Objektno-orijentisano programiranje, JUN 1, Grupa 1
Matematički fakultet, školska godina 2019/2020

Napomena: Na Desktop-u napraviti direktorijum pod imenom `oop_Asistent_Prezime_Ime_Indeks` (npr. `oop_NM_Peric_Pera_mi12082`). Pokrenuti *Intellij Idea* i u napravljenom direktorijumu napraviti projekat sa istim nazivom. U napravljenom projektu, paket takođe nazvati tako.
Kod **ne sme** imati sintaksnih grešaka niti izbacivanje `NullPointerException`-a.
Vreme za rad: **3 sata**
Inicijalni asistenata: `Biljana - BS`, `Nemanja - NM`, `Anja - AB`, `Denis - DA`

U tekstu je dat opis klasa, njihovih atributa i metoda. **Dozvoljeno** je (i ohrabrujemo Vas) dodati nove atribute, klase, metode, enume, interfejse u slučaju da Vam olakšavaju implementaciju, i/ili smatrate da Vam poboljšavaju kvalitet koda i slično. Nekada će zahtevi u zadatku i zahtevati od Vas da dodate novi atribut ili slično.

Da bi se uspešno položio ispit potrebno je osvojiti **barem 25 poena**.

1. Napraviti klasu `Rec` koja ima atribut tipa `String` koji čuva informaciju o sadržaju reči. Obezbediti konstruktor koji prihvata String i čuva ga (velika slova pretvoriti u mala). Obezbediti metode:
   - `Rec dodajBrojNaKraj(int x)` - vraća novu reč koja sadrži originalnu reč sa dodatim brojem na kraju
   - `Rec dodajBrojNaPocetak(int x)` - vraća novu reč koja sadrži originalnu reč sa dodatim brojem na početku
   - `String getRec()` - vraća stringovnu reprezentaciju reči
   - `int getDuzina()` - vraća dužinu reči
2. U klasi `Reč` implementirati interfejs `Comparable<Rec>` i implementirati neophodan metod. Dve reči porediti leksikografski po njihovim stringovnim reprezentacijama.
3. Implementirati klasu `Recnik` koja ima atribute:
   - `List<Rec> reci` - lista reči koje rečnik prepoznaje
   - `List<Rec> lozinke` - lista lozinki koje rečnik prepoznaje
   - `Random random` - generator pseudoslučajnih brojeva

   Implementirati metode:
   - `void dodajRec(Rec rec)` - dodaje reč u listu reči
   - `void dodajLozinku(Rec rec)` - dodaje lozinku u listu lozinki
   - `sortiraj()` - sortira leksikografski listu reči i lozinki u rečniku
   - `Rec odaberiNasumicnuLozinku()` - vraća nasumično odabranu lozinku iz liste lozinki
   - `String toString()` - vraća stringovnu reprezentaciju rečnika - prikazati broj reči i lozinki (implementirati format proizvoljno)
4. U klasi `Recnik` implementirati funkciju `static Recnik ucitajRecnik()` tako da iz datoteke `recnik.txt` učitava reči, formira i sortira rečnik (koristiti funkciju `sortiraj()`). Primer datoteke dat je nakon teksta zadatka. Datoteka sadrži indikator `r` ili `l` koji označavaju da li je u pitanju reč ili lozinka. Prilikom parsiranja datoteke, reči i lozinke dodavati koristeći metode `dodajRec(Rec rec)` i `dodajLozinku(Rec rec)`.
5. Implementirati klasu `Okruzenje` koja poseduje atribute `Rec lozinka` i `TextArea log` (prihvatiti ih kroz konstruktor). Implementirati metode:
   - `void prijaviPokusaj(Rec rec)` - U `log` **dodati** poruku `Pokusavam rec "nekaRec"` (gde je `nekaRec` prosleđena reč)
   - `boolean proveriLozinku(Rec rec)` - Prijaviti pokušaj (pozvati `prijaviPokusaj(rec)`) i proveriti da li je `rec` jednaka atributu `lozinka`.
6. Implementirati apstraktnu klasu `Algoritam` koja ima atribut `Okruzenje okruzenje` koji se prosleđuje kroz konstruktor. Klasa poseduje apstraktan metod `abstract Optional<Rec> izvrsi()`. Funkcija će vraćati praznu opcionu vrednost onda kada njen rad ne uspe da pronađe reč koja je lozinka, a inače će vraćati opcionu vrednost koja sadrži reč koja predstavlja pogođenu lozinku.

   Za pravljenje `Optional` objekta možete koristiti `Optional.of(T t)` ili `Optional.empty()` pomoćne funkcije.
7. Implementirati klasu `NasumicnoAlgoritam` koja nasleđuje `Algoritam`. Klasa pored reference na okruženje kroz konstruktor dobija i najveći broj pokušaja koje algoritam može da iskoristi prilikom svog rada. Implementirati metod `Optional<Rec> izvrsi()` tako da generiše nasumične reči čija je dužina jednaka dužini lozinke (dužinu lozinke možete pročitati preko reference na okruženje koju ima svaki algoritam). Izbeći isprobavanje istih lozinki tako što se reči koje algoritam generiše sačuvaju u skupu reči (`Set<Rec>`). Kao pokušaj se računa svaki pokušaj pogađanja lozinke za reč koja nije pre pokušana. Ako se desi da algoritam generiše reč koja je isprobana pre, nastaviće generisanje reči dok ne generiše reč koja pre nije probana.

8. Implementirati klasu `RecnikAlgoritam` koja nasleđuje `Algoritam`. Klasa pored reference na okruženje kroz konstruktor dobija i referencu na rečnik koji će koristiti prilikom svoga rada. Implementirati metod `izvrsi()` tako da pogađa lozinku pokušavajući sve reči dostupne u rečniku.

9. Implementirati klasu `RecnikSaBrojevimaAlgoritam` koja nasleđuje `RecnikAlgoritam`. Klasa pored reference na okruženje kroz konstruktor dobija i referencu na rečnik koji će koristiti prilikom svoga rada. Implementirati metod `izvrsi()` tako da pogađa lozinku pokušavajući da kombinuje reči dostupne u rečniku sa ciframa koje dodaje na početak i kraj. Dakle za jednu reč iz rečnika (na primer `biblioteka`), algoritam će probati 10 varijanti te reči sa ciframa na kraju (`biblioteka0`, `biblioteka1`, ..., `biblioteka9`) i 10 varijanti te reči sa cifarama na početku (`0biblioteka`, `1biblioteka`, ..., `9biblioteka`).

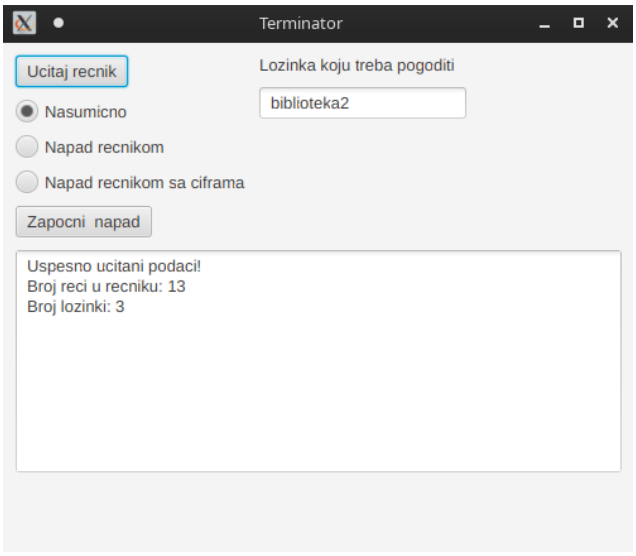
10. Implementirati klasu `Main` koja nasleđuje klasu `Application` biblioteke `javafx` i izgleda kao na slikama. Obezbediti da je uvek odabrano tačno jedno radio dugme kao i da je podrazumevano odabrano radio dugme koje ima vrednost Nasumicno.

11. Kada korisnik klikne dugme `Ucitaj recnik`, potrebno je učitati rečnik iz datoteke `recnik.txt` i u veliko tekstualno polje ispisati informacije u rečniku koji je učitan (slika 1). Pored toga, potrebno je odabrani nasumično jednu lozinku iz skupa učitanih lozinki iz datoteke i postaviti je u tekstualno polje sa gornje strane na korisničkom interfejsu (slika 1, reč `biblioteka2`).

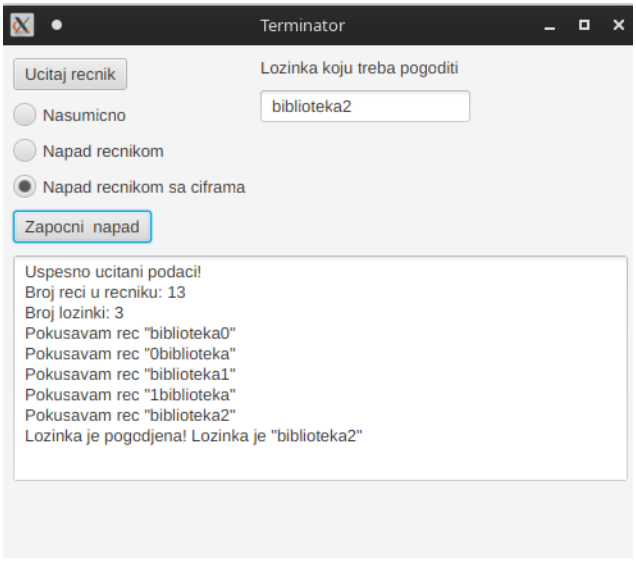
    Klikom na dugme `Zapocni napad`, u zavisnosti od odabranog radio dugmeta, konstruiše se potrebni algoritam i pokušava se pogađanje lozinke. U veliko tekstualno polje potrebno je prijavljivati reči koje algoritam pokušava kao i ishod rada algoritma, odnosno da li je lozinka pogođena ili nije. Na slici 2 je dat primer u kojem je pokušan napad koristeći rečnik sa kombinacijom cifara.

    Primer sadržaja datoteke:

```
r,IGRA
r, Racunar
r,telefon
r,jezik
r, ProgRam
l,igra
r,Programiranje
r, biblioteka
l,biblioteka2
r,Tastatura
r, Mis
r,Stampac
r,Sto
l,igra1
r,Stolica
r,Lampa
```



Slika 1: Učitavanje rečnik i odabir lozinke



Slika 2: Pokušaj napada rečnikom sa brojevima