Oblikovanje programske potpore

Završni ispit

1. veljače 2019.

Izjavljujem da tijekom izrade ove zadaće neću od drugoga primiti niti drugome pružiti pomoć, te da se neću koristiti nedopuštenim sredstvima. Ove su radnje teška povreda Kodeksa ponašanja te mogu uzrokovati i trajno isključenje s Fakulteta. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.

JMBAG	*Ime i prezime*	*Vlastoručni potpis*

Minimum za prolaz završnog ispita je 12 bodova, maksimum 36 bodova **GRUPA B**

1. (1 bod) Generičke aktivnosti procesa programskog inženjerstva mogu se podijeliti u četiri osnovne skupine. Navedite ih prema vremenskom redoslijedu.

   Rj. Specifikacija, oblikovanje i implementacija, validacija i verifikacija, evolucija.

2. (1 bod) Navedite tipične načine opisa zahtjeva sustava.

   Rj. strukturiranim prirodnim jezikom, specijalnim jezikom za opis oblikovanja (npr. SDL), grafičkom notacijom (npr. UML) i matematičkom specifikacijom (FSM, teorija skupova, logika).

3. (1 bod) Što je to projektni dnevnik zaostataka (engl. *product backlog*) kod pristupa SCRUM radnog okvira? Tko je odgovoran za sadržaj, raspoloživost i sortiranje dnevnika zaostataka?

   Rj. Sortirana lista svih zahtjeva na proizvod i jedino mjesto zahtjeva za bilo kakvim promjenama. Vlasnik proizvoda (engl. product owner) je odgovoran za njegov sadržaj, raspoloživost i sortiranje.

4. (1 bod) Kako se naziva princip dobrog oblikovanja programske potpore kod kojeg se traži grupiranje međusobno povezanih elemenata, a sve ostale elemente se stavlja izvan grupe?

   Rj. Povećanje kohezije

5. (1 bod) Koja su 3 osnovna tipa vidljivosti u UML-dijagramu razreda, kako ih označavamo i što ona znače?

   Rj.

   Javni (public, +) – svi razredi mogu pristupiti atributu / metodi

   Zaštićeni (protected, #) – svi razredi iz hijerarhije podrazreda ili iz samog razreda mogu pristupiti tom atributu / metodi

   Privatni (private, -) – samo je moguće pristupiti iz samog razreda tom atributu / metodi

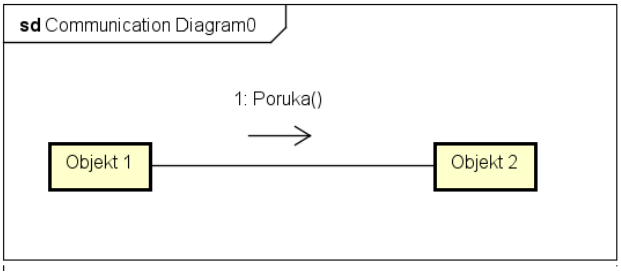
6. (1 bod) Što je metoda u objektno orijentiranim programskim jezicima i kako se prikazuje poziv metode na UML-sekvencijskom dijagramu?

   Rj. Metoda je način izvođenja ili implementacija neke operacije. To je procedura, funkcija, rutina,proceduralna apstrakcija koja se koristi za implementaciju ponašanja razreda. Poziv metode prikazuje se strelicom s nazivom poruke (metoda je poziv ili slanje poruka).

7. (1 bod) Definirajte alociranje odgovornosti (engl. *responsibility*) razredima.

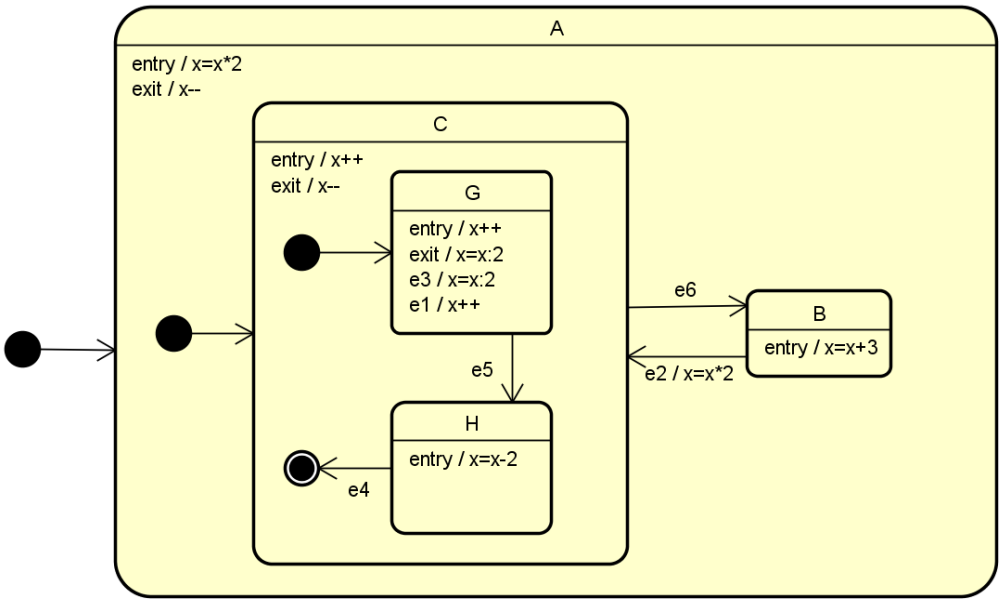
Rj. Odgovornost (engl. *responsibility*) je nešto što sustav mora izvršiti.
Sve odgovornosti jednog razreda moraju biti jasno povezane.
Ako jedan razred ima previše odgovornosti, razmotri podjelu toga razreda u različite razrede.
Ako razred nema odgovornosti, tada je vjerojatno beskoristan.
Ako se neka odgovornost ne može pripisati niti jednom od postojećih razreda, mora se kreirati novi razred.

8. (1 bod) Koji tip poruke je prikazan na komunikacijskom dijagramu na slici?



Rj. Asinkrona poruka (engl.*asynchronous message*)

9. (2 boda) Za zadani UML-dijagram stanja odredite vrijednost varijable x po završetku izvođenja događaja (ulazna vrijednost je **x = 2**), ako je redoslijed događaja koji se izvode: **e5, e6, e2, e3.** Navedite akcije!



2, 4, 5, 6|3, 1| 0, 3| 6, 7, 8| 4
Rj. x = 4

10. (1 bod) Objasnite što su značke i kako se kreću u UML dijagramu aktivnosti (engl. a*ctivity diagram*).

Rj. Značke su dio semantike bez grafičkog prikaza. Značka može predstavljati: upravljački tijek; objekt; podatak. Kreću se od izvorišta prema odredištu vezama ovisno o: ispunjenim uvjetima izvornog čvora, postavljenim uvjetima veza (engl. Edge guard conditions), preduvjetima ciljnog čvora.

11. (1 bod) Kod kojeg je arhitekturnog obrasca (engl. *architectural pattern*) jedna od temeljnih značajki implicitno pozivanje komponenata?

Rj: Kod arhitekture zasnovane na događajima (engl. *event-based architecture*).

12. (1 bod) Koji je cilj provođenja aktivnosti ispitivanja u procesu oblikovanja programske potpore?

Rj. Otkrivanje informacija o ispravnosti i kvaliteti, te poboljšanja pronalaženjem kvarova i problema ispitivane programske podrške.

Priznaje se i kraće rješenje: pronalaženje pogrešaka.

13. (2 boda) Za funkcijsko ispitivanje ekvivalentnim particijama (engl. *equivalence partition*):

a) (1 bod) Navedite korake ispitivanja.

b) (1 bod) Na primjeru jednog ulaznog 3-znamenkastog broja u intervalu [700, 997], navedite potrebne minimalne ispitne slučajeve.

Rj.

1. Odredi particije za sve ulazne varijable.
2. Za sve particije odaberi vrijednosti ispitivanja.
3. Definiraj ispitne slučajeve koristeći odabrane vrijednosti.
4. Odredi očekivane izlaze za odabrane ispitne slučajeve i provedi ispitivanje.

Ispitni slučaj	ulazna vrijednost	očekivani izlaz
IS1	699	ne prolazi
IS2	700	prolazi
IS3	801	prolazi
IS4	997	prolazi
IS5	998	ne prolazi

*priznaju se i dodatni ispitni slučajevi: (npr. 500, ne prolazi) (npr. 1200, ne prolazi)

14. (2 boda) Za sljedeću funkciju:

a) (1 bod) Nacrtajte graf tijeka programa (engl. *control flow graph*). Uz program navedite odgovarajuće oznake na grafu.

b) (1 bod) Odredite gornju granicu broja ispitnih slučajeva koja jamči potpuno pokrivanje svih naredbi programa. Navedite formulu i izračunajte.

```
public void TDM(A[], x, y , z , B[])
{
     int i = x;
     int j = y;
     int k = x;
     do {
       if (i < y && (j >= z  && A[i] <= A[j])) {
             B[k] = A[i++];
         } else {
             B[k] = 4+A[j]/2;
             j = j++;
         }
        k++;
     } while (k < z);
     return;
}
```

Rj.

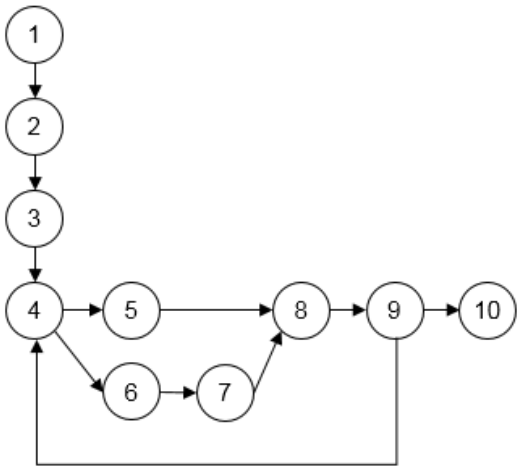
a) Ubaciti novo rješenje!!

```
public void TDM(A[], x , y , z , B[])
{
    int i = x ;  (1)
    int j = y ;  (2)
    int k = x;   (3)

    do {
          if (i < y && (j >= z || A[i]<= A[j])) (4) {
             B[k] = A[i++]; (5)
          } else {
             B[k] = 4+A[j]/2; (6)
             j = j++;   (7)
          }
          k++; (8)
    } while (k < z);(9)
    return; (10)



}
```



b) CV(G) = 11 - 10 +2*1 =3

15. (1 bod) Definirajte logičku posljedicu (Γ |= φ) putem modela ili putem teorema o dedukciji.

Rj. φ je logička posljedica skupa formula Γ ako je svaki model od Γ ujedno i model od φ ili

φ je logička posljedica skupa formula Γ ako i samo ako je (Γ ⇒ φ) tautologija (priznaje se i: Γ ∧ ¬φ kontradikcija)

16. (1 bod) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:

"*U poslužitelju Sparc T8-4 postoje barem dva procesora*."

Rj. poslužitelj(x) - x je poslužitelj

procesor(x) - x je procesor

postoji_u(x,y) – x postoji u y

=(x,y) - x je jednako y

Sparc T8-4 - konstanta

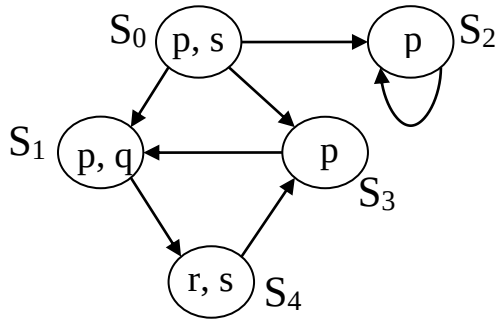
poslužitelj(Sparc T8-4) ⇒ ∃x∃y (procesor(x) ∧ procesor(y) ∧ postoji_u(x, Sparc T8-4) ∧ postoji_u(y, Sparc T8-4) ∧ ¬=(x,y)))

17. (1 bod) Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. *Computational Tree Logic*):

" U svakom stanju nakon početnog stanja (ali ne i u početnom stanju) postoji put na kojem cijelo vrijeme vrijedi stop=1."

Rj. AX EG (stop=1), priznaje se i: AG AX EG (stop=1), jer se moglo i tako shvatiti

18. (1 bod) Za zadani model implementacije Kripke strukturom M prema slici potrebno je odrediti skup svih stanja koja zadovoljavaju formulu **EG (p)**.



Rj. Takva stanja su = S0, S2

19. (2 boda) Za arhitekturu protoka podataka, navedite i objasnite podjelu prema vrsti izvršavanja obrade podataka.

Rj. Skupno – sekvencijska (engl. Batch-sequential)

Podaci se između koraka (podsustava za obradu) prenose u cijelosti

Svaki podsustav se izvodi do kraja prije prelaska na idući korak

Svaki podsustav je neovisan o drugima

Cjevovodi i filteri (engl. Pipes-and-filters)

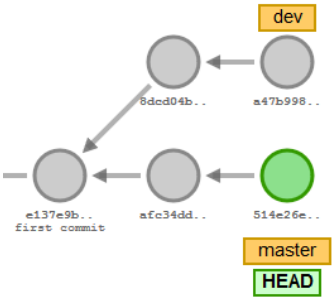
Inkrementalna, konkurentna transformacija podataka kroz korake (filtere), čim podaci postanu dostupni

Komponente: izvori podataka, filteri, cjevovodi i ponori podataka

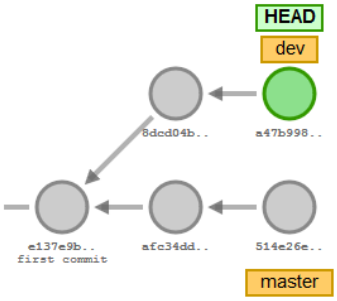
Filteri su neovisni o ostalim filterima, ne pamte stanje

Cjevovodi su FIFO međuspremnici u obliku U/I tokova podataka (input/output stream)

20. (1 bod) Kojom git naredbom repozitorij prelazi iz stanja sa slike A) u stanje sa slike B)?



A)



B)

Rj. git checkout dev

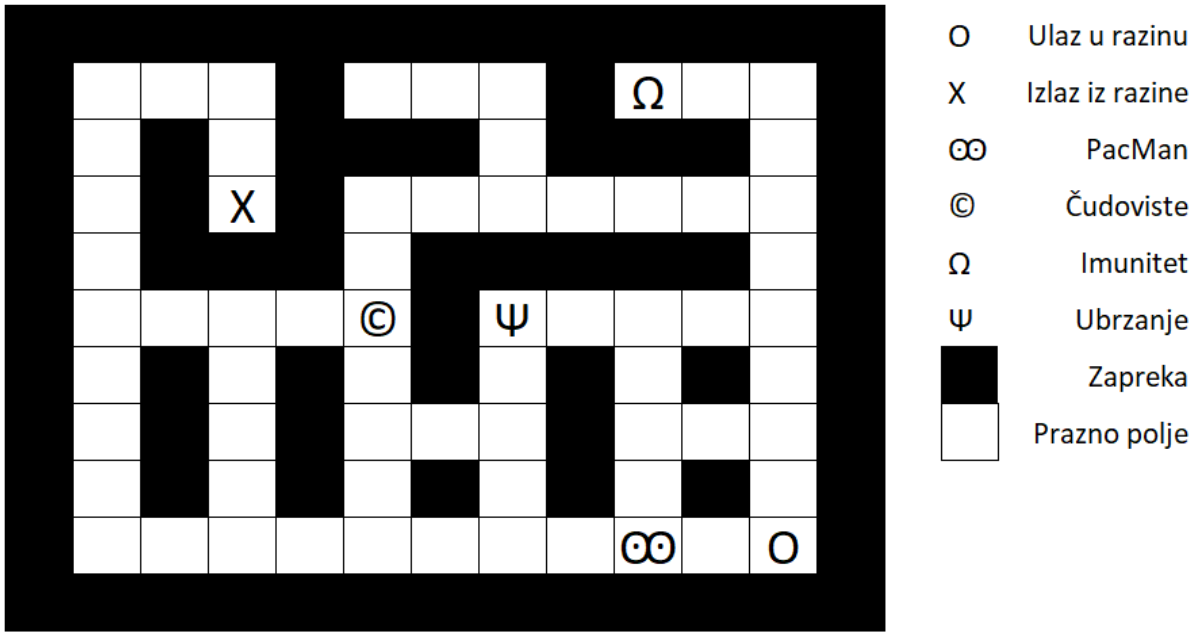
Problemski dio **– Pac-Man**

Igra se izvodi na virtualnoj ploči s 40x20 polja i sastoji se od 13 razina. Postoje 4 vrste polja: slobodno, zapreka, bonus, polje imuniteta i za svaku razinu zadan je drukčiji raspored vrsta polja. Po ploči se kreće figura Pac-Man kojim upravlja igrač i jedno čudovište kojim upravlja računalo. U normalnom stanju i Pac-Man i čudovište se kreću u koracima istovremeno, polje po polje, istom brzinom (1 polje u sekundi).

Pac-Man može prelaziti preko svih vrsta polja osim preko zapreke. Ako dođe na bonus polje, Pac-Man se sljedeće 2 sekunde kreće dva puta većom brzinom od normalne, a ako dođe na polje imuniteta Pac-Man postaje imun na ugriz čudovišta sljedeće 2 sekunde i za to vrijeme titra zeleno. Pac-Man ne može dobiti novi bonus ili imunitet dok je pod utjecajem imuniteta ili ubrzanja.

Pac-Man ima 3 života. Svaki put kada ga dostigne čudovište, Pac-Man gubi život osim ako ima imunitet. Cilj je provesti Pac-Man-a od početnog do završnog polja prije nego što izgubi sve živote. Ukoliko Pac-Man izgubi život, a ima još preostalih života, čudovište se vraća na svoje početno polje, a Pac-Man nastavlja dalje 50% manjom brzinom od normalne sljedeće 2 sekunde, osim ako prije isteka te 2 sekunde ne stane na polja bonus (nastavlja 2x većom brzinom od normalne 2 sekunde) ili imunitet (nastavlja normalnom brzinom s imunitetom 2 sekunde). Ako Pac-Man dođe do izlaza prije nego što ga čudovište dostigne, prelazi na sljedeću razinu, a ako izgubi sve živote prije toga, igra završava.

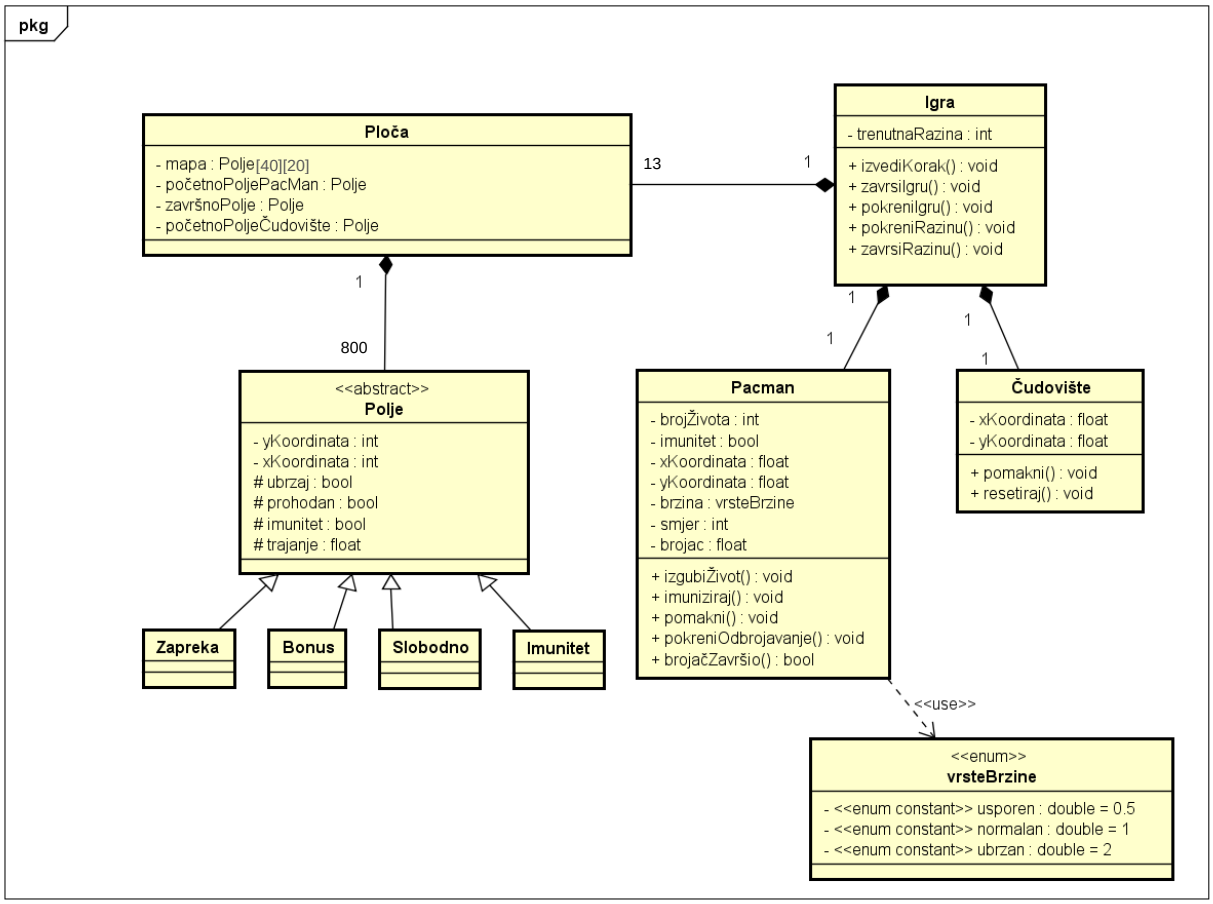
Igra će se implementirati kao web aplikacija koja se izvodi i igra putem web preglednika. Igrač se može registrirati kako bi igra pamtila njegov napredak i najbolje rezultate (engl. *Hall of Fame*). Svi najbolji rezultati se pohranjuju u bazi podataka s kojom je povezan web poslužitelj. Klijent je s poslužiteljem povezan koristeći sigurnu HTTPS vezu. Web poslužitelj i poslužitelj baze podataka se nalaze na različitim računalima.



21. (4 boda) Dijagram razreda

UML-dijagramom razreda (engl. *Class Diagram*) modelirajte igru Pac-Man u korisničkom web pregledniku. Pri modeliranju zanemarite dio igre vezan uz pamćenje napretka i rezultata. U rješenju obavezno koristite bar jedan apstraktni razred.

Rj.

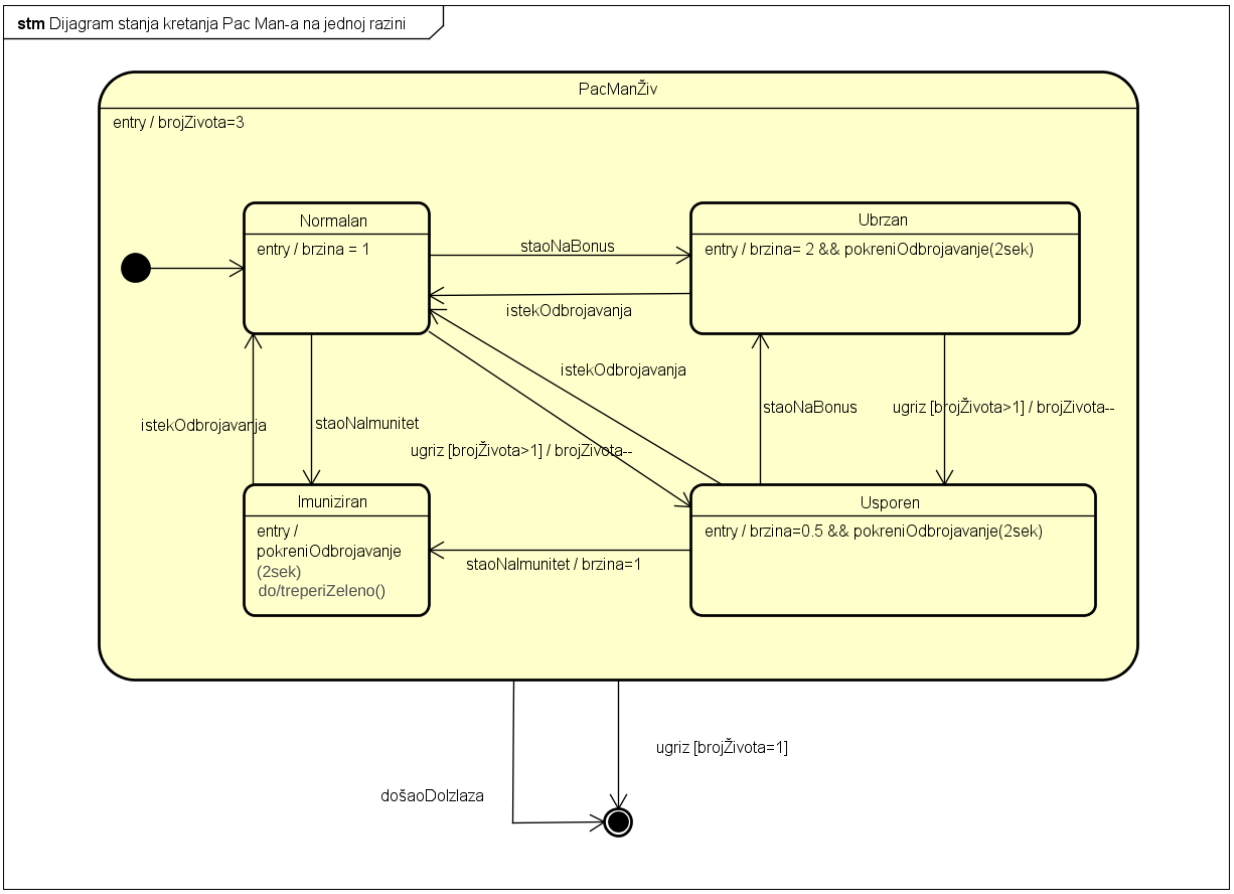


Kriteriji bodovanja:

- Dobro identificirani razredi – 1.25 bodova
- Veze između razreda – 1.25 bodova
- atributi i metode – 1 bod
- višestrukost – 0.5 bodova

22. (4 boda) Dijagram stanja

UML-dijagramom stanja (engl. *Statechart Diagram*) modelirajte stanja figure Pac-Mana u interakciji s različitim vrstama polja i čudovištem, na jednoj razini igre. Pri razradi dijagrama stanja obavezno upotrijebite varijable vezane uz život i brzinu. Pri modeliranju zanemarite dio igre vezan uz pamćenje rezultata i promjenu razine.

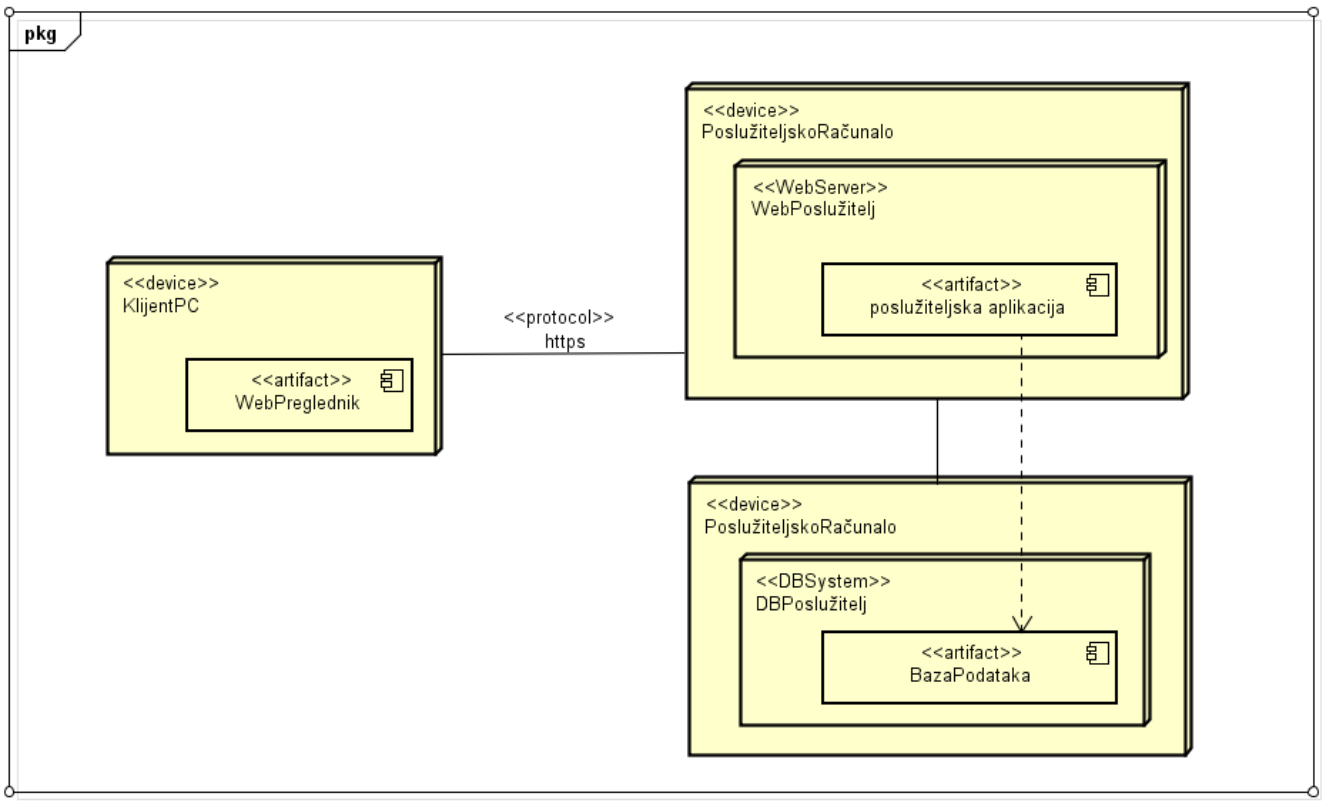


Kriteriji bodovanja:

- Prikazana sva stanja automata: 1,5 bod
- Prijelazi vođeni događajima: 1 bod
- Semantika uvjeta i grananja: 0,5 bod
- Ispravna notacija stanja (naziv, varijable stanja i akcije *entry*/*do*/*exit*): 1 bod

23. (4 boda) Dijagram razmještaja

Izradite UML specifikacijski dijagram razmještaja (engl. *Specification-level Deployment Diagram*) igre Pac-Man. Nazive odredite razumljivo, te navedite odgovarajuće stereotipe.



Kriteriji bodovanja:

- Prikazani svi čvorovi: 1.5 bod
- Prikazane sve komponente: 1 bod
- Prikazane sve veze između čvorova: 0,5 boda
- Prikazane sve veze između komponenti: 0.5 boda
- Stereotipi ispravno navedeni: 0.5 boda