvou pozadí.

5.3.3 Třída Tab

Tato třída je nejdůležitější, protože jeden objekt z této třídy představuje jeden konkrétní tab. Tento tab má parametr **prazec**, což je hodnota, která určuje, na kterém pražci kytary by se musela držet struna, aby kytara vydala požadovaný tón. Tab má dále parametry **struna** a **usek**, tedy číslo struny a úseku, ve kterých se tab nachází. Z těchto hodnot se pak určí pozice tabu X a Y, které jsou potřebné pro vykreslení tabu do zápisu. Důležitým parametrem je ještě **ND**, jehož hodnota nabývající 1, 0.5, 0.25.. určuje délku trvání tabu v jednom taktu.

Pro vykreslení tohoto tabu se volá metoda **public void vykresli(Graphics g)**. V této metodě se pak vykreslí daný tab jako čtverec na dané souřadnici. Do tohoto čtverce se pak vykreslí číslo pražce, na kterém se nachází. Protože tab nemá svů tvar tak jako nota, vypisuje se jeho délka pomocnými písmeny – W, H, Q, E, S.

5.3.4. Vytvoření všech potřebných komponent

Všechny struny a úseky jsou vytvořeny již v konstruktoru grafické komponenty, takže uživatel již nemůže tyto komponenty měnit. Avšak taby už musí uživatel tak jako noty zadat sám. To může provést

opět třemi způsoby: Pomocí myši je vložit kliknutím do určitého úseku na určité struně zápisu tabu, načtením not ze souboru a následným převodem na taby, Zadáním not do notové osnovy a následným převodem na taby.

5.3.4.1. Vložení tabu ze třídy Tab pomocí myši

Pro práci s myší jsou naimplementovány do této komponenty metody ze třídy ***MouseListener*** a ***MouseMotionListener***. Tím se nám vytvoří několik metod, které tyto třídy obsahují. My však využijeme jen několik základních a to **public void mousePressed(MouseEvent e), public void mouseMoved(MouseEvent e), public void mouseDragged(MouseEvent e), public void mouseReleased(MouseEvent e) .**

Metoda **public void mousePressed(MouseEvent e)** se volá vždy, když uživatel klikne tlačítkem myši do kreslící plochy. Pokud klikne do prostoru vymezeným úseky a strunami a je přepnuto na tlačítko s popiskem "**Vkládání**", volá se metoda **public void pridejTab(e)** a pokud je to možné přidá se do seznamu tabů nový tab na souřadnicích kurzoru myši. Pokud se však klikne na již přidaný tab, tak tento tab se nalezne v **seznamTab** označí se pomocí metody **tab.oznac(true)**. Pokud však je přepnuté tlačítko na funkci "**Odstraňování**", tak tímto již přidaný vybraný tab se odstraní ze zápisu tabu a to metodou **seznamTab.remove(aktualni)**, kde **aktualni** značí index tabu na který bylo kliknuto.V této metodě se ještě také volá další metody **fixOrder()** a **takt()**, které funguje obdobně jako ve třídě **NoteGraphicsComponent.**

Metoda **public void mouseMoved(MouseEvent e)** se volá pokaždé, když se pohne myší v zápise. V této metodě se tedy testuje, jestli se kurzor myši nenachází zrovna nad nějakým úsekem, strunou či tabem a volají se metody **testStruna(e), testTab(e)**, **testUsek(e)**, které slouží pro testování jestli, se kurzor myši přávě nenachází, nad některou ze strun, tabů nebo úseků.

Metoda **public void mouseDragged(MouseEvent e)** se spustí pokaždé, pokud se klikne a táhneme myší v kreslící ploše. Tato metoda se využívá na přemisťování tabů v zápise. Volá se v ní metoda **pohniTab(e)**, čímž se přemístí vybraný tab na nové místo v zápise, pokud se tedy na tomto místě již nenachází jiný tab. Zde se také volají další dvě metody: fixOrder() takt().

5.3.4.2.Vložení not ze souboru a převod do tabu

Druhým způsobem, jak se mohou vložit noty do notového zápisu, je načíst je ze souboru. K tomu slouží opět metoda **public void ctiZeSouboru(String path)** ze třídy NoteGraphicsComponent. Tím se tedy načtou nové noty ze souboru a automaticky se provede převod na taby v metodě **NoteToTab**, která se volá v hlavní třídě **NoteTest**. V ní se pak pomocí specifických operací provádí převod a výsledkem jsou převedené taby v zápise tabu, které odpovídají notám.

5.3.4.3. Vložení not do notového zápisu a následný převod na taby

Vkládání nových not do notového zápisu již bylo popsáno ve třídě NoteGraphicsComponent. Pokud se tedy vkládají noty do zápisu, automaticky se hned převádějí na taby pomocí metody **NoteToTab** a vkládají se zároveň i do tabového zápisu.

5.3.5 Vykreslení všech komponent

Nyní když jsou již vytvořené všechny potřebné komponenty, které se nacházejí v komponentě TabGraphicsComponent, je možné je vykreslit do vykreslovací plochy této komponenty. To se provádí pomocí *Override* metody **public void paintComponent(Graphics g)** a to ve třech cyklech, kdy se nejdříve postupně vykreslí všechny **úseky,** dále všechny **struny** a nakonec všechny **taby** z příslušných vektorových seznamů. Tato metoda se volá hned po inicializaci objektu této třídy a pak pokaždé, když se zavolá metoda **repaint()**, ve které se nejdříve celá plocha smaže a nadále pak zavolá **paintComponent**, aby se celá plocha vykreslila znovu. Překreslení celé plochy se provádí vždy, když

se provedou změny v tomto zápise a to například v kládáním nových tabů, jejich přemisťování, editace, mazání a jiné akce s tím spojené.

5.4. Třída TextNoteGraphicsComponent

Poslední komponentou, která se nachází v JAppletu NoteTest, je objekt **displayTextNote** ze třídy ***TextGraphicsComponent***.Tato komponenta obsahuje pouze jeden ***Vector*** seznam a to **seznamTextNote** ze třídy **TextNote**. Dále obsahuje další parametry jako jsou hodnoty rozměrů kreslící plochy a jiné.Tato třída tedy obsahuje další třídu a to **TextNote**, jejíž objekty se nachází v seznamTextNote.

5.4.1 Třída TextNote

Tato třída obsahuje obdobné parametry jak třída Note jako jsou **ton** pro určení tónu noty, **okt** pro určení oktávy noty, **pulton** jestli je ton noty je zvyšen o pultón a **usek** znázorňující pořadí této noty v zápise. Tato třída má opět metodu pro své vykreslení a to **public void vykresli(Graphics g)**. V této metodě se tedy podle čísla tónu určí náležející písmeno z tónové řady (c, d, e, f, g, a, h), dále podle toho jestli je tón zvýšen o půltón se přidá přípona "-is"
(např.: "cis"). Nakonec podle čísla oktávy ve které se nachází daný tón se přidají příslušné čárky " ' ". Tato výsledná nota se pak zapíše do kreslící plochy na pozici určenou daným úsekem jako text.

5.4.2 Vytvoření a vykreslení komponent

Komponenty ze třídy TextNote se vytváří pouze převodem z notového zápisu ve tříde NoteGraphicsComponent. Proto se tedy tento zápis nedá přímo měnit. Celý obsah komponenty se pak vykreslí opět pomocí Override metody **public void paintComponent(Graphics g)** a to tak, že se v cyklu volají na jednotlivé komponenty ze seznamu seznamTextNote metoda **vykresli**, a tím se tedy postupně vykreslí všechny tyto textové noty.

5.5 Třída Guitar pro přehrávání hudby

Nyní když již jsou v notovém, či tabovém zápise vloženy noty, tak je posledním úkolem tento zápis následně přehrát. Pro tento úkol slouží třída **Guitar,** která je rozšířena jako **vlákno.**

Nejdůležitějším úkolem objektu této třídy je přehrát vytvořený notový zápis. Tato třída dědí ze třídy ***Thread***. To znamená, že objekt této třídy se chová jako vlákno. Třída obsahuje Override ametodu **run()** ze třídy **Thread**, kterou se dané vlákno spouští. Tím se tedy spustí paralelní běh této části programu k ostatním částím programu. Tento paralelní běh je pro přehrávání a další funkce výhodný, proto jsou zde vlákna využita.

Jedním z nejdůležitějších objektů v této třídě je pole výsledných tónů, které se budou přehrávat. Toto pole – **tony[]** je ze třídy **Tone.** Dalšími třídními proměnnými jsou dva sdílené objekty **sh1** a **sh2** ze třídy ***shared***. Tyto dva objekty jsou určeny pro komunikaci tohoto vlákna s dvěma dalšími vlákny z grafických komponent tříd NoteGraphicsComponent a TabGraphicsComponent, které slouží pro zvýrazňování přehrávaných not a tabů.

5.5.1 Třída Tone

Objekt této třídy, vytvořené přímo pro třídu Guitar, představuje jeden konkrétní tón se svými vlastnostmi, které jsou ve tvaru, který je potřebný pro přehrávání tohoto tónu pomocí syntetizátoru ve třídě Guitar. Protože seznam not ze třídy NoteGraphicsComponent má noty uloženy v jiném tvaru, než jaký je vyžadován, je ve třídě Play, proveden převod na tyto tony a vytvoření nového pole tónů. Tato třída má tři nejdůležitější vlastnosti, které vyplývají z hudební teorie a to **hlasitost** tónu, která je reprezentována proměnnou **int velocity, délka trvání** tónu – proměnná **int duration**, hodnota v

milisekundách, **výška** tónu – proměnná **int tone**, která může nabývat hodnot 0 – 127. Toto číslo pak určuje pořadí požadovaného tónu ve vybraném kanálu zvukové banky. U noty se však nijak neudává přímo časový údaj v žádné časové jednotce, jak dloho tato nota má znít. To se určí pomocí dalších dvou parametrů. Prvním je **ND**, což je hodnota délky tónu v jednom taktu. Pokud tedy tón zní celý takt, má hodnotu 1. Pokud jen půl taktu, tak má hodnotu 0.5 a takto se dá pokračovat dále. Podle této hodnoty délky noty v taktu jsou pak odvozeny názvy not. Nota celá, půlová, čtvrťová atd. Tato proměnná tedy nemůže v našem případě přesáhnout hodnotu 1. Tato hodnota nám však nestačí pro určení přímo časové hodnoty délky tónu. K tomu potřebujeme ještě druhý parametr a to **tempo.** Tato hodnota, jak vyplývá z teorie, nám udává počet úderů noty čtvrťové za jednu minutu.

Hodnoty proměnných **ND** a **tempo** už tedy stačí pro určení délky trvání tohoto tónu. Výpočet se provede podle jednoduchého vzorce, který je odvozen z vědomostí o tempu a délky noty v taktu, což je znázorněno na obrázku *Obrázek 9*.

```
this.duration=(int)((ND*60*4*1000)/(tempo));
```

Obrázek 9: Výpočet doby trvání noty

Hodnota proměnné **velocity**, tedy hlasitosti, pro nás není tak důležitá, protože notový zápis se bude přehrávat s konstantní hlasitostí a je tedy nastavena na jednu hodnotu, která se dále nebude měnit. V tomto případě má hodnotu 110.

5.5.2 Běh samotného vlákna

Po vytvoření vlákna ve třídě **Play**, se před jeho spuštěním nejdříve musí nastavit sdílené objekty pro komunikaci mezi vlákny. Dále pak tomuto vláknu musíme předat pole not, které budou přehrávány. Ty získáme tedy převedením not ze třídy Nota do tónů ze třídy Tone. Jakmile je tohle všechno nastaveno, může se vlákno spustit. To se provede zavoláním metody **start()** na toto vlákno. Tím se spustí ale metoda **run()**, která se tedy nevolá přímo. Tato metoda tedy představuje běh celého vlákna. Obsah této metody je zobrazeno na obrázku *Obrázek 10.*

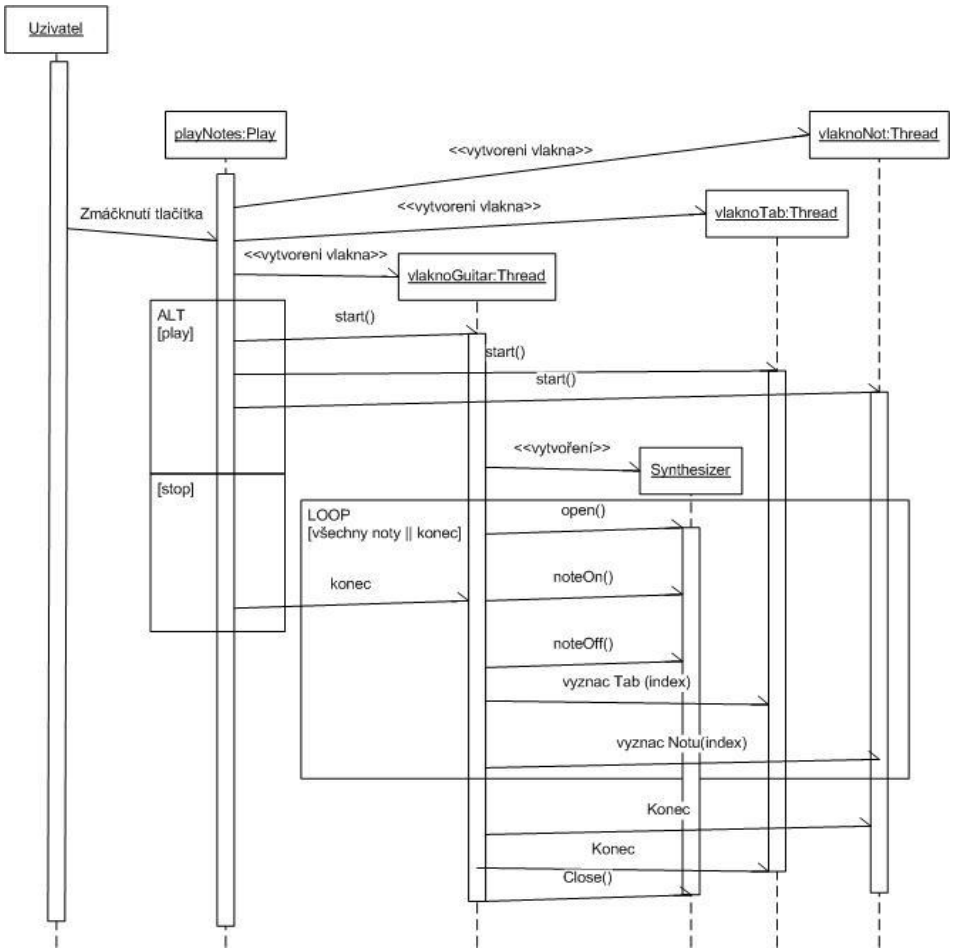
```
public  void run()
{
	int nNoteNumber = 0;
	int nVelocity = 0;
	int nDuration = 0;
	Synthesizer  synth = null;
	synth = MidiSystem.getSynthesizer();


synth.loadAllInstruments(MidiSystem.getSoundbank(Guitar.class.getResourceAsStream("
soundbank-deluxe.gm")));

	synth.open();
	MidiChannel []channels = synth.getChannels();
	MidiChannel  channel = channels[7];

	for(int i=0;(i<tony.length) && !(konec);i++)
	{
		sh1.index=i;
		sh2.index=i;
		nNoteNumber = tony[i].tone;
		nVelocity = tony[i].velocity;
		nDuration = tony[i].duration;

		channel.noteOn(nNoteNumber, nVelocity);
		Thread.sleep(nDuration);
		channel.noteOff(nNoteNumber);
	}

	Thread.sleep(nDuration);
	sh1.index=-1;
	sh2.index=-1;
	synth.close();
}
```

Obrázek 10: Obsah metody run

V tomto kódu nejsou vyznačeny try – catch příkazy, které by měly ohraničovat různé volání metod, u kterých by mohlo dojít k vyhození výjimky. Pro pochopení funkce to však není potřeba. Při implementaci po nás vývojové prostředí tyto příkazy bude požadovat.

V tomto vláknu se tedy nejdříve vytvoří prázdný syntetizátor, do kterého získáme nový syntetizátor z **MidiSystem**, pokud bude tedy dostupný. Aby bylo možné se tímto syntetizátorem pracovat, musí se nejdříve otevřít a to pomocí metody **synth.open()**. Tím se vyhradí zdroje zařízení pro naše používání. Dále se načte pole dostupných midi kanálů ze *syntetizátoru* do pole **channels**. Z tohoto pole dostupných kanálů se pak vybere jeden kanál, například kanál s indexem č. 7. Tento kanál se uloží do proměnné **channel.** Dále již následuje cyklus **for**, ve kterém se postupně berou jednotlivé tóny z pole **tony**. Z aktuálního tónu se uloží do proměnných **nNoteNumber** číslo tónu, do **nVelocity** hlasitost a do **nDuration** délku přehrávání tónu. Pro zapnutí tónu se zavolá na námi vybraný kanál metoda **channel.noteOn(nNoteNumber, nVelocity)** s danými parametry. Třetí parametr nDuration se použije tak, že na danou dobu se uspí vlákno pomocí **Thread.sleep(nDuration).** Po této uplynulé době se zavolá na daný kanál metoda **channel.noteOff(nNoteNumber)**, která daný tón vypne. Je tedy patrné že tón zněl po celou dobu, kdy bylo vlákno uspané, než byla zavolána metoda noteOff. Tím se tedy zajistily všechny tři vlastnosti, které tón má mít. Další věcí kromě přehrávání jednotlivých tónů se v tomto cyklu nastavuje index sdílených objektu **sh1** a **sh2** na hodnoty indexu právě přehrávaného tónu. Tím se tedy dalším dvěma vláknům **not** a **tab** z tříd NoteGraphicsComponent a TabGraphicsComponent předává index aktuální noty, či tabu, které má dané vlákno v tu chvíly zvýrazňovat. Tohle se pak tedy opakuje pro všechny tóny co jsou v poli, přehrávané v daném pořadí, pokud tedy nenastane to, že uživatel zmáčkne tlačítko „**stop**“. Pokud se tak stane, nastaví se vláknu

ukazatel **konec** na hodnotu true a tím se ukončí právě probíhající cyklus, který přehrává noty. Před ukončením vlákna se syntetizátor zavře a tím se uvolní zdroje, které doteď využíval.
Běh vláken je znázorněný v sekvenčním diagramu na obrázku *Obrázek 11*.



Obrázek 11: Sekvenční diagram běhu vláken

5.5.3 Záznam hudby do souboru

Záznam se spustí pomocí tlačítka "**Rec**" ze třídy **Play**. Záznam se provádí tak, že se spustí vlákno stejně jako při spuštění tlačítkem "**Play**", avšak s tím, že se nastaví ukazatel **record** na hodnotu true, která ve vláknu určuje, že se bude zaznamenávat hudba do souboru.
Pro záznam hudby slouží v tomto vláknu objekt **sequence** třídy ***Sequence***. V sequenci se pak vytvoří nová stopa – **track**. Záznam hudby se neprovádí žádným vstupním streamem, ale pomocí signalizačních zpráv. Zprávy obsahují záznamy o čísle noty a její hlasitosti a času, jak dlouho tato nota zní. Do stopy se pak zaznamenávají, tyto zprávy vždy když se přehrávají příslušné noty. Před ukončením vlákna se tato sequnce uloží do souboru, jehož místo bylo vybráno uživatelem. Tento soubor je ve formátu **.mid**. Tento formát je typický pro MIDI standart. Tyto soubory se dají otevřít ve většině dostupných programech pro přehrávání hudby.

5.6 Metody pro převod mezi jednotlivými zápisy

Protože jednotlivé zápisy tabů, not a konkrétnách tónů nemají jednotný formát, bylo potřebné naimplementovat metody pro převod mezi nimy.

5.6.1 Převod not ze třídy Nota do tónů ze třídy Tone

Tento převod se provádí vždy před spuštěním vlákna **Guitar** a následně se převedené tóny předají tomuto vláknu. Největší problematikou tohoto převodu je to, že pole třídy Tone, které jsou potřebné pro přehrávání výsledného notového zápisu ve vláknu Guitar, mají jednotlivé tóny seřazeny lineárně za sebou, kdy každý tón má svoje vlastní číslo. Výšku tónu je tedy možné ihned určit. Avšak objekty ze třídy Nota tuto posloupnost nemají rovnoměrnou a jejich výška tónu se musí určit pomocí tří parametrů – **ton**, **okt**, **pulton**. Kombinací těchto parametrů se provede postupně přepočet pro každou notu a výsledkem je tedy pole tónů třídy **Tone**, které je již použitelné pro přehrávání.

5.6.2 Převod z tabů na noty

Dalším převodem, který se v programu provádí je převod z tabů ze třídy Tab v grafické komponentě TabGraphicsComponent na na noty ze třídy Nota v komponentě NoteGraphicsComponent. Pro tento převod slouží metoda **TabToNote**, která se nachází v hlavní třídě NoteTest, avšak většina převodu se provádí již v konkrétním tabu ze třídy **Tab**. Tento převod se provádí pokaždé, když se provede změna v komponentě TabGraphicsComonent.

Základním úlohou tohoto převodu je z parametrů tabu – struna, prazec, určit parametry noty – ton, okt, pulton. To se provede sadou operací, které spočítají podle znalosti toho, kde které struně na kterém pražci se nachází jaká nota. Tento převod je oproti zpětnému převodu z not na taby jednodušší.

5.6.3 Převod z not na taby

Posledním převodem, který program provádí je převod not třídy Nota na taby ze třídy Tab. Tento převod je náročnější, protože pro každou notu existuje několik různých tabů, které se nacházají na jiných strunách a jiných pražcích a nelze jednoznačně vybrat odpovídající tab k dané notě. Špatně zvolený tab však není žádnou chybou, protože na tomto tabu bude znít nota správně, avšak může se stát to, že zápis v tabu pak bude pro kytaristu hůře hratelný, než kdyby se zvolila jiná pozice tabu. Proto by se tento převod dal více vylepšit, kdy by se analyzovaly všechny taby jako celek a tím by se tento zápis případně sám upravoval. V tomto případě však je zde převod proveden jednoduše, kdy se bere nejbližší tab, který je k dispozici.

5.7 Distribuce celé aplikace pomocí technologie Java Web Start

Jak již z názvu této aplikace – **Webová kytara** vyplývá, bude možné tuto aplikace spustit z webového prohlížeče přímo z internetu. K tomu jsem využil schopností technologie Java Web Start. Následující text je čerpán ze zdroje **[6].**

Je to technoligie, která byla vyvinuta firmou Sun Microsystems pro aplikační software, který je určený pro Java Platformu. Nejdůležitější funkcí tohoto fremworku je *umožnit uživateli spouštět pomocí webového prohlížeče javovské aplikace a to přímo z internetu*. Narozdíl od Java Appletů, které také mouho spouštět aplikaci přímo z internetu, Web Start aplikace neběží přímo v okně webového prohlížeče, ale v samostatném okně. Také jejich bezpečnostní mechanismus – Sandbox, kerý je určený pro oddělení bežících programů, nemá tolik omezení jako Applety. Tyto omezení se však dají i měnit. Java Web Start poskytuje sérii tříd v balíčku **javax.jnlp**, který poskytuje aplikaci několika služeb.Sun představil verzi Web start 1.0 v roce 2001. Od standartní edice Java Platformy – J2SE se stává Web Start standartní částí **Java Runtime Enviroment (JRE)**. Web Start využívá ke své práci **Java Network Launching Protocol (JNLP)**, který je s Web Start úzce spjat. Tento JNLP protocol, který je definován pomocí XML schematu, specifikuje, jak spouštět Web Start aplikace. JNLP obsahuje sadu

pravidel, jak naimplementovat spouštěcí mechanismus. Příslušný JNLP soubor pak obsahuje informace, jako jsou umístění jar souboru, jméno spouštěcí main metody aplikace a další parametry, jako jsou například bezpečnost. Správně nastavený prohlížeč pak předá tento JNLP soubor javovskému provoznímu prostředí (JRE) , které započne stahování příslušné aplikace na počítač a po dokončení stahování jej spustí. Velmi důležitým rysem Web Startu je to, že pokud uživatel nemá nainstalovanou Javu, automaticky spustí stahování a instalaci příslušného JRE. Dalším rysem je to, že specifukuje, jakou JRE verzi bude program potřebovat ke spuštění. Zajímavou vlastností je také to, že uživatel nemusí zůstávat připojený k internetu, aby spouštěl stažené programy, protože se spouštějí z lokální cache.

Web Start tedy funguje tak, že potom co uživatel klikne na odkaz, který by měl spustit danou aplikaci, webový prohlížeč předloží URL webserveru, který odpoví namísto HTML souboru JNLP souborem. JNLP klient pak analyzuje tento soubor, požádá o všechny zdroje jako je jar soubor, počká než se všechny zdroje získají a na závěr spustí aplikaci.

5.7.1 Implementace a použití technologie Java Web Start

Abychom mohli naši vytvořenou aplikaci spustit pomocí této technologie, je nutné splnit tři základní body a to: 1. Vytvořit JAR archiv naší aplikace, 2. Podepsat JAR archiv, 3.Vytvořit JNLP soubor.

5.7.1.1. Vytvoření JAR archivu

Tento JAR archiv je specifický pro Javu a je to v podstatě stejný archiv jako ZIP, takže je možné jej rozbalit tak jako ZIP pomocí například Total Commader. V tomto archivu se nachází složka s balíčkem tříd naší aplikace a soubor s manifestem, ve kterém jsou informace důležité pro spuštěné aplikace.

Vytvoření JAR arichivu se může provést pomocí vývojového prostředí. V NetBeans se to provádí pomocí stlačení pravého tlačítka na příslušný projekt, který chceme zkompilovat, a vybere možnost Clean and Build. Po tomto se buildne jar soubor a uloží se do složky dist v adresáři příslušného projektu, např.: C:\Users\bura\Sound2\dist\Sound2.jar.

Dalším způsobem vytvoření JAR souboru je pomocí příkazové řádky. Po zadání příkazu: **`jar cf Sound.jar NoteTest.class`** do příkazové řádky, se třída NoteTest.class zabalí do archivu Sound.jar. Pokud nemáme definouvanou cestu k Javě v proměnné PATH, je nutné tuto cestu celou vypsat. Příklad cesty: C:\Program Files\Java\jdk1.6.0_03\bin. V adresáři bin se nachází všechny potřebné nástroje.

5.7.1.2. Podepsání JAR archivu

Pokud chceme, aby naše aplikace měla větší práva jako applet, tedy abychom mohli přistupovat na disk či síť, je nutné tento JAR archiv podepsat. Pokud však tyto práva v aplikaci nevyžadujeme, můžeme tento bod vynechat. V mé aplikaci to však vyžadováno je.

Podepsání archivu se provádí pomocí keytool nástroje. To můžeme provést pomocí příkazové řádky:

1.Vytvoříme vlastní sadu klíčů příkazem:

```
keytool -genkey -keystore mykeys -alias myself
```

2.Vytvoříme self-signed certifikát:

```
keytool -selfcert -alias myself -keystore mykeys
```

3.Nakonec podepíšeme certifikátem naši aplikaci:

```
jarsigner -keystore mykeys Sound.jar myself
```

Opět pokud nemáme definovanou cestu k Javě v proměnné PATH, je nutné vypsat celou cestu ke keytool a jarsigner.

5.7.1.3. Vytvoření JNLP souboru

JNLP soubor s koncovkou .jnlp, jak už bylo zmíněné, je soubor formátu značkovacího jazyku XML, který poskytuje Web Startu informace pro spuštění, čí jiné informace, jako je cesta k souborům aplikace, nastavení zabezpečí, spouštěcí třída apod.

Obsah tohoto souboru pak je zobrazen na obrázku *Obrázek 12*.

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+"
  codebase="http://tombura.wz.cz">
<information>
  <title>Sound</title>
  <vendor>Java Developer Connection</vendor>
  <homepage href="" />
  <description>Demonstration of JNLP</description>
</information>
<offline-allowed/>
<security>
  <all-permissions/>
</security>
<resources>
  <j2se version="1.2+" />
  <jar href="Sound.jar"/>
</resources>
<application-desc main-class="sound2.NoteTest" />
</jnlp>
```

Obrázek 12: Obsah JNLP souboru

Popis důležitých částí JNLP souboru:

codebase="http://tombura.wz.cz" – určuje zdroj dané aplikace. Soubor JAR se tedy může se tedy nacházet na serveru, ale může zde být i cesta ke složce souboru na disku

<security>

<all-permissions/>

</security>

- Nastavení zabezpečení. V našem případě chceme přistupovat na disk, takže zadáme tag

<all-permissions/>, který určuje, že máme všechna možná práva povolena.

<jar href="Sound.jar"/>

- Nastavení zdrojového souboru aplikace, která bude spuštěna.

<application-desc main-class="sound2.NoteTest" />

- Nastavení spouštěcí main třídy. Nesmí se zapomenout uvést u balíček, ve kterém se nachází tato třída.

Soubor tedy vytvoříme pomocí nějakého textového editoru, jako je například PsPad, ale stačí i obyčejný NotePad a uložíme jej s koncovkou .jnlp.

5.7.2 Použití Web Start v aplikaci Webová kytara

Pokud tedy máme již vytvořený archiv JAR z naší aplikace, je tento archiv podepsaný a máme vytvořený JNLP soubor, je nutné umístit zdrojové soubory na místo, které bylo určeno v souboru jnlp v parametru codabase. Pokud je zdrojová cesta na disku, pak pouze nakopírujeme JAR soubor na toto místo. Poté tedy stačí otevřít normálním dvojklikem soubor JNLP a spustí se již samotný Web Start, který si načte aplikaci z cesty na disku a spustí jej. Pokud však chceme danou aplikaci spouštět z internetu, což je hlávním účelem Web Start a je to potřebné v naší aplikaci Webová kytara, musíme JAR soubor nahrát na nějaký webový server. Já jsem si pro tyto účely založil účet na webzdarma.cz, což pro tyto účely stačí. Můj JAR soubor jsem umístil do hlavního adresáře své domény. URL k tomuto souboru je tedy: http://tombura.wz.cz/Sound.jar, a adresář s tímto souborem je tedy nadefinován v souboru JNLP takto: **codebase="http://tombura.wz.cz".** Soubor JNLP pak můžeme také umístit na tento server, ale i kamkoliv jinam, protože na jeho umístění tolik nezáleží. Pro spuštění aplikace pomocí Web Start z webu, pak už jen stačí stáhnout tento JNLP soubor ze serveru a spustit jej. Web Start pak sám automaticky stáhne do cache paměti příslušný spouštěcí zdrojový soubor. Po dokončení stahování se pak spustí samotná aplikace. Je vhodné si pro tuto aplikaci vytvořit svoji webovou stránku, na které bude odkaz na zmíněný JNLP soubor například jako tlačítko pro spuštění aplikace. V mém případě to lze provést přímo na hlavní stránce: http://tombura.wz.cz/

6. Uživatelská část programu

6.1 Popis hlavních částí programu

Základem celého programu je uživatelské grafické rozhraní, které je rozděleno na tři základní části: **První část** je umístěna v horní části okna aplikace a skládá se z ***hlavního panelu*** a ***grafického rozhraní***. V tomto panelu se nachází tlačítka a textové pole a slouží pro přehráváná výsledného notového zápisu a další ovládání celkového programu. Grafické rozhranní (TextGraphicsComponent), které se nachází hned pod hlavním panelem, slouží pro textové zobrazení tónů, které jsou zadány do notového zápisu, tedy správné názvy tónů i se značkami, do jaké patří oktávy apod.

Druhá část se nachází ve střední části okna a skládá se z ***grafického rozhranní pro zápis not do notové osnovy (NoteGraphicsComponent)*** a z příslušného ***panelu s tlačítky***, která jsou potřebná pro *zadávání*, *editaci* a *mazání not*.

Samotné ***grafické rozhranní*** se pak skládá z 5 hlavních linek, které jsou vyznačeny černě, a 6ti pomocných linek vyznačených světle šedě, z kterých se 2 nachází nad hlavnímy linkami a další 4 se nachází pod hlavnímy linkami.

Třetí část se nachází ve spodní části okna programu a opět se skládá z příslušného panelu s potřebnými tlačítky a z grafického rozhranní, které slouží pro zápis tabů. Grafické rozhranní (TabGraphicsComponent) se skládá z 6ti linek. Každá znich pak představuje jednu kytarovou strunu. Každá struna je označena svým názvem, tak jak jsou nalazeny (EADGBE).

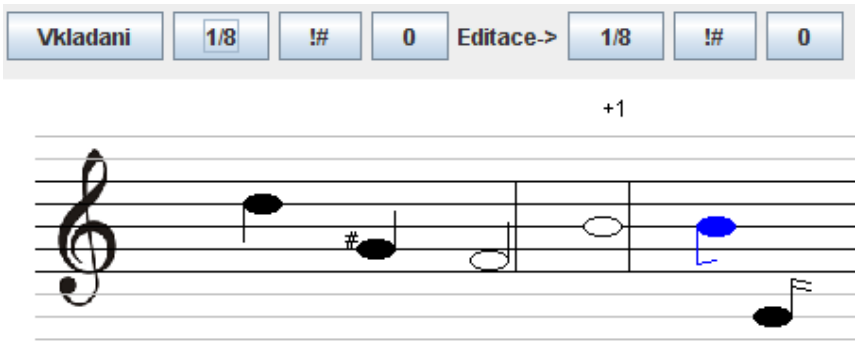
6.2. Práce s programem

6.2.1.Zadávání, editace a mazání not

Pro tento účel slouží druhá část grafického rozhranní. Pokud chceme do notové osnovy zadávat nové noty, musí být nejdříve v panelu s tlačítky první tlačítko přepnuto na funkci "**Vkládání**". Pak již je možné zadávat noty do notové osnovy. To se provádí kliknutím myši do notové osnovy a to přímo do na konkrétní linku, či do konkrétní mezery mezi nimy. To na jaké lince či v jaké mezeře se nota nachází pak určuje výšku tónu noty, která se zvyšuje od nejnižší linky po nejvyšší. Dalšímy tlačítky může uživatel ovlivnit další vlastnosti přidávané noty. Prvním tlačítkem je tlačítko pro změnu ***hodnoty délky noty v jednom taktu***. Tato délka může v našem případě nabývat pěti hodnot – ***1, ½. ¼, 1/8 , 1/16***. Tyto hodnoty pak odpovídají názvům not – nota ***celá, půlová, čtvrťová, osminová, šestnáctinová***. Podle toho mají také vkládané noty svůj tvar. Druhé tlačítko slouží pro to, jestli bude přidávaná nota ***zvýšena o půtón***. Toto tlačítko tedy bude nabývat hodnoty "**#**", pokud bude daná nota zvýšena o půltón, a hodnoty "**!#**" pokud tato přidávaná nota nebude zvýšena. Posledním tlačítkem ovlivňující přidávané noty je tlačítko, které slouží pro ***zvýšení tónu noty o jednu oktávu.*** Toto tlačítko pak nabývá hodnot "**0**" pokud uživatel přidává notu, která není zvýšena o oktávu, či hodnoty "**+1**" pokud přidávaná nota bude zvýšena o jednu oktávu.

Editace přidaných not pak může uživatel provádět tažením not v notové osnově a tím změnit výšku noty či pořadí not. Další editaci konkrétní noty pak může provádět pokud kliknutím označí notu, kterou chce editovat. Tato nota se pak zbarví modře. Obdobnými tlačítky v panelu pak může měnit opět délku noty, zvýšení noty o půltón či, zvýšení noty o celou oktávu.

Pokud je v panelu přepnuto tlačítko na "***Mazání***", může pak uživatel kliknutím myši na vybrané noty, tyto noty mazat z notového zápisu. Popisované grafické rozhranní je pak vidět na obrázku *Obrázek 13*



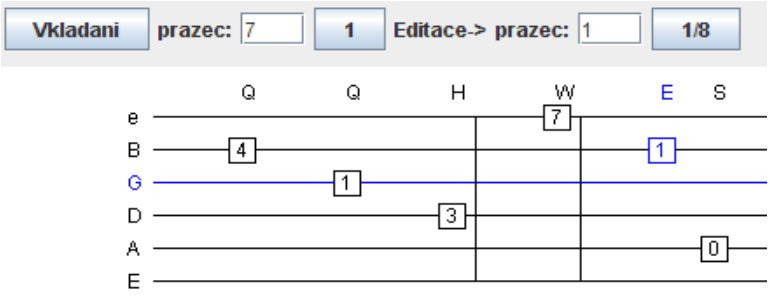
Obrázek 13: Grafické rozhranní části pro zadávání not

6.2.2. Zadávání, editace a mazání tabů.

To se provádí ve třetí části programu. Pokud je v příslušném panelu s tlačítky přepnuto tlačítko s funkcí "***Vkládání***" může uživatel kliknutím do zápisu tabu na příslušné lince a v příslušném úseku přidávat nové taby. Dalšími ovládacími prvky pak může ovlivnit vlastnosti přidávaných tabů. Zadáním čísla do ***textového pole*** a následným potvrzení enterem se nastavuje číslo ***kytarového pražce***, na kterém se přidávaný tab nachází. Toto číslo může být pouze v rozsahu 0 – 22. Dalším tlačítkem, které je obdobné, jako v druhé části programu je ***tlačítko na změnu hodnoty délky tabu v jednom taktu***. Tato délka se u každého tabu pak znázorňuje nad každým tabem příslušným písmenem. Tyto taby pak může uživatel dále editovat tak, že kliknutím a tažením může měnit jejich umístění na strunách, či může měni pořadí přehrávaných tabů. Další editace tabů může provádět tak, že kliknutím myši označí tab, který chce editovat a obdobnými prvky jako pro přidávaní tabů, může tyto vlastnosti měnit.

Pokud je v panelu přepnuto tlačítko na "***Mazání***", může pak kliknutím myši mazat vybrané taby ze

zápisu. Popisované grafické rozhranní je pak vidět na obrázku *Obrázek 14*.



Obrázek 14: Grafické rozhranní pro zápis tabů

6.2.3.Převod mezi jednotlivými zápisy tabů a not

Tento převod provádí program automaticky. Tedy při vkládání prvků do jednoho ze zápisů se rovnou převádějí a vkládají prvky do druhého zápisu. Zároveň se také provádí převod do textového zápisu not.

6.2.4.Přehrávání hudby ze zápisu, uložení hudby

Pro přehrávání je v hlavním panelu určeno tlačítko "**Play**". Nejdříve je však nutné mít v textovém poli zadanou hodnotu tempa, tedy rychlosti, jakou se bude skladba přehrávat. Toto tempo může nabývat hodnot 40 – 208, kdy skladba s tempem s nejnižší hodnotou se bude přehrávat nejpomaleji a skladba s nejvyšší hodnotou tempa se bude přehrávat nejrychleji. Po stlačení tlačítka "Play" se pak přehrává výsledná hudba odvozená od zadaných not a zadaného tempa. Před přehráním samotného zápisu se ještě přehrávají čtyři doby metronomu, aby uživatel byl připravený, v jakém tempu se bude skladba přehrávat. V průběhu přehrávání not a se pak postupně zvýrazňují v notovém a tabovém zápise právě přehrávané noty/taby. Toto přehrávání jde také ukončit dříve, než se celý zápis přehraje a to pomocí tlačítka "**Stop**". Tuto hudbu je pak možné zaznamenat do souboru a to tlačítkem "**Record**". Po stisknutí tohoto tlačítka program nejdříve vyzve uživatele, aby zadal cestu a název souboru, který se má vytvořit. Následně se pak tedy opět přehraje notový zápis, ale s tím že se v průběhu hudba zaznamenává a nakonec se zapíše do souboru a to ve formátu **.mid**, což formát pro ukládání MIDI zvuku.

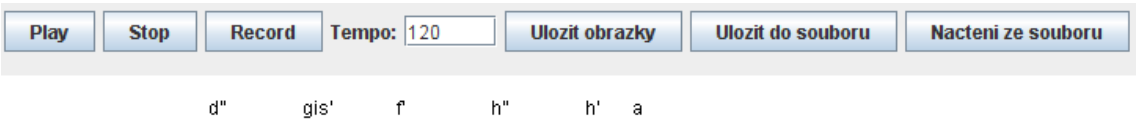
6.2.5.Uložení a načtení zápisu

Pokud si chce uživatel uložit jednotlivé zápisy pro jeho vlastní potřebu, může to provést stlačením tlačítka "**Ulozit obrazky**". Po stlačení tohoto tlačítka program požádá o zvolení místa a názvu souborů, které se mají uložit. Následně se na toto zvolené místo uloží tři obrázky ve formátu JPEG, prvím obrázkem je zápis textu jednotlivých not, druhým obrázkem je zápis not v notém zápise a třetím obrázkem je zápis tabů.

Pokud si však chce uživatel uložit zápis pro další použití v tomto programu, může to provést tlačítkem "**Uloz do souboru**". Tím se tedy opět program zeptá na cestu a název ukládaného souboru. Následně se pak uloží noty do textového souboru v určitém formátu, ve kterém se na každém řádku ukládá jedna nota a jednotlivé hodnoty na tomto řádku pak představují vlastnosti této konkrétní noty.

Pokud uživatel chce načíst noty ze souboru, pak stlačí tlačítko "**Nacti ze souboru**". Tím se tedy do notového a tabového zápisu vloží noty z již uloženého souboru, který je formátu, určeném pro tento program.

Grafické rozhranní pro prvky pro přehrávání, ukládání a načítání notového zápisu jsou vidět na obrázku *Obrázek 15*.



Obrázek 15:Grafické rozhranní pro ovládání celého programu a textový zápis not

7. Závěr

Výsledkem práce na tomto projektu je tedy funkční aplikace Webová kytara, kterou je možné spustit z webového prohlížeče na internetu. Tato aplikace pak nabízí vkládání not do notového zápisu, který je možno dále upravovat. Dále tato aplikace nabízí vkládání not pomocí tabů, do zvláštního zápisu těchto tabů. Tyto zápisy si nazvájem odpovídají a v průběhu zadávání not či tabů se provádí jejich vzájemný převod. Aplikace také poskytuje písemný zápis not. Jednou z nejdůležitějších funkcí je přehrání notového zápisu a tato hudba může být zaznamenána do souboru. Zápisy not a tabů se nakonec dají uložit jako obrázky pro tisk, či jiné použití a také se dají uložit do souboru, který pak může sloužit pro další použití v tomto programu, kdy se dají noty z tohoto souboru opětovně načíst.

Tato aplikace staví na základních znalostech hudební teorie a pojmů. Důležitou věcí je také to, že využívá k práci s hudbou aplikační rozhranní Java Sound API a pro spouštění aplikace z internetu využívá technologie Java Web Start, která může obstojně nahradit Applety, které také slouží pro spouštění aplikací z internetu.

Možné rozšíření této aplikace by mohlo spočívat ve vylepšení celkového grafického rozhranní, protože při tvorbě spíše hrála roli funkční stránka, nežli vzhledová. Další možné programové rozšíření aplikace by mohlo být také v oblasti přídání dalších hudebních nástrojů. To se dá provést vložením nové zvukové banky, která poskytuje jiné nástroje, než jsou k dispozici. K tomu by se tedy pak dále mohl program rozšířit o zápis pomocí i jiného než kytarového tabu, který bude určený pro případné nové nástroje.

Forma této aplikace je spíše určena pro začátečníky v oblasti hudby, protože celková funkce nezahrnuje úplně všechny prvky potřebné pro zapsání složitých skladeb, což obdobné programy na profesionální úrovni poskytují. Avšak pro zapsání jednoduché skladby je tento program dostačující a má dobrý základ pro to, aby jej bylo možno vylepšit na profesionálnější úroveň.

Literatura

[1] Zenkl, Luděk. *ABC hudební nauky*. Editio Praha, 2003.
ISBN 80-86385-21-3

[2] *Ultimate-guitar.com*
URL:< http://www.ultimate-guitar.com/tabs/l/led_zeppelin/>

[3] *www.midi.org*
URL:< http://www.midi.org/>

[4] *java.sun.com*
URL:< http://java.sun.com/docs/books/tutorial/sound/>

[5] java.com
URL: < http://java.com/download>

[6] *wikipedia.org*
URL:< http://en.wikipedia.org/wiki/Java_Web_Start>

Seznam příloh:

Příloha 1: NetBeans projekt celého programu se všemi zdrojovými třídami a potřebnými soubory na přiloženými na CD

Příloha 2: Webovou stránku pro prezentaci funkčnosti programu a zkompilovaná výsledná aplikace i s potřebným souborem pro správnou funkci přiložené na CD

Příloha 3: Instalační program Java prostředí přiložené na CD