Dežurni nastavnik: _____

Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Testiranje softvera (SI3TS / 13S113TS)
Nastavnik: Prof. dr Dragan Bojić
Asistent: Doc. dr Dražen Drašković
Ispitni rok: Januar 2019.
Datum: 18.01.2019.

Kandidat:* _____

Broj indeksa:* _____

Ispit traje 2.5 sata, a u toku prvog sata nije dozvoljeno napuštanje ispita.
Upotreba literature nije dozvoljena.

Zadatak 1	_____/10	*Zadatak 4*	_____/12
Zadatak 2	_____/10	*Zadatak 5*	_____/10
Zadatak 3	_____/8	*Zadatak 6*	_____/10

Ukupno na ispitu: _____**/60** ***Ukupno na domaćem*:*** _____**/40**

Rok u kome je odbranjen domaći*: _____ (primer: januar 2019)

Ukupno: _____/100

Ocena: _____ (_____)

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**. * popunjava student.

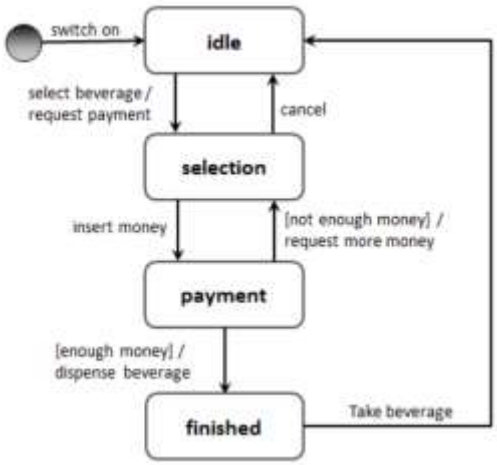
1. [10] Funkcija “Štampanje poruke” čita dva znaka i, u zavisnosti od njihovih vrednosti, preduzima specificirane akcije.
   - Prvi znak mora biti “A” ili “B”.
   - Drugi znak mora biti cifra.
   - Ako je prvi znak "A" ili "B", a drugi znak cifra, datoteka mora biti ažurirana.
   - Ako je prvi znak netačan (nije "A" ili "B"), mora se ispisati poruka X.
   - Ako je drugi znak netačan (nije cifra), mora se ispisati poruka Y.

a) Nacrtati uzročno-posledični graf sa odgovarajućim ograničenjima uzroka. Uzeti da su uzroci definisani kao: C1 - Prvi znak je A, C2 - Prvi znak je B, C3 - Drugi znak je cifra. Posledice su definisane kao: E1 - Ažuriranje datoteke, E2 - Štampanje poruke X, E3 - Ispis poruke Y

b) Koliki bi bio maksimalan broj test slučajeva na osnovu dobijenog grafa?

c) Odrediti test primere metodom senzitizacije putanja.

2. [10] Dati dijagram stanja pokazuje ponašanje automata za napitke (engl. beverage). Modelom je opisana interakcija između korisnika i automata.



a) Navesti skup stanja, skup ulaznih događaja, skup akcija.

b) Navesti minimalan broj test scenarija da se pokriju svi prelazi automata (*0-switch cover*).

c) Kombinovanjem stanja i ulaznih događaja, odrediti sve nedefinisane/nelegalne kombinacije. Za svaku kombinaciju predložiti akciju (može biti promena stanja, neka od postojećih akcija, ili - što znači ignorisanje kombinacije).

d) Navesti (ako postoji) još neku interakciju između korisnika i automata koja bi u stvarnosti trebalo da bude moguća, a koja nije pokrivena dopunjenim modelom iz tačke c).

3. [8] Dat je izvod iz specifikacije zahteva za bibliotečki sistem:

Bibliotekari mogu:

1. Registrovati nove članove.
2. Vratiti knjige od člana (u smislu evidentiranja u sistemu).
3. Naplatiti kaznu članu.
4. Dodati nove knjige u sistem sa njihovim ISBN, autorom i naslovom.
5. Ukloniti knjige iz sistema.
6. Dobiti odziv sistema u roku od 5 sekundi.

Članovi mogu:

7. Pozajmiti najviše 3 knjige odjednom.
8. Pogledati istoriju knjiga koje su pozajmili / rezervisali.
9. Biti novčano kažnjeni zbog toga što nisu vratili knjigu u roku od 3 nedelje.
10. Dobiti odziv sistema u roku od 3 sekunde.
11. Pozajmiti knjigu bez naknade na maksimalno 4 nedelje.
12. Rezervisati knjige (ako su na pozajmici kod drugog člana).

Svi korisnici (bibliotekari i članovi):

13. Mogu da traže knjige po ISBN, autoru ili naslovu.
14. Mogu pregledati sistemski katalog.
15. Sistem će odgovoriti na zahteve korisnika u roku od 3 sekunde.
16. Korisnički interfejs će biti jednostavan za korišćenje.

Nabrojati sve nedoslednosti (sukobljenost) između parova pojedinačnih zahteva.

4. [12] Data je programski kod za sortiranje niza celih brojeva. Glavni program napisan u Javi treba da zada ulazni niz celih brojeva, od strane korisnika i da pozove traženu metodu:

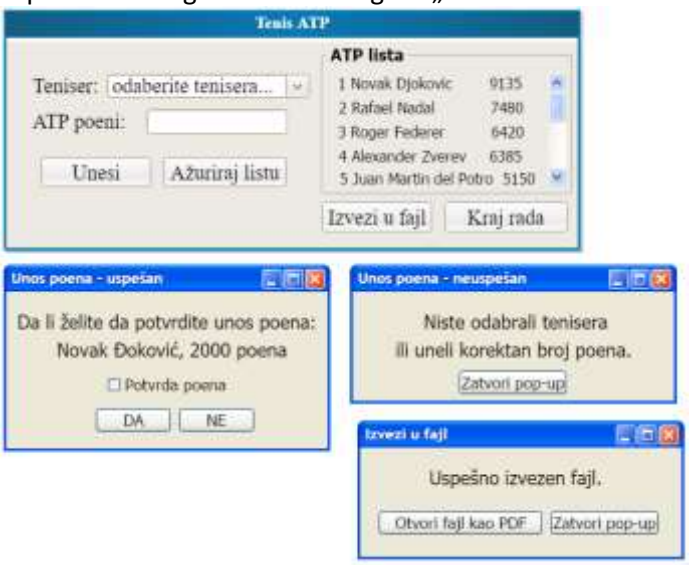
```
1  public static void sort(int arr[]) {
2       int n = arr.length;
3       for (int i = 1; i < n; i++) {
4               int key = arr[i];
5               int j = i-1;
6               while ( (j > -1) && ( arr [j] > key ) ) {
7                       arr [j+1] = arr [j];
8                       j--;
9               }
10              arr[j+1] = key;
11              printArray(arr); //metod za prikaz niza
12      }
13 }
```

a) [4] Napisati test primere koji uspešno testiraju petlje u metodi *sort(int[])*. Za svaki test kao ulazni podatak potrebno je definisati dužinu niza celih brojeva i sve elemente niza. Za svaki test napisati i šta pokriva ili ne pokriva u programskom kodu.

b) [4] Napisati *JUnit* testove koji bi testirali izvršavanje metoda *sort* i *printArray*. Pretpostaviti da metoda *printArray* stavlja sve elemente u jedan string, a zatim štampa na standardnom izlazu. Ukoliko je neophodno možete implementirati još neku dodatnu metodu za potrebe testiranja.

c) [4] Ukoliko bi relacioni operatori u liniji 6 bili izmenjeni mutacionim operatorom ROR (zamenjuje relacioni operator > drugim relacionim operatorom), napisati pomoću mutacionog skora kako bi to uticalo na testiranje. Šta bi programer morao da izmeni/doda u programskom kodu, tako da je obezbeđena obrada mogućih grešaka prilikom izvršavanja ovog programa (*runtime*), uključujući i mutirane programe sa izmenama u liniji 6.

5. [10] Neka je data aplikacija za unos poena tenisera nakon završenog turnira i ažuriranje ATP liste. Aplikacija ima glavni prozor i nekoliko iskačućih (pop-up) prozora. Glavni ekran ima padajuću listu za izbor tenisera, numeričko polje za unos ATP poena i 4 dugmeta. Prvo dugme „Unesi“ služi za unos poena odabranom teniseru. Unos poena može biti uspešan i neuspešan. Za uspešan unos, potrebno je odabrati polje „Potvrda poena“ i dugme DA. Ukoliko nije odabran teniser ili nisu uneti ATP poeni, iskače poruka da je neuspešan unos. Drugo dugme „Ažuriraj listu“ ažurira poene, sortira listu i prikazuje je u prozoru. Inicijalno, po pokretanju aplikacije, ATP lista ima trenutne poene i prikazanu neku listu (kao na slici). Treće dugme „Izvezi u fajl“ pravi PDF fajl od prikazane ATP liste. Četvrto dugme „Kraj rada“ zatvara glavni prozor aplikacije.



Definisati sve događaje opisane u ovoj specifikaciji oznakama E1, E2, ... Ex, a zatim testirati ovaj program koristeći EFG (*Event Flow Graph*) graf i napisati sekvence događaja dobijene iz EFG. Smatrati da kada iz glavnog prozora korisnik aplikacije pređe u iskačući prozor, on nema fokus nad glavnim prozorom, već samo nad prozorom koji se otvorio. Otvaranje izvezenog fajla smatrati događajem koji se otvara u okviru iste aplikacije, kao PDF fajl, i čijim se zatvaranjem fokus vraća na glavni prozor.

6. [10] Dat je sledeći programski kod u programskom jeziku Java:

```
package factorial;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class Factorial extends JFrame{
     private JTextField txtNum;
     private JLabel lblNum, lblRes;
     private JButton btnCompute;

     public static int ComputeFactorial(int number){
         int n = number-1;
         do{
             number = number*n;
             n--;
         }while(n>=1);
         return number;
     }
     public static void createlog(String tekst){
     ... //metoda koja loguje poruke u txt fajl
     }
     public Factorial(){
         super("GUI Factorial");
         Container c = getContentPane();
         c.setLayout(new FlowLayout());
         lblNum = new JLabel("Enter an integer: ");
         txtNum = new JTextField(10);
         lblRes = new JLabel();
         btnCompute = new JButton("Compute");
//nastavak koda u desnoj koloni
```

```
btnCompute.addActionListener( new ActionListener(){
     public void actionPerformed(ActionEvent e){
               String str = txtNum.getText();
               int tmp = Integer.parseInt(str);
               tmp = ComputeFactorial(tmp);
               String str2="The factorial of " +
                              str + " is "+tmp;
               lblRes.setText(str2);
               createlog(str2);
              }
         });
         c.add(lblNum);
         c.add(txtNum);
         c.add(btnCompute);
         c.add(lblRes);
         setSize(200,150);
         show();
     }
     public static void main(String args[]){
         Factorial app = new Factorial();
         app.setResizable(false);
         app.setLocation(400,200);
         app.addWindowListener(
         new WindowAdapter(){
           public void windowClosing(WindowEvent e){
                 System.exit(0);
            }
         });
     }
}
```

a) Da li ovaj programski kod može da se testira kao: (*na crti pored napisati kako bi se testiralo ako je moguće ili razlog zašto ne bi moglo da se testira, ako nije moguće*)

intra-metod testiranje _____

_____

intra-klasno testiranje _____

_____

inter-klasno testiranje _____

_____

b) Skicirati graf kontrole toka koji reprezentuje gorenavedeni objekno orijentisani kod. Metode realizovati kao podgrafove.

c) Koje druge strategije testiranja je moguće izvesti nad ovim kodom (prvenstveno iz crne kutije, bele kutije, GUI testiranja), i za svaku moguću strategiju i vrstu testiranja napisati šta bi bio cilj tog testiranja i koje bagove/defekte bismo mogli da pronađemo u ovoj aplikaciji. Nije potrebno za svaku strategiju izvršiti detaljno testiranje, već kod svake vrste prikazati po primer kojim bi se defekat otkrio.