PRAVILA ZA IMPLEMENTIRANJE PROJEKTNOG ZADATKA IZ PREDMETA „PROGRAMIRANJE U JAVI“

Svaki student od akademske godine 2024/2025 mora prilikom pristupanju ispitu implementirati projektni zadatak dodijeljen tijekom zimskog semestra. Zadatak se dodjeljuje u dogovoru s profesorom, pri čemu student može profesoru predložiti svoju ideju za projektni zadatak ili zatražiti dodjeljivanje teme od strane profesora.

Zadatak se ocjenjuje u sklopu ispitnog roka koji studenti prijavi, pri čemu student mora dokazati da je sam izradio zadatak. U suprotnom, studentu se ne priznaje izrada projektnog zadatka i dodjeljuje novi zadatak. Rješenje se boduje s maksimalnim brojem bodova koji ovisi o tome na kojem ispitnom roku je zadatak predan na sljedeći način:

- Ako se projektni zadatak uspješno preda tijekom ispitnih rokova u veljači, studentu se dodjeljuje maksimalno 30 bodova.
- Ako se projektni zadatak uspješno preda tijekom ispitnih rokova u lipnju ili srpnju, studentu se dodjeljuje maksimalno 20 bodova.
- Ako se projektni zadatak uspješno preda tijekom ispitnih rokova u rujnu, studentu se dodjeljuje maksimalno 10 bodova.

Svaki projektni zadatak mora imati implementirano sljedeće funkcionalnosti:

1.	Implementiranje klasa koje utjelovljuju entitete korištene u projektnom zadatku. Svaka klasa mora biti smještena u paket s klasama koje imaju zajednička svojstva (npr. entiteti moraju biti u jednom paketu, a glavna klasa za pokretanje aplikacije u drugom paketu).
2.	Korištenje apstraktnih klasa, sučelja, zapisa, zapečaćenih sučelja te „builder pattern“ oblikovnog obrasca kako bi se iskoristile sve objektno orijentirane paradigme programskog jezika Java.
3.	Hvatanje i bacanje iznimaka na svim mjestima u programu gdje se mogu dogoditi. Svaka iznimka se mora logirati korištenjem Logback biblioteke. Osim toga je potrebno kreirati barem dvije označene i dvije neoznačene iznimke te ih bacati i hvatati u programskom kodu aplikacije te logirati korištenjem Logback biblioteke. Klase iznimaka moraju biti smještene u zaseban paket.
4.	Korištenje zbirki iz tipa lista, setova i mapa, uz korištenje lambda izraza za filtriranje i sortiranje svih entiteta u aplikaciji.
5.	Korištenje barem dvije generičke klase u aplikaciji koje su smještene u paket zajedno s entitetima. Jedna klasa mora imati samo jedan parametar, a druga klasa mora imati dva parametra generičkog tipa.
6.	Korištenje tekstualnih datoteka koje učitavaju podatke o korisničkim imenima i lozinkama prilikom prijave korisnika u aplikaciju. Potrebno je koristiti i binarne datoteke kojima se serijaliziraju i deserijaliziraju podaci o obavljenim promjenama podataka u projektnom zadatku (na primjer, nakon unošenja novih podataka te promjene postojećih).
7.	Implementirati JavaFX ekran za prijavu korisnika u aplikaciju koja čita podatke iz tekstualne datoteke o korisničkim imenima i „hashiranim“ lozinkama iz tekstualne datoteke kreirane u šestom koraku. Svaka aplikacija mora imati barem dvije korisničke role.

© Autor: doc.dr.sc. Aleksander Radovan, prof.struč.stud., e-mail: aradovan@tvz.hr

8.	Implementirati JavaFX ekran koji će za svaki entitet omogućavati korištenje funkcionalnosti pretrage i filtriranje podataka (korištenjem tablice TableView), dodavanje novog entiteta, promjene postojećih entiteta te brisanje entiteta. Svaka akcija promjene i brisanja entiteta mora uključivati dodatnu potvrdu korisnika da je suglasan s promjenom ili brisanjem korištenjem JavaFX dijaloga.
9.	Implementirati JavaFX ekran koji će omogućavati prikaz svih promjena koje su obavljene u aplikaciji projektnog zadatka korištenjem serijaliziranih podataka iz šestog koraka. Svaka promjena mora sadržavati podatak koji je promijenjen, staru i novu vrijednost, rolu koja ga je promijenila te datum i vrijeme kad se ta promjena dogodila.
10.	Kreirati bazu podataka koja će sadržavati podatke o svim entitetima koji se koriste u aplikaciji te implementirati klasu koja će implementirati funkcionalnosti kreiranje konekcije s bazom podataka, izvršavanje upita nad bazom podataka, dohvaćanje podataka iz baze podataka te zatvaranje konekcije s bazom podataka.
11.	Korištenjem niti implementirati funkcionalnosti osvježavanja podataka na ekranu aplikacije te konkurentno pristupanje dijeljenom resursu kojem pristupa više niti kroz sinkronizaciju niti (npr. jedna nit ispisuje detalje o posljednje promijenjenom podatku koji dohvaća iz serijalizirane datoteke, a za drugu nit koja sprema nove promjene u serijaliziranu datoteku osigurana je sinkronizaciju s tom prvom niti).
12.	Nijedna klasa ne smije imati više od 200 linija koda. Analiza pomoću SonarQube alata ne smije prikazivati nijedan problem. Javadoc dokumentacija i import naredbe nisu uključene u tih 200 linija koda.
13.	Javadoc dokumentacija iznad klasa i metoda je obvezna.