

Zadatak 1

Na sistemu za rezervaciju termina kod lekara korisnik unosi:

- Ime i prezime – samo slova i razmak, dužina od 3 do 30 karaktera
- Broj zdravstvene knjižice – tačno 11 cifara
- Broj dana od današnjeg dana (željeni datum pregleda)
 - Datum pregleda mora biti nakon današnjeg datuma
 - Dozvoljeno je zakazivanje najviše 30 dana unapred
- Tip pregleda – Opšti, Specijalistički

U zavisnosti od unosa korisnika, rezultat je jedan mogućih poruka:

- Neispravno ime i prezime
- Broj knjižice nije validan
- Termin uspešno zakazan za <Opšti | Specijalistički> pregled

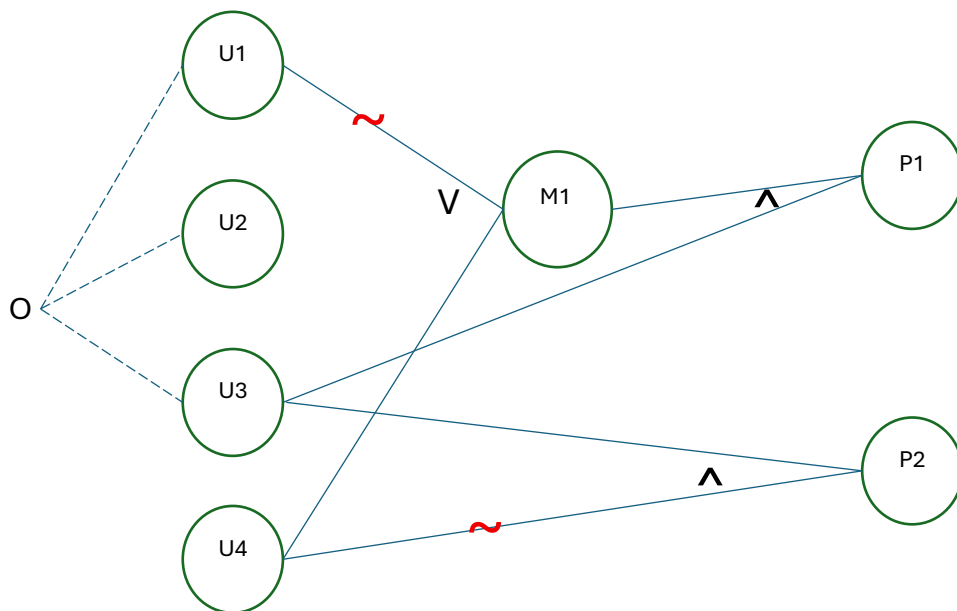
Za dati opis sistema za rezervaciju definisati:

- Klase ekvivalencije (ulazne i izlazne),
- granične slučajeve,
- tabelu test primera.

Na osnovu dobijene tabele test primera implementirati testove u biblioteci *Cocubmer* koristeći *Given, When, Then* patern.

Zadatak 2

Za uzročno-posledični graf sa slike tehnikom **senzitivizacije putanja** odrediti minimalni skup kombinacija test primera.



Zadatak 3

Testiranje bele kutije (pokrivenost odluka / grana). Data je metoda `hasPassedExam` u klasi `ExamResultService`. Metoda vraća `true` ukoliko je student položio ispit, u zavisnosti od: broja poena ostvarenih na ispitu, broja poena sa laboratorijskih vežbi, informacije da li je detektovano prepisivanje. U suprotnom, metoda vraća `false`.

Potrebno je:

1. Skicirati dijagram toka (*control flow diagram*) za datu metodu
2. Identifikovati sve odluke u kodu
3. Kreirati tabelu test primera koja obezbeđuje 100% pokrivenost odluka
4. Za svaki test primer: navesti ulazne vrednosti, navesti očekivani rezultat, objasniti koje odluke i grane su pokrivene

Kod klase `ExamResultService`

```
public class ExamResultService {  
    public boolean hasPassedExam(int examPoints, int labPoints, boolean cheatingDetected) {  
        if (examPoints < 0 || examPoints > 100 || labPoints < 0 || labPoints > 30) {  
            return false;  
        }  
  
        if (cheatingDetected) {  
            return false;  
        }  
  
        if (examPoints >= 50) {  
            if (examPoints + labPoints >= 60) {  
                return true;  
            } else {  
                return false;  
            }  
        } else {  
            return false;  
        }  
    }  
}
```

Zadatak 4

Klasa `OrderService` zadužena je za obradu porudžbina.

Metoda `placeOrder` služi za obradu porudžbine tako da prvo proverava dostupnost proizvoda u skladištu, zatim obrađuje plaćanje i na kraju čuva porudžbinu u bazi samo ako su svi prethodni koraci uspešni. Prilikom testiranja ove metode potrebno je proveriti različite scenarije:

- Uspešna porudžbina
- Neuspešna porudžbina usled nedovoljno zaliha

Metoda `applyDiscountAndPlaceOrder` dodaje složenost jer prvo izračunava popust na osnovu količine proizvoda i ukupne vrednosti porudžbine, a zatim izvršava istu logiku kao `placeOrder`. Testiranje ove metode uključuje scenarije:

- Neuspešna porudžbina zbog plaćanja
- Uspešna porudžbina uz popust od 15%

Dijagram klasa dat je u nastavku.

