Praktikum iz objektno-orijentisanog programiranja (13S112POOP)
Projektni zadatak – Java

Napisati skup klasa sa odgovarajućim metodama, konstruktorima, operatorima i destruktorima za realizaciju softverskog sistema za sviranje klavira. Potrebno je obezbediti mogućnost sviranja pomoću tastature ili miša, kao i učitavanje i snimanje kompozicija. Podržani formati fajlova treba da budu TXT i MIDI a treba predvideti i mogućnost proširenja drugim formatima. Opis formata navedenih fajlova je dat u prilogu ovog dokumenta.

Korisnik (naručilac) softvera, želi da softver pruži sledeće funkcionalnosti:

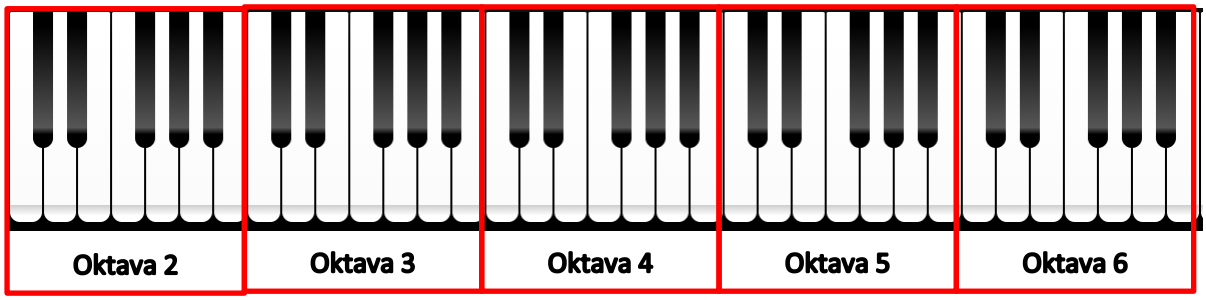
- Interakciju sa korisnikom putem grafičkog korisničkog interfejsa
- Učitavanje podataka
    - učitavanje podataka o kompoziciji
    - učitavanje podataka o notama
- Osnovnu obradu
    - prikaz podataka o kompoziciji
    - sviranje pomoću tastature ili miša
    - snimanje kompozicije
- Eksportovanje kompozicije
    - formatiranje fajlova po TXT ili MIDI formatu
- Kraj rada

Za uspešno rešenje zadatka potrebno je izvršiti analizu zahteva. Kao rezultat analize, potrebno je dopuniti i precizirati funkcionalnu specifikaciju softverskog alata. Na osnovu specifikacije, potrebno je napisati sistem klasa u jeziku Java koje realizuju traženi softver. U nastavku su navedeni neki elementi specifikacije. Od studenata se očekuje da dopune one stavke koje nisu dovoljno precizno formulisane, odnosno dodaju nove stavke (tamo gde to ima smisla) ukoliko uoče prostor za unapređenje. Izmene i dopune specifikacije mogu da donekle odudaraju od zahteva naručioca softvera u onoj meri u kojoj to neće narušiti traženu funkcionalnost. Takođe, priloženi UML dijagram koji opisuje zahtevani softver se ne mora obavezno poštovati, već samo predstavlja skicu potencijalnog rešenja. Prilikom izrade specifikacije voditi računa o potencijalnom unapređenju softvera na osnovu naknadnih zahteva.

Prilikom izrade rešenja, od studenata se očekuje intenzivno korišćenje svih onih mogućnosti koje pružaju specifikacija i biblioteke jezika Java, kao što su kolekcije, algoritmi, regularni izrazi, iteratori, lambda izrazi i sl. **Rešenja koja ne vode računa o ovom aspektu neće moći da dobiju maksimalan broj poena.** Takođe, voditi računa o **objektno orijentisanom dizajnu rešenja**, čistoći, čitkosti i komentarisanju programskog koda.

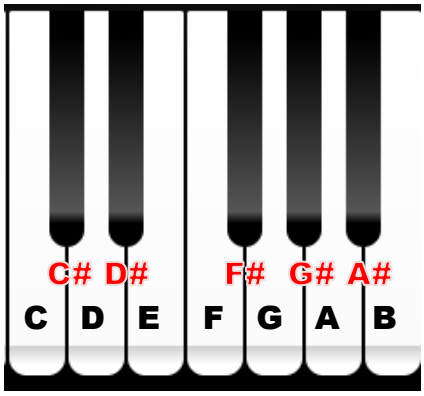
Osnovni pojmovi

Kompozicija se sastoji od muzičkih simbola. Muzički simboli su nota i pauza. Muzički simbol ima svoje trajanje koje se zadaju u vidu razlomka. Nota je opisana oktavom (ceo broj u opsegu 2-6) i visinom (C, D, E, F, G, A, B, poznatije kao: DO, RE, MI, FA, SOL, LA, SI). Izgled uprošćene verzije klavira koja se koristi u ovom projektu sa označenim oktavama dat je na slici 1.



Slika 1. Izgled klavira

Notu je moguće povisiti dodavanjem zanaka # (povisilica), te je povišena nota C u stvari C# (otuda i naziv programskog jezika). Povišene note se na klaviru sviraju na crnim dirkama desno od osnovne note. Složena nota (dvozvuk ili akord) se sastoji od više drugih nota koje se sviraju istovremeno, a sve moraju biti istog trajanja.



Na slici 2. dat je uvećan prikaz jedne oktave sa označenim tonovima. Crnim slovima su na dirkama označeni tonovi koje predstavljaju bele dirke, dok su crvenim slovima označeni tonovi koje predstavljaju crne dirke iznad njih, tj. povišeni tonovi (snizilice nećemo razmatrati u ovom projektu). Raspored tonova je u isti u svim oktavama, a radi jednostavnosti uzeto je da se klavir sastoji od 5 celih oktava.

Slika 2. Izgled jedne oktave

Funkcionalna specifikacija

U nastavku je zadat deo korisničkih zahteva koje treba razraditi i, po potrebi, dopuniti tako da se dobije funkcionalna aplikacija.

Interakcija sa korisnikom

Korisnik može da interaguje sa programom izborom u datom trenutku dostupnih opcija putem grafičkog korisničkog interfejsa. Interakcija može da se vrši putem tastature ili miša. U zavisnosti od izabrane opcije i njenih parametara, program izvršava zadatu opciju ili ispisuje poruku greške. Poruka greške treba da bude što je moguće detaljnija da bi korisniku pomogla da grešku otkloni. Sve eventualne parametre koji su potrebni prilikom rada aplikacije je potrebno zatražiti od korisnika. Ukoliko korisnik ne zada ništa, koristiti vrednosti fiksirane u programu.

Učitavanje podataka

Predvideti postojanje konfiguracionog fajla sa potrebnim mapiranjima karaktera na note i MIDI brojeve. U svakom redu datoteke nalaze se podaci za po jedan ulazni karakter i to: karakter koji se opisuje, tekstualni opis note i MIDI broj note, odvojeni znakom zareza, kao u navedenom primeru, koji opisuje karakter ‘t’ koji se mapira na notu C4 čiji je MIDI broj 60:

t,C4,60

Dato mapiranje potrebno je iskoristiti prilikom iscrtavanja klavira. Na zahtev korisnika na svakoj dirki klavira se iscrtava karakter koji je potrebno pritisnuti na tastaturi da bi se proizveo ton te dirke, što će detaljnije biti opisano u nastavku dokumenta.

Korisnik odabirom ponuđene opcije iz menija i unosom putanje do ulaznog fajla može zahtevati učitavanje kompozicije. Prilikom učitavanja dovoljno je podržati samo tekstualni format fajla. Ulazni fajl se sastoji od sekvence karaktera koji opisuju muzičke simbole. Može sadržati prelaze u novi red, ali njih treba ignorisati. Dekodovanje ulaznog fajla i pretvaranje karaktera u muzičke simbole se vrši uz pomoć mapiranja koje se učitava po pokretanju programa. Na osnovu mapiranja se određuje visina i oktava u kojoj se nota nalazi. Prilikom parsiranja fajla pauze i trajanja nota se određuju prema sledećim pravilima:

- blanko znak označava pauzu u trajanju 1/8
- znak | između simbola označava pauzu u trajanju 1/4
- trajanje nota navedenih unutar uglastih zagrada sa znakom razmaka između svake note je 1/8
- navođenje nota unutar uglastih zagrada bez znaka razmaka između njih označava da sve note treba odsvirati u istom trenutku i sve note su trajanja 1/4
- trajanje nota navedenih van zagrada je 1/4

U nastavku je dat primer dela ulaznog fajla, a u tabeli ispod i tumačenje svakog od simbola:

[Yui] t | Yu [t u y]

Ulazni simboli	**Simboli koje označavaju**	**Trajanje simbola**
[Yui]	note D#4, E4 i F4	1/4 (sviraju se istovremeno)
_ (blanko znak)	pauza	1/8
t	nota C4	1/4
|	pauza	1/4
Yu	note D#4 i E4	1/4 (sviraju se jedna za drugom)
_ (blanko znak)	pauza	1/8
[t u y]	note C4, E4 i D4	1/8 (sviraju se jedna za drugom)

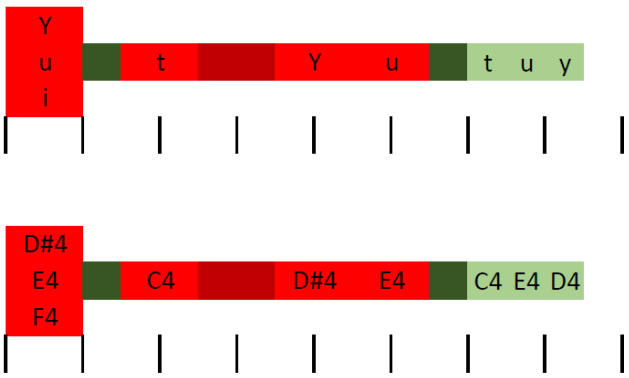
Prilikom učitavanja podataka o kompozicijama i notama koristiti regularne izraze za parsiranje datih fajlova. Predvideti način za oporavak od grešaka u slučaju neispravnog formata datoteke ili nepostojanja tražene datoteke.

Osnovna obrada

Korisniku je potrebno omogućiti sledeće:

- Učitavanje i prikaz kompozicije.
  Korisnik može izabrati da se učitana kompozicija prikazuje u vidu slova iz tekstualnog fajla iz kog je vršeno učitavanje ili u vidu tekstualnih opisa nota, koje je moguće pronaći u fajlu sa mapiranjem. U oba slučaja se sve note koje se sviraju u jednom trenutku ispisuju vertikalno (jedna ispod druge). Prilikom ispisivanja pauza dovoljno je ostaviti prazan prostor. Trajanja 1/4 označiti crvenom bojom pozadine ili slova, a trajanja 1/8 zelenom bojom. Ispod ispisa tekstualnih opisa stavljati markere u vidu crnih linija koje označavaju četvrtine. Na slici 3. prikazani su primeri oba načina ispisa datog dela kompozicije.

[Yui] t | Yu [t u y]



Slika 3. Primeri ispisa kompozicije

- Automatsko sviranje kompozicije.
  Korisnik može zatražiti da se učitana kompozicija odsvira. Potrebno je obezbediti mogućnost privremenog i trajnog zaustavljanja sviranja. Tokom sviranja na prikazanom klaviru potrebno je crvenom bojom obeležavati dirke tonova koji se u tom trenutku sviraju. Takođe, tekstualni prikaz kompozicije je potrebno pomerati ulevo tokom sviranja tako da je prvi prikazani ton uvek onaj koji se trenutno svira. Po završetku sviranja kompozicije potrebno je vratiti tekstualni prikaz u prvobitno stanje.
- Sviranje klavira.
  Korisnik u bilo kom trenutku (i pre učitavanja kompozicije) može svirati po klaviru pomoću miša (klikom na prikazane dirke) ili tastature (pritiskanjem tastera na tastaturi koji odgovaraju dirkama klavira). Obezbediti mogućnost da se na dirkama klavira ispišu tekstualni opisi nota radi lakšeg sviranja. Ukoliko prilikom sviranja klavira postoji učitana kompozicija i korisnik odsvira ton koji se i očekuje potrebno je pomeriti prikaz kompozicije ulevo (kao i kod automatskog sviranja) sve do sledećeg tona koji se svira (navedeni koncept je često moguće naći u igricama koje testiraju kucanje na tastaturi, a u kojima se pojavljuju i pomeraju ulevo reči koje je potrebno otkucati). Ukoliko se očekuje više tonova istovremeno njihov redosled se može zanemariti, ali je potrebno da svi budu odsvirani u kratkom vremenskom intervalu. Po završetku kompozicije ili na zahtev korisnika potrebno je vratiti tekstualni prikaz na početak kompozicije.
- Snimanje kompozicije.
  Potrebno je obezbediti mogućnost snimanja kompozicije. Korisnik može izabrati opciju za početak snimanja, a zatim svira po prikazanom klaviru na jedan od opisanih načina. Po završetku, korisnik bira opciju za snimanje kompozicije koju je svirao. Program treba da obezbedi eskportovanje odsvirane kompozicije u tekstualnom ili MIDI formatu u fajl sa zadatom putanjom. Pauze na početku i kraju kompozicije je potrebno ignorisati. Trajanje tonova određivati na osnovu dužine pritiska tastera.

Kraj rada

Korisnik može da zahteva kraj rada. Od korisnika se traži potvrda za napuštanje programa, uz upozorenje ukoliko program nakon poslednjeg snimanja kompozicije nije generisao neki fajl. Korisniku treba ponuditi mogućnost da eksportuje kompoziciju pre napuštanja programa, ukoliko to želi.

Testiranje rada programa

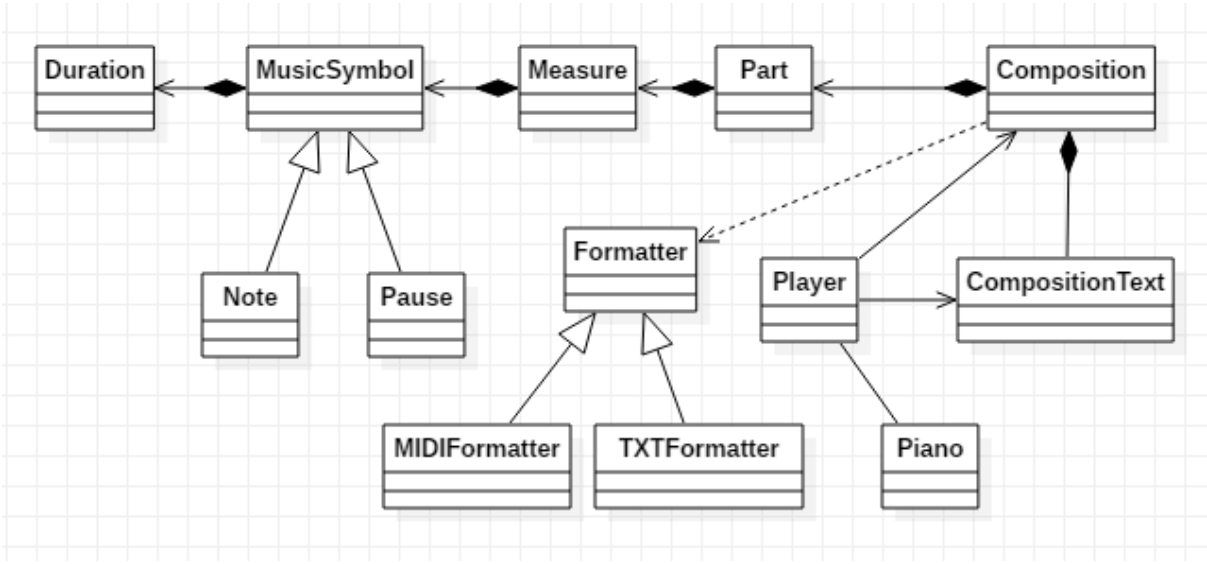
Test primere za testiranje rada realizovanog softverskog rešenja moguće je naći na sajtu: https://virtualpiano.net/. Sajt omogućava sviranje virtuelnog klavira pomoću tastature računara, prema istim pravilima koja važe i u ovom projektu. Za veliki broj kompozicija obezbeđeni su skupovi karaktera koje je potrebno pritiskati, koji ujedno predstavljaju i ulazne fajlove ovog softvera, a moguće ih je pronaći pod opcijom “Music sheets”.

Rezultat rada programa, odnosno korektnost generisanih fajlova je moguće proveriti alatima za reprodukciju.

MIDI fajl može se proveriti pomoću velikog broja audio alata, među kojima je i Windows Media Player, koji dolazi instaliran uz Windows operativni sistem i podržava puštanje midi fajlova.

Dijagram klasa

Na osnovu prethodne funkcionalne specifikacije formiran je sledeći dijagram klasa. Dijagram klasa nije detaljan, te ga treba tumačiti kao skicu koja načelno ukazuje na arhitekturu softvera. Studenti mogu da koriste ovaj dijagram kao referencu i, po potrebi, prošire ga da bi ga usaglasili sa eventualnim dopunama specifikacije.



Prilikom implementacije rešenja, obratiti pažnju na objektno orijentisani dizajn i intenzivno koristiti kolekcije i algoritme standardne biblioteke jezika Java i lambda funkcije gde god je to moguće. Primetiti da postoje dva različita formata izlaznih fajlova.

Specifikacija MIDI formata

Kompletnu specifikaciju MIDI formata fajla moguće je pronaći na sledećoj adresi: http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html

Za potrebe projekta potrebno je iskoristiti Java Sound API :

https://www.oracle.com/technetwork/java/javasounddemo-140014.html

Na navedenom linku moguće je naći kompletnu dokumentaciju, a na narednoj strani dokumenta dat je primer upotrebe ovo biblioteke. U primeru je predstavljena jedna klasa sa glavnom funkcijom, a po pokretanju korisnik ima mogućnost da unosi MIDI brojeve nota koje program potom svira.

```
package muzika;

import javax.sound.midi.MidiChannel;
import javax.sound.midi.MidiSystem;
import javax.sound.midi.MidiUnavailableException;
import javax.sound.midi.Synthesizer;
import java.util.Scanner;

public class MidiPlayer {
    private static final int DEFAULT_INSTRUMENT = 1;
    private MidiChannel channel;

    public MidiPlayer() throws MidiUnavailableException {
        this(DEFAULT_INSTRUMENT);
    }

    public MidiPlayer(int instrument) throws
                                    MidiUnavailableException {
        channel = getChannel(instrument);
    }

    public void play(final int note) {
        channel.noteOn(note, 50);
    }

    public void release(final int note) {
        channel.noteOff(note, 50);
    }

    public void play(final int note, final long length)
                    throws InterruptedException {
        play(note);
        Thread.sleep(length);
        release(note);
    }
    private static MidiChannel getChannel(int instrument)
                    throws MidiUnavailableException {
        Synthesizer synthesizer = MidiSystem.getSynthesizer();
        synthesizer.open();
        return synthesizer.getChannels()[instrument];
    }

    public static void main(String[] args) throws Exception {
        MidiPlayer player = new MidiPlayer();
        Scanner scanner = new Scanner(System.in);
        int note;
        while (!Thread.currentThread().isInterrupted()) {
            System.out.print("Note (1..127) : ");
            note = scanner.nextInt();
            player.play(note, 200);
        }
        scanner.close();
    }
}
```