ZADAĆA 3 – UNIT TESTIRANJE

-Datum objave zadaće 3.01.2019. god. Datum predaje zadaće 7.01.2019. u 8:00:00.

-Prvobitno je planirano da će rok za zadaću biti 48h sati, nakon dodatnog dogovora sa studentima zadaća je objavljena dan ranije. Rok se neće pomjerati.

- Zadaća se radi u parovima.

-Uz zadaću priložiti i dokumentaciju o urađenom i naznačiti šta je radio koji član tima.

- Zadaća se sastoji iz više zadataka. Sve zadatke je potrebno kreirati u jednom *solution*-u kao neovisne projekte. *Solution* je neophodno nazvati u obliku ***PrezimeIme1PrezimeIme2***.

-Zadaća se predaje preko zamgera. Obratite pažnju da li je dobro uplodovan fajl i u slučaju problema obavijestiti asistenta grupe najkasnije do 9:00:00 istog dana.

Zadatak 1: 6 bodova

Firma „ETF Music“ je krenula u planiranje sistema za kupovinu i razmjenu pjesama. Ipak, prije toga su odlučili da testiraju ranije razvijeno „iTunes“ rješenja koji radi sličnu stvar (kod je u prilogu zadaće, prilagodite ga vašoj verziji razvojnog okruženja). Osnovne operacije koje to rješenje nudi je registracija različitih vrsta članova, kupovina i razmjena pjesama. Nakon što dobro ispitate način funkcionisanja rješenja, potrebno je uraditi sljedeće:

a) Napisati ***unit test (ili testove)*** koji će obuhvatiti i detaljno istestirati metodu obračuna cijene pjesme. Obavezno koristiti *CollectionAssert* za 3 testa. **(2 boda)**

b) Napisati ukupno ***10 testnih metoda*** (bez testiranja izuzetaka) za rješenje. **(2 bodova)**

c) Obaviti testiranje ***svih*** izuzetaka u rješenju. **(1 boda)**

d) Napisati ***dvije data-driven testne metode*** koje sadrže podatke za testiranje u *XML* fajlu. **(1 boda)**

Napomena: Pri pisanju Unit testova obratiti pažnju na preporuke date na predavanjima, vježbama i pripremi za ispit, a koje se odnose na pokrivenost kôda, imenovanje testnih metoda i klasa kao i organizacije testnog projekta. (**1 bod**)

Zadatak 2: 3 boda

Pored funkcionalnosti koje nudi „iTunes“ rješenje, firma „ETFMusic“ želi da implementira i mehanizme analiziranja naklonosti kupovine korisnika. Da bi se analizirao način uklapanja tih mehanizama u postojeće rješenje, potrebno je, u TDD (*eng. Test Driven Development*) maniru, razviti scenariji testiranja i napisati testove i odgovarajući kod. . Kratak opis metode je sljedeći:

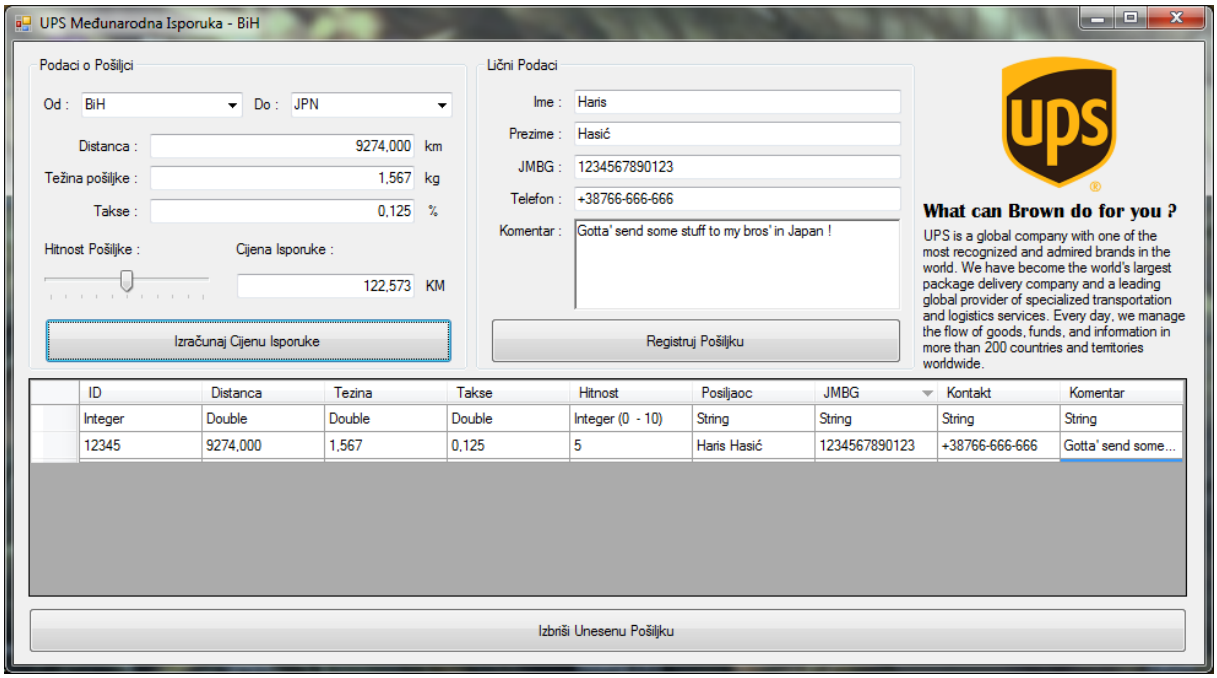
- ***Analiza naklonosti kupovine korisnika*** – Na osnovu odabranog kriterija (umjetnik, album ili žanr) potrebno je proći kroz kolekciju nekog korisnika i utvrditi top 10 listu umjetnika, albuma ili žanrova na koje je taj korisnik potrošio najveću sumu novca.

Potrebno je:

a) U kratkim tezama napisati scenariji testiranja i pisanja koda opisane metode. **(1 bod)**
b) Razviti 3 testne metode koje će u potpunosti testirati navedenu metodu. **(1 boda)**
c) Razviti stub za opisanu metodu koja će simulirati rad iste u svrhu testiranja. **(1 boda)**

Zadatak 3. 5 bodova + 2 boda bonusa

a) Razviti programsko rješenje za poštansku firmu UPS sljedećeg izgleda. Jedan član tima razvija metodu drugi piše test (ektremno programiranje).



Cijena isporuke se računa preko sljedećih formula :

$$\text{Cijena bez takse} = \frac{Distanca}{95{,}643} + \text{Težina Pošiljke} * ((1 + \text{Hitnost} * 0{,}1) * \text{Distanca}) * 0{,}00055$$

$$\text{Konačna cijena} = \text{Cijena bez takse} + (\text{Cijena bez takse} * \text{taksa}(\%))$$

Dugme ***Izračunaj Cijenu Isporuke*** izračunava ukupnu cijenu dostave paketa i u polje cijena isporuke upisuje rezultata u ovisnosti od podataka koji su uneseni u preostalim poljima. U slučaju da nedostaje neki od parametara potrebnih za izračunavanje cijene, a unese se neka željena cijena u spomenuto polje, nedostajući parametar se izračunava i upisuje u prazno polje nakon klika na dugme. To važi za parametre Težina pošiljke, takse, hitnost i cijena isporuke. Ako je sa unesenim parametrima nemoguće izračunati nedostajući (npr. hitnost jer ide u rasponu od 0 - 10) treba prijaviti grešku.

Nakon što se izvrši izračun svih podataka, omogućiti unos *validinih* ličnih podataka za korisnika usluga UPS kompanije. Nakon klika na dugme ***Registruj Pošiljku*** pošiljka se registruje i dodaje se u listu pošiljki koja je prikazana ispod. Odabirom konkretne pošiljke iz tabele te klikom na dugme ***Izbriši Unesenu Pošiljku*** se briše izabrana pošiljka. Perzistentnost podataka riješiti preko XML fajla.

Kreiranje i modeliranje klasa potrebnih da ovo programsko rješenje ispravno funkcioniše je ostavljeno studentima na vlastitu procjenu, ali krajnji rezultat mora zadovoljavati određeni kriteriji složenosti i mora poštovati principe objektno orijentisanog dizajna.

b) Za programsko rješenje je potrebno razviti kod i napisati *unit* testove (jedan član piše kod drugi test) koji će testirati očekivano i neočekivano ponašanje programskih modula (tj. biznis logike, odnosno ispravnost klasa i metoda koje se pozivaju kao rezultat interakcije sa formom, logike upisa u bazu podataka što je u našem slučaju *.xml* fajl i svega ostalog što predstavlja suštinu programskog rješenja.). Minimalna pokrivenost kôda (*code coverage)* iznosi 80%. Posebno dobro obratiti pažnju na pravilno imenovanje testnih klasa i testnih metoda.

(a+b 2 boda)

c) Proširiti rješenje iz Zadatka 1 tako što će se dodati sljedeće funkcionalnosti :

- ***Evidentiranje pošiljki*** – Vanjski servis koji će omogućiti spašavanje podataka o odobrenoj pošiljci.
- ***Konverzija valuta*** – Vanjski servis koji će omogućiti pretvaranje iznosa u KM u neku drugu valutu po vašem izboru.

Kako ovi vanjski servisi još uvijek nisu dostupni za korištenje potrebno je koristiti *Mock* i *Stub* objekte da bi se izvršilo uspješno testiranje rješenja. **(2 boda)**

d) Upotrebom *Coded UI* alata u *Visual Studio*-u razviti 3 UI testa koji će potvrditi ispravan rad funkcionalnosti koje se odnose na korisnički interfejs (UI) rješenja. To se u konkretnom slučaju može odnositi na validaciju, ispis rezultata, proračun nedostajućeg parametra itd.

(1 bod)

Napomena

Pri implementaciji rješenja potrebno je obratiti pažnju na dobru definisanost klasa i odnosa između klasa. Od presudnog značaja za kvantitet i kvalitet vaših napisanih unit testova jeste da UI sadrži samo prezentacijsku logiku dok biznis logiku treba smjestiti smjestiti unutar klasa. Imajte uvijek na umu da je ono što testiramo unit testovima jesu KLASE I ISPRAVNOST REZULTATA ODGOVARAJUĆIH METODA KOJE RADE U POZADINI, a ne forme i pratećeg kôda koja omogućava rad iste.

Primjedbe tipa *„Ova formula za preračunavanje cijene nema smisla, to se računa na sljedeći način...“* i slično se ne uzimaju u razmatranje jer iste služe da studenti imaju materijala za pisanje *unit* testova.