2. laboratorijska vježba

Važna napomena:

*„u svim zadacima **potrebno je napisati Javadoc komentare** za svaki razred, sučelje i enumeraciju te **generirati dokumentaciju**“.*

Naučite pisati pravilan kod u skladu s inženjerskom praksom:

Svi nazivi razreda, metoda i varijabli i slično moraju biti na engleskom jeziku. 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.

O laboratorijskoj vježbi:

U okviru ove laboratorijske vježbe potrebno je modelirati osnovne entitete vezane uz simulaciju upravljanje nogometnim timom. Za potrebe rješavanja laboratorijske vježbe pripremili smo Maven projekt:

1. On sadrži sučelja, enumeracija i razreda s konstantama (paket `hr.fer.oop.lab2.welcomepack`) – sve ovo morate koristiti u implementaciji svog rješenja;
2. Preporučujemo pregledavanje dokumentacije Javadoc za artefakte iz točke 1;
3. Implementaciju rješenja radite unutar ovog Maven projekta (paket `hr.fer.oop.lab2`).
4. Maven projekt preuzmite sa stranice kolegija pod laboratorijskim vježbama.
5. Pokretanjem unaprijed priređenih „testnih razreda“ svaki student može vidjeti je li njegova implementacija zadovoljavajuća. Dodatno na analogan način svaki student treba upotpuniti testiranje same implementacije.

S obzirom da se zadatci nadovezuju, savjetujemo da ih rješavate po redu. **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. Nakon uvoza dobivenih datoteka u nekima će se pojaviti greške. Iste se javljaju jer nedostaju implementacije određenih sučelja. Gdje je prikladno, generirajte odgovarajuće *gettere* i *settere*.

Zadatak 1.: Osoba – Trener i Nogometaš

Nogomet treniraju i igraju osobe. Apstraktna osoba `Person` ima ime `name` (pretpostavljena vrijednost: „John Doe“), državu `country` (pretpostavljena vrijednost: „Noland“) i emociju `emotion` (pretpostavljena vrijednost: 50) koja je cijeli broj od 0 do 100. Pretpostavimo da su dvije osobe jednake ako imaju jednako ime i državu.

Trener `Coach` je osoba koja ima vještinu `coachingSkill` (cijeli broj od 0 do 100, uz pretpostavljenu vrijednost: 50) i omiljenu formaciju `formation` (enumeracija `Formation` sa sljedećim vrijednostima[1]: `F442`, `F352`, `F541`; uz pretpostavljenu vrijednost: `F442`).

Nogometaš `FootballPlayer` je osoba koja ima vještinu `playingSkill` (cijeli broj od 0 do 100, uz pretspostavljenu vrijednost: 50) i prirodnu poziciju `playingPosition` koju igra (enumeracija `PlayingPosition` s vrijednostima `FW`, `MF`, `DF` i `GK` koje označavaju napad, sredinu, obranu i vratarsku poziciju; uz pretpostavljenu vrijednost: `MF`).

Trener se, baš kao i nogometaš, može u potpunosti inicijalizirati konstruktorom. Ime i država im se nakon toga ne mogu promijeniti dok se sve ostalo može. Osim toga, moguć je i pretpostavljeni konstruktor. Sve članske varijable potrebno je ispravno postavljati (tj. ne smiju biti `null`), a vrijednosti moraju biti u odgovarajućem rasponu. Inače se opis pogreške (npr. „Igraceva vjestina ne moze biti veca od 100!“) ispisuje na standardni izlaz za pogrešku `System.err` (npr. ako je reputacija izvan traženog raspona).

Zadatak 2.: Jednostavna kolekcija nogometaša

Prije nego li implementiramo nogometni tim, potrebna nam je struktura podataka unutar koje ćemo spremati nogometaše. S obzirom da još nismo učili kolekcije dostupne u Javi te činjenice da znamo raditi s poljem, implementirat ćemo jednostavnu kolekciju nogometaša `SimpleFootballPlayerCollectionImpl`. Ona mora implementirati sučelje `SimpleFootballPlayerCollection`:

`public interface SimpleFootballPlayerCollection`

Specifikacije jednostavne strukture podataka unutar koje se spremaju igrači. Implementacija ovog sučelja, uz varijable koje smatrate potrebnima (npr. int size), kao člansku varijablu ima polje.

Povratna vrijednost	Metode i njihov opis
`boolean`	`add(FootballPlayer player)` Dodaje igrača u kolekciju ako već nije prije dodan te ako ima mjesta u kolekciji.

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

`int`	`calculateEmotionSum()` Izračun ukupne emocije svih igrača u kolekciji.
`int`	`calculateSkillSum()` Izračun ukupne vještine svih igrača u kolekciji.
`void`	`clear()` Briše sve igrače iz kolekcije.
`boolean`	`contains(FootballPlayer player)` Provjera nalazi li se igrač u kolekciji.
`int`	`getMaxSize()` Najveći mogući broj igrača kolekcije, npr. 20
`FootballPlayer[]`	`getPlayers()` Metoda pomoću koje je moguće doći do pozadinskog polja unutar kojeg se zapravo spremaju igrači.
`int`	`size()` Provjera broja igrača koji se trenutno nalaze u kolekciji.

Zadatak 3.: Tim – Nacionalni i Klupski

Apstraktni nogometni tim `Team` ima svoj naziv `name` (pretpostavljena vrijednost: „`TeamX`"), formaciju `formation` (pretpostavljena vrijednost: `F442`), kolekciju registriranih igrača `registeredPlayers` te kolekciju igrača `startingEleven` (veličine 11) 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****). Pogledati dokumentaciju metode* `add(FootballPlayer player)` ***sučelja*** `SimpleFootballPlayerCollection`*.*

Nacionalni tim `NationalTeam` je nogometni tim koji ima državu `country` (tipa `String`, pretpostavljena vrijednost: „`Noland`") te ukupno može imati do 23 registriranih igrača. *Naravno, registrirani igrači i nacionalni tim moraju imati istu državu*. Nacionalni tim se inicijalizira konstruktorom koji postavlja ime, formaciju i državu ili pretpostavljenim konstruktorom. 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, pretpostavljena vrijednost: 50) te ukupno može imati do 25 registriranih 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 ili pretpostavljenim konstruktorom. Naziv se nakon inicijalizacije ne može mijenjati.

Sve članske varijable potrebno je ispravno postaviti (npr. ne smiju biti `null`) i u odgovarajućem rasponu. Inače se opis pogreške (npr. „Formacija ne smije biti null!“) ispisuje na standardni izlaz za pogrešku `System.err` (npr. ako se želi postaviti formacija `null` ili ako je reputacija izvan traženog raspona).

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

`public interface ManageableTeam`

Timom se može upravljati na način kako je definirano ovim sučeljem.

Povratna vrijednost	Metode i njihov opis
`boolean`	`addPlayerToStartingEleven (FootballPlayer player)` Dodaje igrača u početnu jedanaestoricu ako se igrač nalazi u kolekciji registriranih igrača te ima mjesta u prvih 11. Vraća true ako je igrač dodan, inače vraća false.
`double`	`calculateRating()` Računa i vraća izračunatu ocjenu tima koja je definirana kako slijedi: Klupski tim: 70% zbroj vještina registriranih igrača + 30% zbroj emocija registriranih igrača; Nacionalni tim: 30% zbroj vještina registriranih igrača + 70% zbroj emocija registriranih igrača.
`void`	`clearStartingEleven()` Briše prvu jedanaestoricu.
`Formation`	`getFormation()` Dohvat formacije.
`SimpleFootballPlayerCollection`	`getRegisteredPlayers()` Dohvat kolekcije registriranih igrača.
`SimpleFootballPlayerCollection`	`getStartingEleven()` Dohvat kolekcije prve jedanaestorice.
`boolean`	`isPlayerRegistered (FootballPlayer player)`

	Provjerava je li igrač registriran u tim.
`boolean`	`registerPlayer(FootballPlayer player)` Ako se radi o nacionalnom timu igrač mora imati istu državu kao i nacionalni tim (country) dok ako se radi o klupskom timu igrač mora imati vještinu veću ili jednaku reputaciji kluba (u oba slučaja registracija igrača prolazi samo ako ima mjesta).
`void`	`setFormation(Formation formation)` Postavlja formaciju ako ona nije null.

Zadatak 4.: Upravljanje timom

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

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

`public interface Manager` Trener zna poslove upravljanja timom koji su definirani ovim sučeljem.	
Povratna vrijednost	**Metode i njihov opis**
`void`	`forceMyFormation()` Trener pomoću ove metode timu postavlja svoju omiljenu formaciju.
`void`	`pickStartingEleven()` Služi za odabir početne jedanaestorice iz kolekcije registriranih igrača tima kojim trener upravlja; metoda bi trebala paziti da odabrana jedanaestorica mogu činiti formaciju tima (npr. broj napadača u početnoj jedanaestorici odgovara broju napadača u formaciji tima).
`void`	`setManagingTeam(ManageableTeam team)` Putem ove metode, treneru se postavlja tim koji vodi (ako tim nije null).

Zadatak 5.: Testiranje implementacije

Pripremili smo vam testne razrede uz koje možete testirati vlastite implementacije. Razredi koji sadrže samo jednu `main` metodu pokreću se, te se uz pomoć ispisa otkriva eventualna greška u implementaciji, a ti razredi se nalaze untar paketa `hr.fer.oop.lab2.demo`. **Svaki student je dužan na analogan način nadopuniti postojeće ili stvoriti nove razrede te testirati funkcionalnosti koje nisu pokrivene dobivenim testnim slučajevima.**