Testiranje strategijama bele kutije

WHITE BOX TESTING

Testiranje belom kutijom

- Testiranje strategijama bele kutije (eng. *white box testing*)
- U literaturi još i kao: strukturno testiranje (eng. *structural testing*), testiranje čistom kutijom (eng. *clear box*), otvorenom kutijom (eng. *open box*), staklenom/transparentnom kutijom (eng. *glass box* / *transparent box*) ili zasnovano na kodu (eng. *code-based testing*)
- Primarni izvor za projektovanje testova je izvorni kod sa fokusom na tok kontrole i tok podataka
- Cilj strukturnog testiranja nije da se izvrše sve moguće funkcije programa,
  već da se izvrše/aktiviraju različite programske i strukture podataka u programu
- Grafovi kontrole toka predstavljaju vizuelnu reprezentaciju strukture koda nekog programa.
  Izvršavanjem programa (na primer, za neke test podatke) vrši se izbor neke putanje u grafu.

Tehnike zasnovane na kontroli toka

- Pokrivenost iskaza (eng. *Statement Coverage*) = pokrivenost instrukcija ili pokrivenost koda
- Pokrivenost odluka ili pokrivenost grana (eng. *Decision or Branch Coverage*)
- Pokrivenost uslova (eng. *Condition Coverage*)
- Pokrivenost višestrukih uslova (eng. *Multiple Condition Coverage*)
- Minimalna pokrivenost višestrukih uslova (eng. *Minimal Multiple Condition Coverage*)
- Pokrivenost odluka i uslova (eng. *Decision/Condition Coverage*)
- Modifikovana pokrivenost odluka i uslova (*MC/DC*)

Zadatak 1 - Pokrivenost iskaza

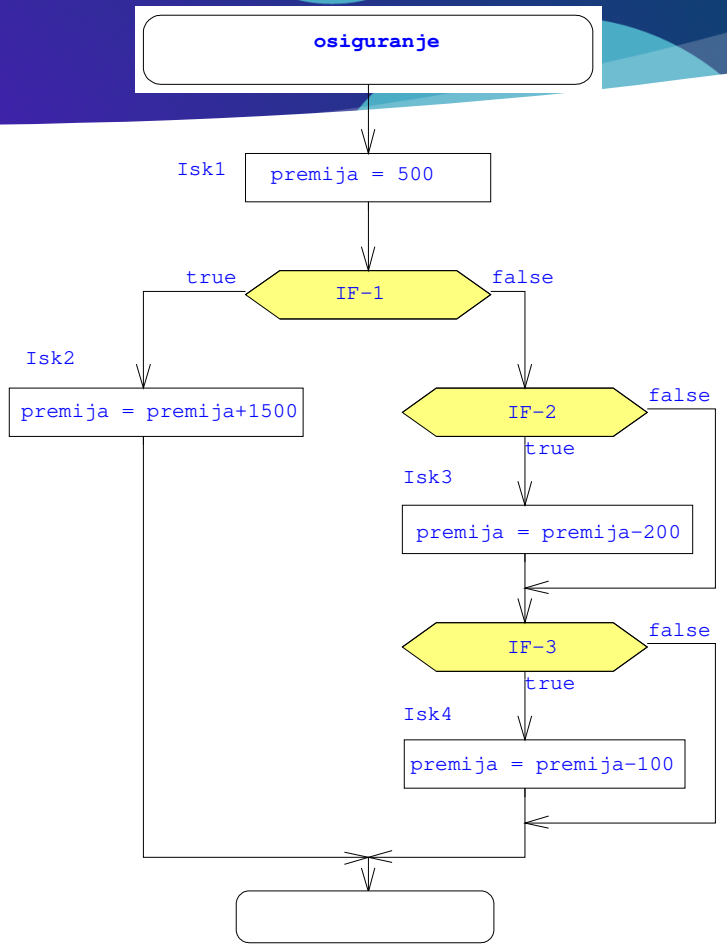
- Za datu funkciju (metodu) realizovanu u programskom jeziku *Java*, odrediti skup test primera koji pokriva sve iskaze u programu.

```
public static void osiguranje(int godine, char pol,
                               boolean brakStatus){
    int premija = 500;
    if ((godine < 25) && (pol == 'm') && !brakStatus){
        premija=premija + 1500;
    } else {
        if (brakStatus || (pol == 'z'))
            premija = premija - 200;
        if ((godine > 45) && (godine < 65))
            premija = premija - 100;
    }
    System.out.println(premija);
}
```

Zadatak 1 - Pokrivenost iskaza - Rešenje

- Prvo ćemo za datu metodu iz postavke zadatka da nacrtamo dijagram toka
- Na osnovu prikazanog dijagrama toka kreira se odgovarajuća tabela u kojoj ćemo prikazati pokrivene iskaze, kao i test primere koji su realizovani:

Pokriveni iskazi	Godine	Pol	Bračni status	Test primer
Isk1, IF-1, Isk2	< 25	m	False	(1) 15 m False
Isk1, IF-1, IF-2, Isk3, IF-3, Isk4	> 45, < 65	z	True	(2) 50 z True



Zadatak 2 - Pokrivenost odluka

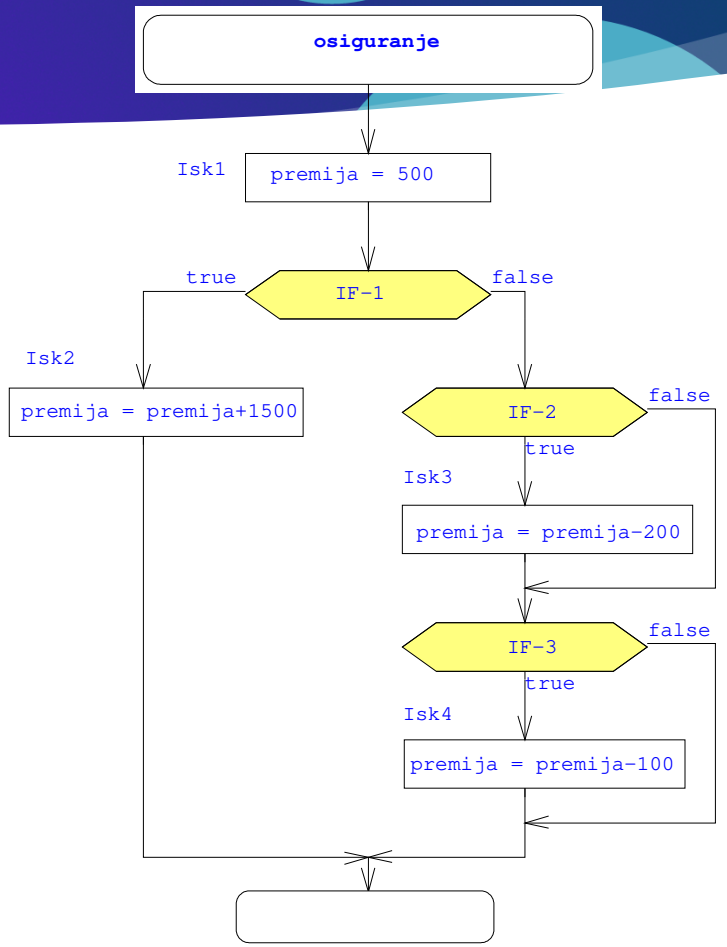
- Za datu funkciju (metodu) realizovanu u programskom jeziku *Java*, odrediti skup test primera koji pokriva sve odluke u programu.

```
public static void osiguranje(int godine, char pol,
                              boolean brakStatus){
    int premija = 500;
    if ((godine < 25) && (pol == 'm') && !brakStatus){
        premija=premija + 1500;
    } else {
        if (brakStatus || (pol == 'z'))
            premija = premija - 200;
        if ((godine > 45) && (godine < 65))
            premija = premija - 100;
    }
    System.out.println(premija);
}
```

Zadatak 2 - Pokrivenost odluka - Rešenje

- Kao i u prethodnom, prvo nacrtamo dijagram toka.
- U tabeli, vrste (redovi) predstavljaju odluke koje treba pokriti testovima. Kolone prikazuju svaki od uslova koji treba da budu ispunjeni da bi se pokrila odgovarajuća odluka, a poslednja kolona definiše test primer.

Pokrivene odluke	Godine	Pol	Bračni status	Test primer
IF-1 True	< 25	m	False	(1) 23 m False
IF-1 False	< 25	z	False	(2) 23 z False
IF-2 True	*	z	*	(2)
IF-2 False	≥ 25	m	False	(3) 50 m False
IF-3 False	≤ 45	z #	*	(2)
IF-3 True	> 45, < 65	*	*	(3)



Zadatak 3 - Pokrivenost uslova

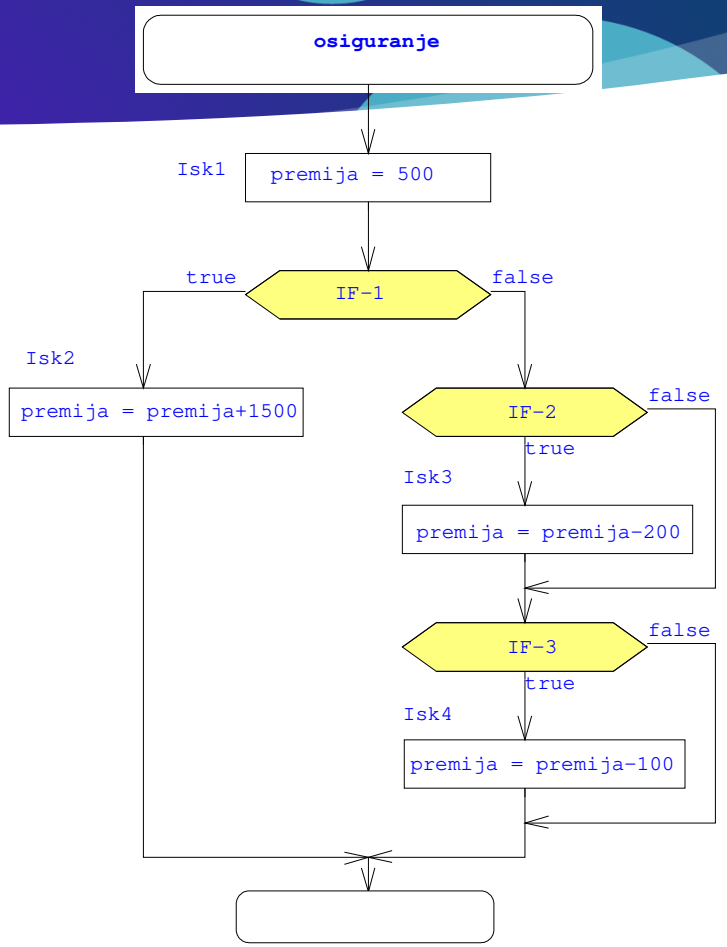
- Za istu funkciju (metodu) realizovanu u programskom jeziku *Java*, iz zadataka 1 i 2, odrediti skup test primera koji pokriva sve uslove.

```
public static void osiguranje(int godine, char pol,
                               boolean brakStatus){
    int premija = 500;
    if ((godine < 25) && (pol == 'm') && !brakStatus){
        premija=premija + 1500;
    } else {
        if (brakStatus || (pol == 'z'))
            premija = premija - 200;
        if ((godine > 45) && (godine < 65))
            premija = premija - 100;
    }
    System.out.println(premija);
}
```

Zadatak 3 - Pokrivenost uslova - Rešenje

- U tabeli, vrste (redovi) predstavljaju elementarne uslove koje treba pokriti (npr. prvi red definiše u prvoj *if* kontroli toka, ispunjenost svih elementarnih uslova na jednu vrednost, a drugi red definiše u prvoj *if* kontroli toka, ispunjenost negacija tih elementarnih uslova). Poslednja kolona je test primer.

Pokriveni uslovi	Godine	Pol	Bračni status	Test primer
IF-1	< 25	z	False	(1) 23 z False
IF-1	≥ 25	m	True	(2) 30 m True
IF-2	*	m	True	(2)
IF-2	*	z	False	(1)
IF-3	≤ 45	*	*	(1)
IF-3	> 45	*	*	(3) 70 z False
IF-3	< 65	*	*	(2)
IF-3	≥ 65	*	*	(3)



Zadatak 3 - Pokrivenost uslova - Rešenje (2)

- Napomena: Ovi test primeri ne izvršavaju *then* klauzule IF-1 i IF-3, kao i (praznu) *else* klauzulu IF-2.
- Pokrivanje svih uslova može se realizovati sa najmanje 3 test primera (godine, pol, bračni status):

  TP-1: 23, z, *False*

  TP-2: 30, m, *True*

  TP-3: 70, z, *False*

Zadatak 4 - Deljenje sa nulom

- Neka je dat deo programskog koda u programskom jeziku Java:

```
1. int z = 0;
2. if(a > b)
3.     z = 12;
4. int rezultat = 72 / z;
```

- Napisati test primere koji vrše:

  a) Pokrivanje iskaza;

  b) Pokrivanje odluka;

  c) Pokrivanje uslova;

- Da li izvršavanjem neke tehnike može doći do potencijalnog problema?

Zadatak 4 - Deljenje sa nulom - Rešenje

```
1. int z = 0;
2. if(a > b)
3.     z = 12;
4. int rezultat = 72 / z;
```

- a) 3 iskaza (na liniji 1, na liniji 3 i na liniji 4)
  Linije 1 i 4 ćemo svakako pokriti prilikom svakog prolaska kroz ovaj program.
- Da bismo obuhvatili i iskaz 3, uslov a > b mora biti ispunjen (*TRUE*).
  Dakle, test primer mora da zadovolji uslov a > b, pa realizovan test primer može biti:
  TP1: a = 3, b = 2 => rezultat = 6
- b) i c) Pokrivanje uslova i pokrivanje odluka je identično, pošto je uslov u liniji 2 prost.
  Moramo da pokrijemo prolazak kroz *THEN* granu (ispunjenost uslova na *TRUE*). TP1 već pokriva to.
- Drugi prolazak koji moramo pokriti je prolazak kroz *ELSE* granu (odnosno iskočiti iz IF).
  To je zapravo ispunjenost uslova na *FALSE*.
- Uslov a > b mora biti neispunjen, odnosno ispunjen b ≥ a, pa drugi realizovan test može biti:
  TP2: a = 2, b = 3 => rezultat = ? ***potencijalni problem!!!***

Zadatak 5 - Pokrivenost odluka i uslova

- Za datu funkciju implementiranu u programskom jeziku *Python* odrediti skup test primera, koji pokriva sve odluke i uslove.

```
def osiguranje(godine, pol, brakStatus):
    premija = 500
    if ((godine < 25) and (pol == 'm') and (not brakStatus)):
        premija=premija + 1500
    else:
        if (brakStatus or (pol == 'z')):
            premija = premija - 200
        if ((godine > 45) and (godine < 65)):
            premija = premija - 100
    print(premija)
```

Zadatak 5 - Pokrivenost odluka i uslova - Rešenje

```
def osiguranje(godine, pol, brakStatus):
    premija = 500
    if ((godine < 25) and (pol == 'm') and (not brakStatus)):
        premija=premija + 1500
    else:
        if (brakStatus or (pol == 'z')):
            premija = premija - 200
        if ((godine > 45) and (godine < 65)):
            premija = premija - 100
    print(premija)
```

- Ova tabela je nastala prostim spajanjem svih odluka (iz tabele pokrivenosti odluka) i svih uslova (iz tabele pokrivenosti uslova), a zatim je minimizovan broj test primera.

Pokrivene odluke i uslovi	Godine	Pol	Bračni status	Test primer
IF-1(odluka)	< 25	m	False	(1) 23 m False
IF-1 (odluka)	< 25	z	False	(2) 23 z False
IF-1 (uslov)	< 25	z	False	(2)
IF-1 (uslov)	>= 25	m	True	(3) 70 m True
IF-2 (odluka)	*	z	*	(2)
IF-2 (odluka)	>= 25	m	False	(4) 50 m False
IF-2 (uslov)	*	m	True	(3)
IF-2 (uslov)	*	z	False	(2)
IF-3 (odluka)	<= 45	*	*	(2)
IF-3 (odluka)	> 45, < 65	*	*	(4)
IF-3 (uslov)	<= 45	*	*	(2)
IF-3 (uslov)	> 45	*	*	(4)
IF-3 (uslov)	< 65	*	*	(4)
IF-3 (uslov)	>= 65	*	*	(3)

Zadatak 6 - Pokrivenost višestrukih uslova

- Za datu funkciju implementiranu u programskom jeziku *Python* odrediti skup test primera, koji pokriva sve višestruke uslove.

```
def osiguranje(godine, pol, brakStatus):
    premija = 500
    if ((godine < 25) and (pol == 'm') and (not brakStatus)):
        premija=premija + 1500
    else:
        if (brakStatus or (pol == 'z')):
            premija = premija - 200
        if ((godine > 45) and (godine < 65)):
            premija = premija - 100
    print(premija)
```

Zadatak 6 - Rešenje

Višestruki uslovi	Godine	Pol	Bračni status	Test primer
IF-1	< 25	m	True	(1) 23 m True
IF-1	< 25	m	False	(2) 23 m False
IF-1	< 25	z	True	(3) 23 z True
IF-1	< 25	z	False	(4) 23 z False
IF-1	≥ 25	m	True	(5) 30 m True
IF-1	≥ 25	m	False	(6) 70 m False
IF-1	≥ 25	z	True	(7) 50 z True
IF-1	≥ 25	z	False	(8) 30 z False
IF-2	*	m	True	(5) //može i (1)
IF-2	*	m	False	(6)
IF-2	*	z	True	(7) //može i (3)
IF-2	*	z	False	(8) //može i (4)
IF-3	≤ 45, ≥ 65	*	*	nije moguće
IF-3	≤ 45, < 65	*	*	(8)
IF-3	> 45, ≥ 65	*	*	(6)
IF-3	> 45, < 65	*	*	(7)