4. laboratorijska vježba

Važna napomena: u svim zadacima potrebno je napisati Javadoc komentare za svaki razred, sučelje i enumeraciju te generirati dokumentaciju. Svi nazivi razreda, metoda i varijabli i slično moraju biti na engleskom. Sav napisani programski kod mora biti napisan u skladu s konvencijama imenovanja varijabli, metoda i razreda (varijable i metode: malo početno slovo, camel-case; razredi i sučelja: veliko početno slovo, camel-case; konstante: uobičajeno sve veliko i razdvajanje podvlakom) te ostalim pozitivnim praksama (uključivo i korektno uvlačenje redaka; smisleno razdvajanje više različitih semantički grupiranih redaka praznim recima, pravilnim razmještajem otvorene i zatvorene vitičaste zagrade i slično). Za više informacija pogledajte http://www.oracle.com/technetwork/java/codeconventions-150003.pdf.

U okviru ove laboratorijske vježbe potrebno je modelirati osnovne entitete nogometne utakmice. Rješenje laboratorijske vježbe napravite u Java projektu `oop-lab4`. Sve razrede, sučelja i enumeracije smjestite u odgovarajuće pakete `hr.fer.oop.lab4.prob<BROJ_ZADATKA>`. **Dodatno, rješenje demonstrirajte pisanjem** `main` **metode u razredima** `Demonstration` **za svaki zadatak posebno.**

S obzirom da se zadatci nadovezuju, savjetujemo da ih rješavate po redu. Po potrebi generirajte odgovarajuće *gettere* i *settere*. **Izbjegavajte** magične brojeve (http://stackoverflow.com/questions/47882/what-is-a-magic-number-and-why-is-it-bad). Ako nadjačavate metodu `equals(Object o)`, **obavezno nadjačajte** i metodu `hashcode()` (http://stackoverflow.com/questions/113511/best-implementation-for-hashcode-method). Savjetujemo da **konzultirate priložene UML dijagrame** koji se nalaze na kraju ovog dokumenta. **Naučite koristiti lambda izraze** u slučaju kada metodama trebate predati `Predicate` (npr. metoda `filterRegisteredPlayers(Predicate<FootballPlayer> criteria)` u sučelju `IManageableTeam`).

Zadatak 1.: Osoba – Trener i Nogometaš

Nogomet treniraju i igraju osobe. Apstraktna osoba `Person` ima ime `name`, državu `country` (tekst) i emociju `emotion` koja je cijeli broj od 0 do 100. Pretpostavimo da su dvije osobe jednake ako imaju jednako ime, državu i emociju.

Trener `Coach` je osoba koja ima vještinu `coachingSkill` (cijeli broj od 0 do 100) i omiljenu formaciju `formation` (enumeracija `Formation` sa sljedećim vrijednostima[1]: `F442`, `F352`, `F541`).

Nogometaš `FootballPlayer` je osoba koja ima vještinu `playingSkill` (cijeli broj od 0 do 100) i prirodnu poziciju `playingPosition` gdje igra (enumeracija `PlayingPosition` s vrijednostima `FW`, `MF`, `DF` i `GK` koje označavaju napad, sredinu, obranu i vratarsku poziciju).

Trener se, baš kao i nogometaš, u potpunosti inicijalizira jednim konstruktorom. Ime, država i emocija im se nakon toga ne mogu promijeniti dok se sve ostalo može. Sve članske varijable

[1] Npr. `F442` označava da se radi o formaciji 4-4-2: jedan vratar, četiri braniča, četiri srednja (vezna) igrača i 2 napadača.

potrebno je ispravno postavljati (tj. ne smiju biti `null`), a vrijednosti moraju biti u odgovarajućem rasponu. Inače se izbacuje neprovjeravana iznimka `IllegalArgumentException` (npr. ako se osobi želi postaviti ime koje je `null` ili ako je vještina izvan traženog raspona) koja ujedno sadrži i odgovarajući tekst o pogrešci (npr. „Ime ne smije biti null!“).

Demonstracija: http://pastebin.com/19bKHGwm

Zadatak 2.: Tim – Nacionalni i Klupski

Apstraktni nogometni tim `Team` ima svoj naziv `name`, formaciju `formation`, kolekciju registriranih igrača `registeredPlayers` te kolekciju igrača `startingEleven` koji čine početnu jedanaestoricu. Jedino registrirani igrači mogu biti izabrani za početnu jedanaestoricu. *Važno je napomenuti kako jedan nogometaš ne može biti registriran za isti tim dvaput niti može biti dodan dvaput u početnu jedanaestoricu (tj.* ***nema duplikata****)*.

Nacionalni tim `NationalTeam` je nogometni tim koji ima državu `country` (tipa `String`) te ukupno može imati do 23 igrača. *Naravno, registrirani igrači moraju imati odgovarajuću državu*. Nacionalni tim se inicijalizira konstruktorom koji postavlja ime, formaciju i državu. Naziv i država se nakon toga ne mogu mijenjati.

Klupski tim `ClubTeam` je nogometni tim koji ima reputaciju `reputation` (cijeli broj od 0 do 100) te ukupno može imati do 25 igrača. *Registrirani igrači mogu biti samo oni čija je vještina veća ili jednaka reputaciji klupskog tima*. Klupski tim se inicijalizira konstruktorom koji postavlja ime, formaciju i reputaciju. Naziv se nakon inicijalizacije ne može mijenjati.

Sve članske varijable potrebno je ispravno postaviti (npr. ne smiju biti `null`) i u odgovarajućem rasponu. Inače, baca se neprovjeravana iznimka `IllegalArgumentException` (npr. ako se želi postaviti formacija `null` ili ako je reputacija izvan traženog raspona) koja ujedno sadrži i odgovarajući tekst o pogrešci (npr. „Formacija ne smije biti null!“).

Timom se može upravljati na način kako je definirano u sučelju `IManageableTeam`:

- `public void registerPlayer(FootballPlayer player) throws NotEligiblePlayerException`
    - Služi za dodavanje igrača u kolekciju registriranih igrača.
    - `NotEligiblePlayerException` – provjeravana iznimka koja se baca u slučaju da nogometaš nema pravo na registraciju u tim (npr. igrač i nacionalni tim nemaju jednaku državu, igrač ima premalu vještinu u odnosu na reputaciju kluba, više nema mjesta u kolekciji registriranih igrača, igrač je već registriran). Iznimka sadrži i tekst u kojem se navodi razlog zašto nogometaš nema pravo na registraciju (npr. „Nogometaš Olenbe (Croatia) ne može se registrirati u nacionalni tim France jer nema odgovarajuću državu.“).
- `public void unregisterPlayer(FootballPlayer player) throws IllegalArgumentException`
    - Služi za uklanjanje igrača iz kolekcije registriranih igrača.

- `IllegalArgumentException` – baca se u slučaju da se igrač ne nalazi u kolekciji registriranih igrača. U njoj je sadržana i poruka o pogrešci (npr. „Igrač Kobsic ne nalazi se u kolekciji registriranih igrača“).

- `public void clearStartingEleven();`
  - Prazni kolekciju početne jedanaestorice.
- `public void addPlayerToStartingEleven(FootballPlayer player) throws NotEligiblePlayerException;`
  - Dodaje igrača u početnu jedanaestoricu.
  - `NotEligiblePlayerException` – baca se u slučaju da se igrač ne nalazi u kolekciji registriranih igrača, više nema mjesta u prvih 11 ili je igrač već uključen u prvih 11. Iznimka sadrži i konkretniji opis pogreške (npr. „Igrač Prekinoski ne nalazi se u kolekciji registriranih igrača tima BRAZIL“).
- `public void removePlayerFromStartingEleven(FootballPlayer player) throws IllegalArgumentException;`
  - Uklanja igrača iz početne jedanaestorice.
  - `IllegalArgumentException` – baca se u slučaju da se igrač ne nalazi u prvih 11. U njoj je sadržana i poruka o pogrešci (npr. „Igrač Pilkotesa ne nalazi se u prvih 11.“).
- `public void setFormation(Formation formation);`
  - Služi za postavljanje željene formacije.
- `public Formation getFormation();`
  - Dohvat formacije.
- `public Collection<FootballPlayer> getRegisteredPlayers();`
  - Vraća **NOVU** kolekciju koja sadrži registrirane igrače.
- `public Collection<FootballPlayer> getStartingEleven();`
  - Vraća **NOVU** kolekciju koja sadrži članove prve jedanaestorice.
- `public boolean isPlayerRegistrable(FootballPlayer player);`
  - Vraća true ako se igrač može registrirati u tim: ako ima mjesta u timu (tj. konkretnije u kolekciji registriranih igrača). **Nacionalni timovi:** ako igrač ima odgovarajuću državu. **Klupski timovi:** ako je igračeva vještina veća ili jednaka reputaciji kluba.
- `public Collection<FootballPlayer> filterRegisteredPlayers(Predicate<FootballPlayer> criteria)`
  - Vraća NOVU kolekciju koja sadrži registrirane igrače koji zadovoljavaju kriterij `criteria`. Ako nema niti jednog takvog igrača, vraća praznu kolekciju.

Nogometni tim je spreman za utakmicu ako zadovoljava minimalan kriterij: **broj igrača u listi početne jedanaestorice je barem 7 (ne morate provjeravati nalazi li se vratar u tom broju)**. Kako bi se to moglo provjeriti, tim implementira sučelje `IMatchInspectableTeam`:

- `public boolean isMatchReady();`
  - Služi za provjeru je li tim spreman za utakmicu. Vraća `true` ako je spreman.
- `public int calculateTeamSpirit();`
  - Računa i vraća momčadski duh[2] koji je definiran kao zbroj emocija početne jedanaestorice.
- `public int calculateTeamSkill();`
  - Računa i vraća vještinu tima koja je definirana kao zbroj vještina početne jedanaestorice.
- `public double calculateRating();`
  - Računa i vraća ocjenu tima koja je definirana:
    - klupski tim: 70% vještina tima + 30% momčadski duh
    - nacionalni tim: 30% vještina tima + 70% momčadski duh

Demonstracija: http://pastebin.com/4S4XAFji

Zadatak 3.: Upravljanje timom

Kako bi trener mogao upravljati ili klupskim ili nacionalnim timom, dodajte mu novu člansku varijablu `managingTeam` koja je tipa `IManageableTeam`.

Trener zna poslove upravljanja timom koji su definirani sučeljem `IManager`:

- `public void registerPlayers(Iterable<FootballPlayer> offeredPlayers, Predicate<FootballPlayer> criteria) throws UnemployedCoachException;`
  - Služi za registraciju igrača od onih ponuđenih (`offeredPlayers`) na temelju nekog kriterija `criteria`. Metoda ne uklanja postojeće registrirane igrače.
  - Implementacija bi trebala paziti na ograničenja što se tiče broja registriranih igrača. U slučaju da veći broj ponuđenih igrača zadovoljava zadani predikat, pri čemu registriranje svih nije moguće zbog ograničenja o maksimalnom broju igrača, uzeti prvih *n* dok se tim ne popuni. Također, metoda neće ni pokušati registrirati igrača za tim ako igrač nema pravo na registraciju (takav pokušaj bi uzrokovao iznimku, ali bolje rješenje je prethodno provjeriti ispravnost i odustati od pokušaja registracije).

[2] Izraz *momčadski duh* koristimo isključivo zbog sveopće prihvaćenosti prijevoda pojma *team spirit*. Drugim riječima, laboratorijska vježba podrazumijeva da nogometaš može biti osoba bilo kojeg spola.

- `public void pickStartingEleven(Predicate<FootballPlayer> criteria) throws UnemployedCoachException;`
    - Služi za odabir početne jedanaestorice iz kolekcije registriranih igrača tima kojim trener upravlja. Pri tome se služi kriterijem odabira `criteria`. Neovisno o kriteriju odabira, metoda bi trebala paziti da odabrana jedanaestorica mogu činiti formaciju tima (npr. broj napadača u početnoj jedanaestorici odgovara broju napadača u formaciji tima). *U slučaju da predikat zadovoljava više od potrebnog broja igrača na nekoj poziciji, odabrati one koji su **ranije** dodani u kolekciju registriranih igrača*. Moguće je da nakon ove metode tim neće imati svih 11 igrača, ali to se smatra valjanom situacijom (iako tim možda neće biti spreman za utakmicu).
- `public void forceMyFormation() throws UnemployedCoachException;`
    - Trener timu postavlja svoju omiljenu formaciju.
- `public void setManagingTeam(IManageableTeam team);`
    - Treneru se postavlja tim kojim može upravljati.

  Ako je trener nezaposlen (tj. članska varijabla `managingTeam` je `null`) baca se provjeravana iznimka `UnemployedCoachException`.

Demonstracija: http://pastebin.com/8KV1CFqw

Zadatak 4.: Utakmica

Utakmica `Match` se sastoji od dva tima koja implementiraju sučelje `IMatchInspectableTeam`: domaći `home` i gosti `away`. Utakmica ima svoj tip `type`, odnosno, može biti natjecateljska ili prijateljska (enumeracija `MatchType` s vrijednostima `COMPETITIVE` i `FRIENDLY`). Također, ima i ishod `outcome` (enumeracija `MatchOutcome` s vrijednostima `NOT_AVAILABLE`, `HOME_WIN`, `DRAW`, `AWAY_WIN`). Konstruktor inicijalizira članske varijable `home`, `away` i `type` koje se nakon toga ne mogu mijenjati. Podrazumijevani ishod utakmice je `NOT_AVAILABLE`.

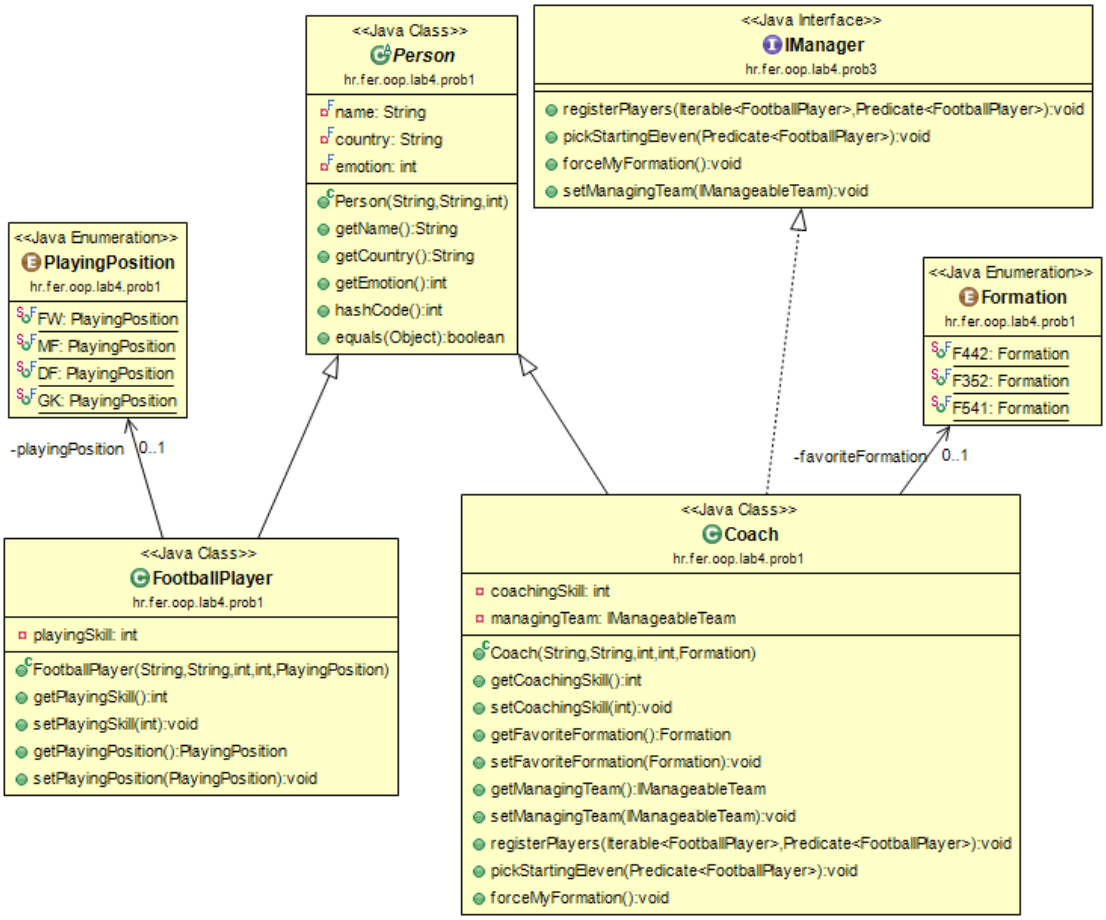
Utakmica može biti igriva, odnosno, implementira sučelje `IPlayableMatch`:

- `public void play() throws NotPlayableMatchException;`
    - Utakmica se odigrava na način da se ocjena pojedinog tima iskaže relativno u odnosu na sumu ocjena obaju timova čime se dobiju vrijednosti **h**(ome), **a**(way) i **min** takve da je **h + a = 1** i **min = min(h,a)**. Rezultat utakmice se određuje generiranjem slučajnog broja u rasponu od 0 do 1 pri čemu je utakmica završila pobjedom domaćina ako je **slučajni broj < h-min/2**, pobjedom gosta ako je **slučajni broj > h + min/2**, a inače neriješeno. Ishod se postavlja u članskoj varijabli `outcome`.
    - `NotPlayableMatchException` je provjeravana iznimka koja se baca u slučaju da utakmica nije igriva. Iznimka sadrži i kratak opis zašto utakmica nije igriva.

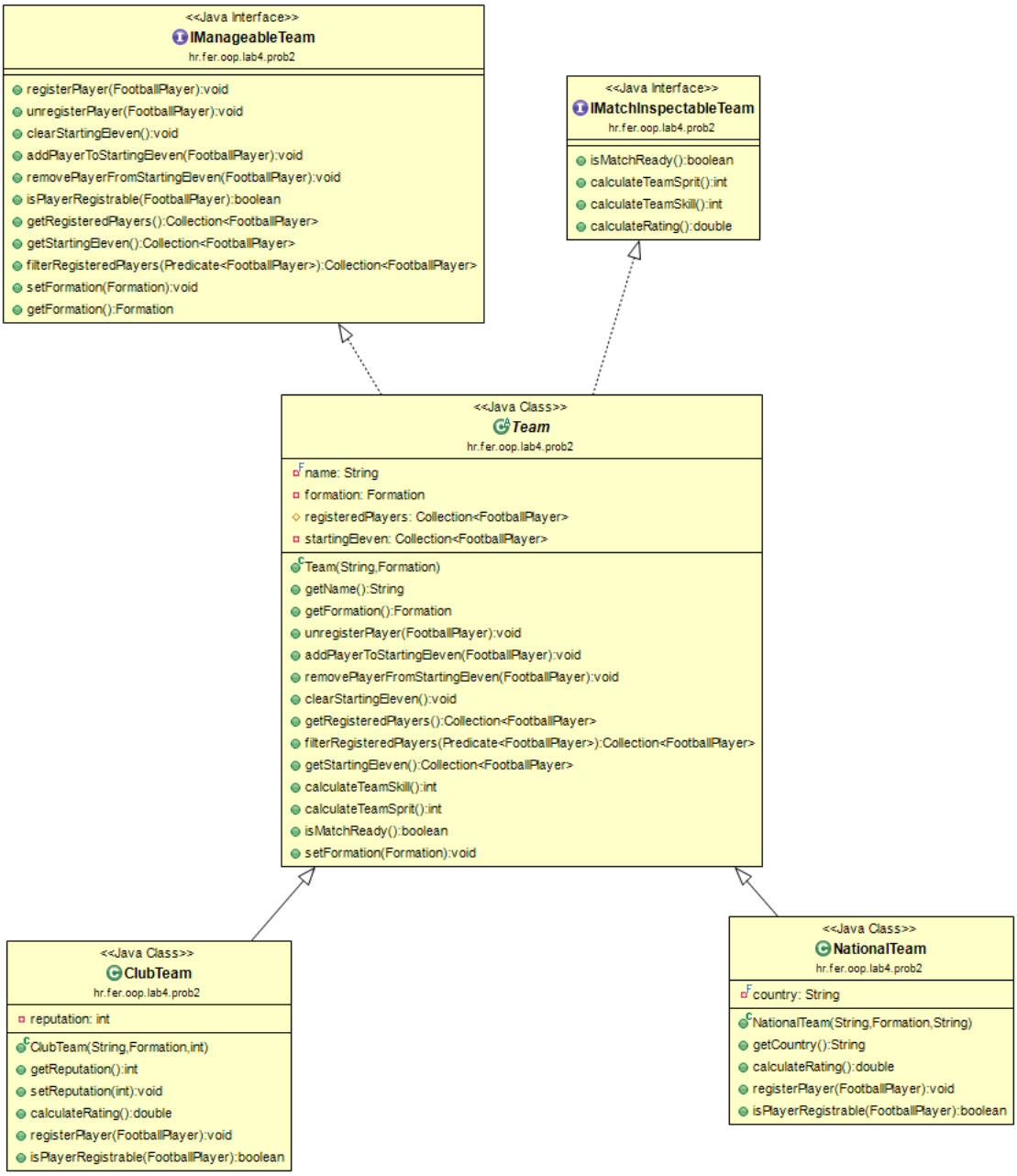
- Uvjeti za igrivu utakmicu:
  - Timovi su postavljeni (`home` i `away`), postavljen je tip utakmice
  - Ako se radi o natjecateljskoj utakmici: ili igraju dva klupska tima ili dva nacionalna tima
  - Timovi su *spremni* za utakmicu (`isMatchReady()` vraća `true`)

Demonstracija: http://pastebin.com/sjrZFYRS

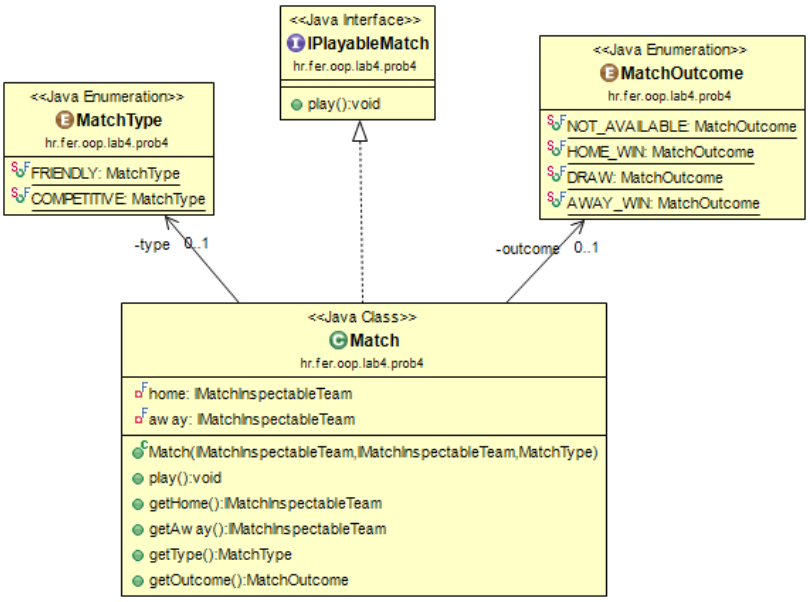
UML dijagrami:



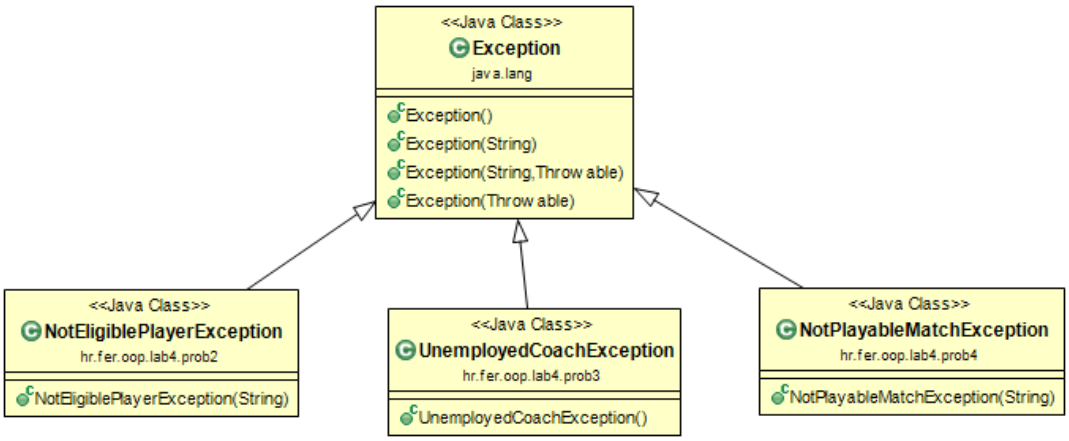
Slika 1: UML dijagram osoba



Slika 2: UML dijagram timova



Slika 3: UML dijagram utakmice



Slika 4: UML dijagram iznimki