Dežurni nastavnik: _____

Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku
Studijski program „Softversko inženjerstvo“

Predmet: Testiranje softvera (13S113TS)
Nastavnici: prof. dr Dragan Bojić, prof. dr Dražen Drašković
Saradnik: mast. inž. Nikola Stanković
Ispitni rok: Januar 2026.
Datum: 18.02.2026.

Kandidat[*]*:* _____

Broj indeksa[*]*:* _____

Rokovi u kojima branite domaće zadatke[*]*:* DZ1 _____ , DZ2 _____

„Danas polažete dva ispita: iz struke i iz poštenja. Ako treba jedan da padnete, neka to bude iz struke, jer ćete kao pošteni ljudi, lako savladati struku, a ako padnete na poštenju, nećete ga savladati nikad.“

Ispit traje 150 minuta. U toku prvih 60 minuta nije dozvoljeno napuštanje ispita.
Upotreba literature nije dozvoljena.

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

Ukupno na ispitu: _____ / 60

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. Ukoliko drugačije nije definisano, kod pitanja koja imaju ponuđene odgovore treba zaokružiti samo jedan odgovor. Na ostala pitanja odgovarati **čitko i precizno**.

[*] popunjava student

1. [12] Razvija se softverski modul za automatski terminal (kiosk) u SPA centru "Oaza" koji služi za rezervaciju termina personalizovanog tretmana i autorizaciju korišćenja te usluge. Korisnik na terminalu prinese svoju člansku karticu (RFID čitač), a zatim putem ekrana unosi sledeće podatke:

1. Trajanje personalizovanog tretmana (u minutima):
   - Minimalno trajanje tretmana je 45 minuta, a uneta vrednost mora biti deljiva sa 15.
   - Maksimalno trajanje jednog tretmana je 180 min.
   - Maksimalno dozvoljeno ukupno vreme po korisniku u toku jednog dana je 360 min.
2. Sigurnosni pristupni kod za ormarić (prvi stepen autorizacije):
   - Mora imati tačno 3 ili 5 cifara.
   - Dozvoljeni su samo numerički karakteri (0–9).
   - Kod ne sme sadržati niz uzastopnih istih cifara (npr. 222 ili 55555).
   - Kod ne sme početi cifrom 9.
3. Datum početka korišćenja paketa (drugi stepen potvrde):
   - Format je strogo definisan kao **dd.mm.yyyy** (npr. 08.03.2026), jer korisnik na ekranu ima samo numeričku tastaturu (tačke se same umeću pri unosu brojeva).
   - Datum mora biti validan (npr. 31.06.2026 nije validan).
   - Godina mora biti u rasponu od 2026. do 2028. (period trajanja licence centra), ali sistem mora osigurati da datum nije u prošlosti u odnosu na trenutni datum (smatrati da je danas 18.02.2026).

a) [3] Definisati sve klase ekvivalencije (ispisati i označiti i legalne i nelegalne sa LK i NK)

ID klase (npr. LK1, NK1)	Tip polja (npr. Trajanje, Kod, Datum)	Opis

b) [3] Identifikovati <u>granične slučajeve</u> (ispisati sve granice koje ste primetili u opisu).

Tip polja (npr. Trajanje, Kod Datum)	Granična vrednost (npr. 998)	Opis (npr. Provera donje granice trajanja tretmana, ili dužina/broj karaktera u formatu pristupnog koda, i slično)

c) [3] Pripremiti <u>test primere</u> koji pokrivaju sve klase, a za legalne koristiti tehniku minimalnog skupa test primera.

ID testa	Trajanje	Pristupni kod	Datum	Izlazna poruka (uspeh/neuspeh)	Pokrivene klase
TP1					
TP2					
TP3					
TP4					

<u>Napomena</u>: * - broj redova datih u tabelama ne prejudicira tačan broj potrebnih odgovora

d) [3] Kako biste razrešili zavisnosti u tabeli sa klasama ekvivalencije, ako bi dodatni uslovi bili sledeći: *Datum rezervacije mora biti najmanje 48 sati unapred u odnosu na trenutak unosa, ali ne više od 30 dana od današnjeg datuma. Ako je izabrani datum nedelja, sistem dozvoljava samo termine duže od 90 minuta.* Ukoliko ovo nije moguće rešiti definisanom tabelom u tački a), predložiti kako bi se korelisali ishodi u odnosu na ove različite dodatne uslove. Dati predlog rešenja.

2. [8] Odgovoriti zaokruživanjem tačnih odgovora na sledeća pitanja (pitanje ima najmanje jedan tačan odgovor, ali može imati i više tačnih odgovora):

P1) Koja je osnovna svrha *JUnit* testova u razvojnom procesu?

a) Testiranje celokupnog sistema od strane krajnjih korisnika.

b) Automatizacija testiranja najmanjih delova koda (metoda i klasa).

c) Provera da li softver radi ispravno na različitim operativnim sistemima.

P2) Koji su glavni ciljevi integracionog testiranja?

a) Provera interakcije i komunikacije između povezanih modula/komponenti.

b) Testiranje algoritamske ispravnosti unutar jedne izolovane funkcije.

c) Pronalaženje grešaka u definisanim interfejsima (API-jima) između podsistema.

P3) Šta se primarno testira tokom *GUI* testiranja (testiranje grafičkog korisničkog interfejsa)?

a) Odziv sistema na pritisak tastera i vizuelni izgled elemenata.

b) Validnost podataka smeštenih u relacionu bazu podataka.

c) Redosled tabulatora (*tab order*) i dostupnost elemenata (*accessibility*).

d) Efikasnost algoritama za sortiranje podataka unutar back-end servisa.

P4) U kontekstu objektno orijentisanog (OO) testiranja, šta predstavlja dodatni suštinski izazov u odnosu na proceduralno testiranje?

a) Polimorfizam, jer ista poruka može pozvati različite metode u zavisnosti od tipa objekta.

b) Nemogućnost pristupa privatnim (*private*) atributima klase iz *JUnit* testova koji se nalaze u drugom paketu.

c) Nasleđivanje, jer nasleđene metode treba ponovo testirati u novom kontekstu potklase.

d) Korišćenje if-else naredbi unutar metoda klase.

P5) Šta je tačno za kombinatorno testiranje (npr. *All-Pairs testing*)?

a) Ono značajno smanjuje broj test slučajeva u poređenju sa testiranjem svih mogućih kombinacija (*brute force*).

b) Zasnovano je na zapažanju da većinu grešaka uzrokuje interakcija najviše dva ili tri parametra.

c) Garantuje da će svaka moguća vrednost svakog parametra biti testirana bar jednom u kombinaciji sa svakom vrednošću svakog drugog parametra.

P6) Koja je razlika između sistemskog testiranja i testiranja prihvatanja (*Acceptance Testing*)?

a) Sistemsko testiranje fokusira se na tehničke zahteve i integritet celog sistema, dok se prihvatanje fokusira na poslovne potrebe korisnika.

b) Sistemsko testiranje vrši razvojni tim (ili nezavisni QA), dok testiranje prihvatanja obično vrše klijenti ili krajnji korisnici.

c) Sistemsko testiranje je uvek testiranje belom kutijom, dok je testiranje prihvatanja isključivo tehnika crne kutije.

3. [8] Data je Java funkcija koja transponuje kvadratnu matricu. Ukoliko je prosleđena matrica *null* ili dimenzije matrice nisu korektne, funkcija ispisuje poruku o grešci, a u suprotnom transponuje originalnu matricu, bez kreiranja nove matrice.

```
    public static void transpose(int[][] matrix, int n) {
1.    if (n > 0 && matrix != null) {
2.      for (int i = 0; i < n; i++) {
3.          for (int j = i; j < n; j++) {
4.              if (i == j)
5.                  continue;
6.              int t = matrix[j][i];
7.              matrix[j][i] = matrix[i][j];
8.              matrix[i][j] = t;
9.          }
10.       }
11.   }
12.   else {
13.     System.out.println("Nevalidni podaci!");
14.   }
15. }
```

a) **[2]** Nacrtati graf toka kontrole za datu funkciju.

b) **[2]** Navesti formulu i odrediti broj ciklomatske kompleksnosti date funkcije.

c) **[4]** Testirati sve petlje u datoj funkciji koristeći Bajzerovu heurističku tehniku.

Rešenje:

4. [10] Date su klase *Repository* i *Service*, kao i test klasa *ServiceTest* sa jednim definisanim testom.

```
class Repository {
    int get(int id){ return id; }
}

class Service {
    private final Repository repo;

    Service(Repository repo) {
        this.repo = repo;
    }

    int compute(int id) {
        int v = repo.get(id);
        if (v < 0) return -1;
        if (v == 0) return 0;
        return v * 2;
    }
}
```

```
class ServiceTest {
   @Test
   void TP_01() {
       Repository repo =
       mock(Repository.class);
       when(repo.get(-1)).thenReturn(-1);

       Service s = new Service(repo);
       assertEquals(-1, s.compute(-1));
       verify(repo).get(-1);
   }
}
```

a) **[4]** Ukratko objasniti šta je *mock* objekat i koja je njegova svrha u datom testu.

b) **[3]** Čemu služi poziv metode *verify* u datom testu kojim se ispituje da li je metoda *get* pozvana sa argumentom -1? Zašto nije dovoljno da jedina verifikaciona tačka testa bude poziv metode *assertEquals*?

c) **[3]** Odrediti procentualnu pokrivenost **odluka** u klasi *Service* koja se postiže izvršavanjem implementiranog testa u klasi *ServiceTest*. Ako ta pokrivenost nije maksimalna, implementirati dodatne testove kojima se postiže 100% pokrivenost odluka ove klase.

5. **[12]** Data je hijerarhija klasa, pri čemu važi sledeće: ***save***() baca *SaveException* ako je sadržaj prazan; ***append***() postavlja *modified = true*; ***save***() postavlja *modified = false* ako uspe; ***MarkdownDocument*** redefiniše ***wordCount***() da ne broji specijalne znake kao posebne reči.

```
abstract class Document {
    protected String content;
    protected boolean modified;
    public Document(String content) { ... }
    public Document() { … }
    public abstract int wordCount();
    public void append(String text) { ... }
    public void save() throws IllegalStateException { ... }
    public boolean isModified() { ... }
}
class TextDocument extends Document {
    public int wordCount() { ... }   // reči odvojene blanko znakom
}
class MarkdownDocument extends TextDocument {
    @Override
    public int wordCount() { ... }   // ignoriše oznake #, *, _
}
class ReadOnlyDocument extends Document {
    @Override
    public void append(String text) {
        throw new UnsupportedOperationException();
    }
    @Override
    public int wordCount() { ... }
}
```

a) **[3]** Konstruisati konačni automat za klasu *Document* na bazi stanja promenljivih i izuzetka (da li se desio).

b) **[3]** Definisati skup testova koji pokriva sva stanja i prelaze.

c) **[3]** Implementirati testove iz tačke b), bez korišćenja posebnih biblioteka. Obratiti pažnju na to da je klasa *Document* apstraktna.

d) **[3]** Nacrtati jedinstveni *Yo-Yo* graf za sledeće scenarije:

- Klijent (metod *main* klase *Test*) kreira prazan tekst dokument, dodajte tekst, ispituje broj reči i snima dokument.
- Klijent (metod *main* klase *Test*) kreira prazan *markdown* dokument, dodaje tekst, ispituje broj reči i snima dokument.

Rešenje:

Dodatan prostor za rad:

6. **[10]** Jedna *GUI* aplikacija se sastoji od tekstualnog polja za unos teksta i dugmadi ***New***, ***Open***, ***Save***, ***Append*** i ***Close***. Za ovaj *GUI* su definisana sledeća stanja:

- S_0 – Nema otvorenog dokumenta.
- S_1 – Dokument je otvoren, ali nije izmenjen.
- S_2 – Dokument je izmenjen.
- S_3 – Greška pri čuvanju dokumenta.

Dat je podskup pravila ponašanja aplikacije. Potrebno je samostalno dopuniti ovaj skup dodatnim pravilima koja nedostaju:

- Klik na ***New*** uzrokuje prelazak u stanje S_1.
- Klik na ***Append*** uzrokuje prelazak u stanje S_2.
- Klik na ***Save*** uzrokuje prelazak u stanje S_3, ako je sadržaj prazan, a u suprotnom u S_1.
- Klik na ***Close*** uzrokuje pojavljivanje dijaloga potvrde ukoliko je *modified = true.*
- Zatvaranje aplikacije uzrokuje prelazak u stanje S_0.

a) **[3]** Konstruisati *FSM* (konačni automat) datog *GUI* sistema.

b) **[3]** Konstruisati *EFG* (graf toka događaja) datog *GUI* sistema.

c) **[2]** Na posebnoj slici dopuniti *EFG* iz prethodne tačke nelegalnim interakcijama (*FIP*). Ukratko navesti pravila za dodavanje nelegalnih interakcija.

d) **[2]** Konstruisati test sekvence za pokrivanje svih grana grafa iz tačke c).

<u>Rešenje:</u>

Dodatan prostor za rad: