## Univerzitet u Kragujevcu

## Fakultet tehničkih nauka u Čačku

## Drugi kolokvijum iz TESTIRANJA SOFTVERA- I grupa

Kreirati Java projekat u Eclipse radnom okuženje. U okviru paketa **stan** ubaciti 2 klase: Stan.java i AgencijaProdajeStanova.java (kopirati linije koda iz ovog dokumenta sa druge strane).

Za sledeće klase kreirati JUnit testove:

* **Stan.java**

Testirati klasu jediničnim testovima tako da pokrivenost testova na nivou klase bude 100%.

Kod metoda koje bacaju izizetak:

* + setCena()
  + setKvadratura()

Kreirati testove za izuzetke (13)

* **AgencijaProdajeStanova.java**

Kreirati za metode unutar klase parametrizovane testove:

* + Pokrivenost metode primenom parametrizovanih testova treba da bude 100%
  + Minimalan broj parametara za test je 4
  + Primeniti anotacije @BeforeEach i @AfterEach

Svi testovi primenom pravila:

* Pokreću se samo na Windows operativnom sistemu
* Moraju imati naziv metode i klase sa sufiksom ili prefiksom “Test”
* Vreme izvršavanja po testu ne sme biti duže od 5 sekundi. (10)

Testove kreirane za klasu Stan.java objediniti u jednu kolekciju testova, dok parametrizovane testove za klasu AgencijaProdajeStanova.java objediniti u drugu kolekciju testova. Obe kolekcije testova pokrenuti primenom TestRunner-a. U TestRunner-u primenom ***Logger*** klase logovati sledeće podatke:

* Opis svih grešaka
* Vreme i broj izvršenih testova
* Da li su uspešno izvršeni svi testovi ili ne (7)

Predmetni nastavnik

Dr Olga Ristić, docent

**//Stan.java**

**public** **class** Stan {

**private** String adresa = **null**;

**private** **int** kvadratura = 0;

**private** **int** cenaPoKvadratu = 0;

**public** Stan(String adresa, **int** kvadratura, **int** cenaPoKvadratu) {

**super**();

**this**.adresa = adresa;

**this**.kvadratura = kvadratura;

**this**.cenaPoKvadratu = cenaPoKvadratu;

}

**public** **int** getKvdratura() {

**return** kvadratura;

}

**public** **void** setKvadratura(**int** kvadratura) {

**if** (kvadratura < 0)

**throw** **new** RuntimeException("Kvadratura ne sme biti manja od 0");

**this**.kvadratura = kvadratura;

}

**public** **int** getCenaPoKvadratu() {

**return** cenaPoKvadratu;

}

**public** **void** setCenaPoKvadratu(**int** cenaPoKvadratu) {

**if** (cenaPoKvadratu < 0 && cenaPoKvadratu > 0)

**throw** **new** RuntimeException("Cena mora biti veca od 0 i manja od 5000");

**this**.cenaPoKvadratu = cenaPoKvadratu;

}

**public** String getAdresa() {

**return** adresa;

}

**public** **void** setAdresa(String adresa) {

**if** (adresa == **null**)

**throw** **new** RuntimeException("Morate uneti adresu");

**this**.adresa = adresa;

}

**public** **double** izracunajCenu() {

**return** (**this**.kvadratura \* **this**.cenaPoKvadratu);

}

**public** **boolean** povoljanStan() {

**if** (**this**.izracunajCenu() <= 50.000)

**return** **true**;

**else**

**return** **false**;

}

@Override

**public** String toString() {

**return** "Stan [adresa=" + adresa + ", kvadratura=" + kvadratura + ", cena po kvadratu=" + cenaPoKvadratu + "]";

}

}

**//** AgencijaProdajeStanova**.java**

**import** java.util.LinkedList;

**public** **class** AgencijaProdajeStanova {

**public** **static** LinkedList<Stan> *stan* = **new** LinkedList<Stan>();

**public** **static** **void** dodajStan(Stan a) {

**if** (a == **null**)

**throw** **new** NullPointerException("Stan ne sme biti null");

**if** (*stan*.contains(a))

**throw** **new** RuntimeException("Stan vec postoji");

*stan*.add(a);

}

**public** **static** LinkedList<Stan> pronadjiStan(String adresa) {

**if** (adresa == **null**)

**return** **null**;

LinkedList<Stan> novaLista = **new** LinkedList<Stan>();

**for** (**int** i = 0; i < *stan*.size(); i++)

**if** (*stan*.get(i).getAdresa().equals(adresa))

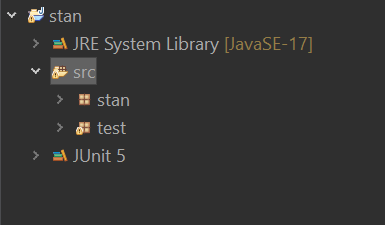
novaLista.add(*stan*.get(i));

**return** novaLista;

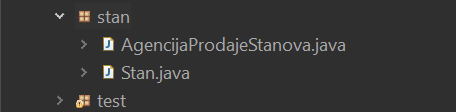
}

}

1. Prvo u eclipse morate da kreirate nov projekat, posle toga idete na src desni klik da kreirate 2 nova paketa(Package, desni klik na src>New>Package), jedan za klase jedan za testove.



2. Posle toga u prvi Package(u ovom slucaju stan), ubacujete klase iz vorda koje su vam date. Pazite da kada kreirate nove klase(desni klik na package>New>Calss)da ih nazovete isto kao sto su nazvane u wordu(u ovom slucaju Stan,AgencijaProdajeStanova).

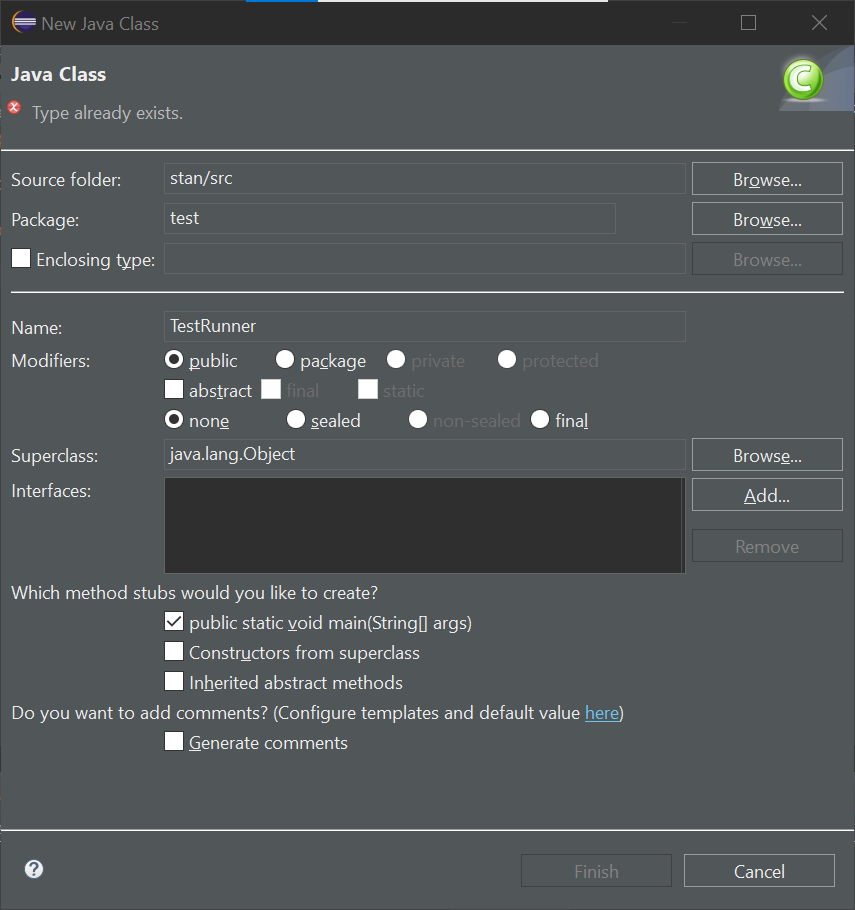


Napomena!

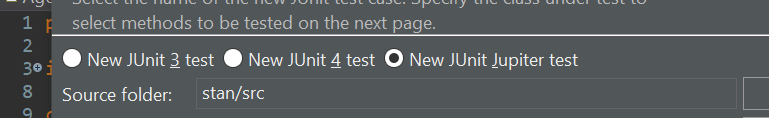
U sustii se ovde dosta stvari uci na pamet, veoma je bitno da sve sto budete pisali pisete pravilno kako bi ste mogli da importujete to u import zaglavlje klasa. Ako se nesto crveni a ne daje vam opciju za import niste lepo napisali rec. Imaj te ovo na pameti u natavku.

3. Sada kreirate clase testova(desni klik na package test>New>JUnit Test Case). Potrebno je kreirati jednu klasu za sve jedinicne testove, vezane za klasu stan, i dve posebne klase za testiranje metoda u AgencijaPronadjiStan(u jednoj cete da testirate metodu dodajStan, a u drugoj pronadjiStan).

Takodje je potrebno napraviti klasu **TestRunner**(package>New>Class)(kada udjete u telo za kreiranje cekirajte opciju public static void main(String[] args)).

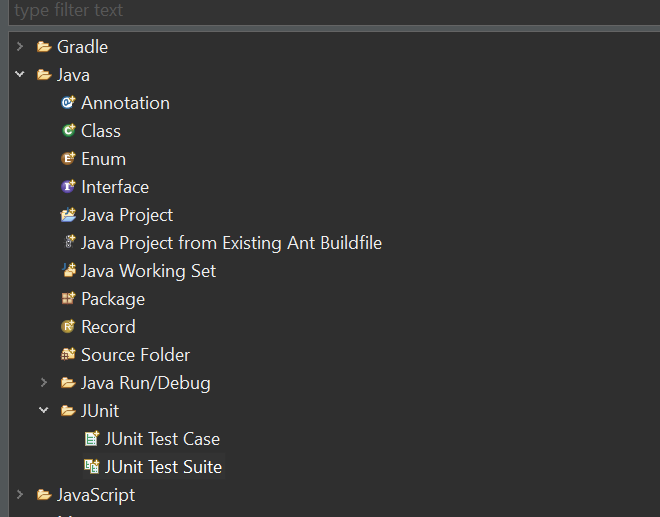


Bitna stvar je da kata kreirate JUnit Test Case cekirate verziju Junita

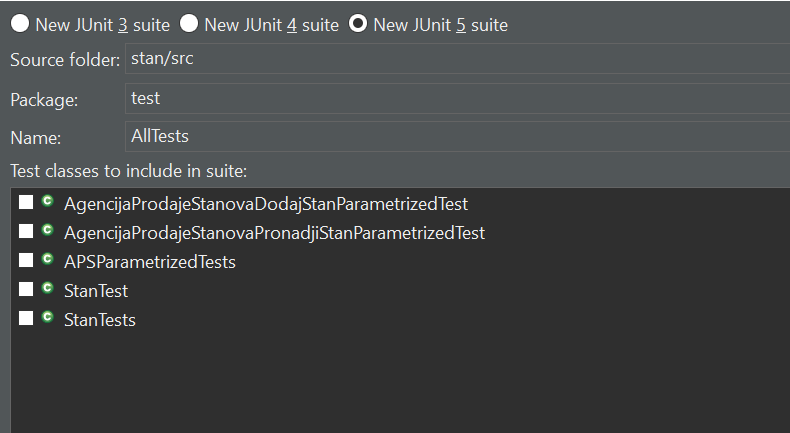


Ja mislim da sam ja koristio JUnit 4, ali to proverite sa stanicem kad budete radili. Zapamtite da samo ako ste izabrali JUnit 4 to izaberete I za sve ostale testove, kao I za test suitove.

Nakon toga potrebno je krirati testsuitove (dva, jedan za jedinicne testove, drugi za parametrizovane).Desni lik na package>New>Other>Java>JUnit>JUnit Test Suite

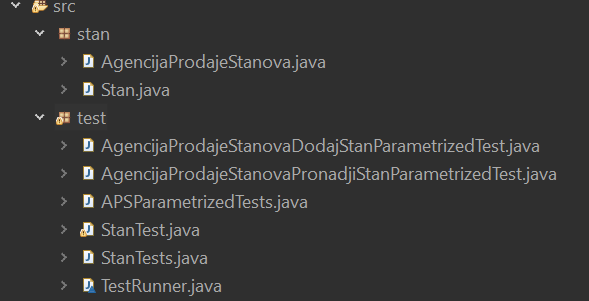


Kada kreirate test suite otvara se sledeci prozor.



Kako klase testova kreiramo kao NazivKlaseTest, test suite bi trebalo nazvati NazivKlaseTests. U prvom slucaju kada kreiramo test suite za jedinicne testova selektujemo samo klasu gde se oni nalaze(u obom slucaju StanTest), kada budete kreirali suite test za parametrizovane selektovacete obe klase gde se nalaze ti testovi(u ovom slucaju AgencijaProdajeStanovaDodajStanParametrizedTest I AgencijaProdajeStanovaPronadjiStanParametrizedTest.

Kada to sve kreirate to treba ovako da izgleda.



**4. Jedinicni tstovi**

Sto se tice jedinicnih testova rade se testovi za metode svih getera I setera, sve metode koje se desavaju u navedenoj klasi I uglavnom test toString metode.

Na pocetku testa treba kreirati sledece

public Stan S;

*@Rule*

public final TestRule timeout = Timeout.*seconds*(5);

*@BeforeClass*

public static void tProveriOperativniSistem() {

Assume.*assumeTrue*(System.*getProperty*("os.name").contains("Windows"));

}

*@Rule*

public final ErrorCollector ec = new ErrorCollector();

*@Before*

public void init() {

S = new Stan("Nemanjina",150,100);

}

*@Rule*

public final TestName name = new TestName();

*@Rule*

public ExpectedException exception = ExpectedException.~~none~~();

Ove gore linije se uce na pamet I uvek se ovao pisu, jedino sto se menja su linije

Public Stan s(tu kreirate instancu klase koju zeite da testirate) I

Public void init(ovde kreiranom objektu dodeljujemo vrednosti, ako niste sigurni koje vrednosti su potrebne kao parametri pogledajte konstruktor u klasi koju trstirate).

GeterTestovi su svi u sustini isti. Potrebno je proveriti njihovo ime I njihovu funkcionalnost.

*@Test*

public void getKvadraturaTest() {

try {

*assertEquals*("getKvadraturaTest",name.getMethodName());

}catch(Throwable t) {

ec.addError(t);

}

int or = 150;

try {

int sr = S.getKvdratura();

*assertEquals*(or,sr);

}catch(Throwable t) {

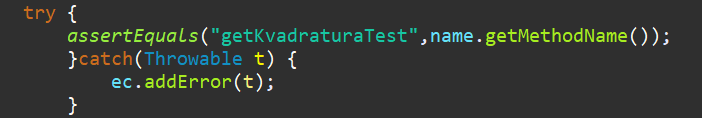
ec.addError(t);

}

}

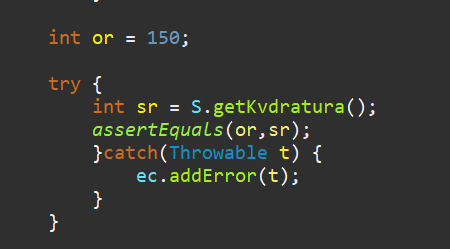
Ovo je primer testa za metodu getKavadratura, kao sto vidite nazivu metode dodata je rec test, iznad deklaracije testa bitno je da stoji @Test, da bi program to razumeo kao test.

Prvo proveramao ime testa



U medodu assertEquals prepisete ime testa pod navodnicima, a kao drugi parametar prosledite name.getMethodName(), kao sto vidite na slici, I to se smesta u try deo, u catch deo se uvek pise ovako.

Posle toga treba testirati geter



Prvo kreiramo neku vrednost kao sto je gore dat primer sa int or = 150; posle toga u try delu deklarisemo sr = S.getKvadratra(), i onda pozivamo metodu assertEquals(or,sr). catch deo je kao sto sam rekao isti. Bitno je obratiti paznju da se u vrednost or upisuje vrednost koja je deklarisana ranije u kodu u liniji:

S = new Stan("Nemanjina",150,100);

Tako se rade I ostali geter testovu, trebalo bi da mozete da prekopirate celo telo tresa I da izmenite potrebno.

**Primeri ostalih geter testova za ovaj konkretan primer:**

*@Test*

public void getCenaPoKvadratuTest() {

try {

*assertEquals*("getCenaPoKvadratuTest",name.getMethodName());

} catch (Throwable t) {

ec.addError(t);

}

int ocekivaniRezultat = 100;

try {

int stvarniRezultat = S.getCenaPoKvadratu();

*assertEquals*(ocekivaniRezultat,stvarniRezultat);

} catch (Throwable t) {

ec.addError(t);

}

}

*@Test*

public void getAdresaTest() {

try {

*assertEquals*("getAdresaTest",name.getMethodName());

} catch (Throwable t) {

ec.addError(t);

}

String ocekivaniRezultat = "Nemanjina";

try {

String stvarniRezultat = S.getAdresa();

*assertEquals*(ocekivaniRezultat,stvarniRezultat);

}catch (Throwable t) {

ec.addError(t);

}

}

**SeterTestovi**

Kod seter testova ne moramo testirati ime testa, ali moramo da proverimo da li test u sebi ima neki izuzetak. Na primer vrednost parametra kvadraturaStana ne sme biti <0.

*@Test*(expected = RuntimeException.class)

public void setKvadraturaTest1() {

int or = 150;

int sr = S.getKvdratura();

*assertEquals*(or,sr);

S.setKvadratura(-1);

}

Ovde namerno postavljamo pokusavamo da izazovemo gresnu sa linijom S.setKavdratura(-1).Kako bi test znao da namerno pokusavamo da izazovemo gresku mora se napisati sledece.Iznad naziva testa treba da stoji:

*@Test*(expected = RuntimeException.class)

Tako ce test znati da namerno izazivamo gresku.

Nakon toga testiramo metodu, ali na ispravan nacin:

*@Test*

public void setKvadraturaTest2() {

int ocekivaniRezultat = 150;

int stvarniRezultat = S.getKvdratura();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

S.setKvadratura(100);

ocekivaniRezultat = 100;

stvarniRezultat = S.getKvdratura();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

}

Ovde se iznad naziva pise samo @Test, prvo sto radimo proveramo geter, zasto ni ja ne znam. Posle toga postavljamo novu vrednost I menjamo ocekivani rezultat. Onda ih poredimo.

S.setKvadratura(100);

ocekivaniRezultat = 100;

stvarniRezultat = S.getKvdratura();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

Svi ostali seter testovi se rade slicno, tako da cete verovatno moci ponovo da ih kopirate.

**5. Prieri ostalih seter testova:**

*@Test*(expected = RuntimeException.class)

public void setCenaPoKvadratuTest1() {

int ocekivaniRezultat = 100;

int stvarniRezultat = S.getCenaPoKvadratu();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

S.setCenaPoKvadratu(-1);

}

*@Test*(expected = RuntimeException.class)

public void setCenaPoKvadratuTest2() {

int ocekivaniRezultat = 100;

int stvarniRezultat = S.getCenaPoKvadratu();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

S.setCenaPoKvadratu(5100);

}

*@Test*

public void setCenaPoKvadratuTest3() {

int ocekivaniRezultat = 100;

int stvarniRezultat = S.getCenaPoKvadratu();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

S.setCenaPoKvadratu(2000);

ocekivaniRezultat = 2000;

stvarniRezultat = S.getCenaPoKvadratu();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

}

*@Test*(expected = RuntimeException.class)

public void setAdresaTest1() {

String ocekivaniRezultat = "Nemanjina";

String stvarniRezultat = S.getAdresa();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

S.setAdresa(null);

}

*@Test*

public void setAdresaTest2() {

String ocekivaniRezultat = "Nemanjina";

String stvarniRezultat = S.getAdresa();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

S.setAdresa("Svetog Save");

ocekivaniRezultat = "Svetog Save";

stvarniRezultat = S.getAdresa();

*assertEquals*(ocekivaniRezultat, stvarniRezultat);

}

Sada treba testirati metode koje racunaju nesto, proveravaju(vracaju true ili false) I toString().

Ukratko morate da proverite sve moguce scenatije. Napravite ocekivani rezultat, pozivanjem metode formirate stvaran rezultat I onda ih poredite.

Kada sam rekao da treba da se provere svi slucajevi mislio sam na sledece:

Metoda izracunajCenu samo racuna cenu stana I njoj je potreban jedan test.

Metoda povoljanStan, mora da sadrzi dva testa, jedan koji ci prema uslovu vratiti ture, I drugi koji ce vratiti false (pozivnjem ove metode sa vrenostima koje smo dodelili na pocetku rezultat ovog testa treba da bude true)

Metoda toString potrebno je da podestir ocekivanu rednost bas kao string koji ce ispisati metoda to string, obratite paznju kako bi trebao da izgleda ispis na osnovu metode u klasi.

Ovi testovi trebaju da izgledaju ovako(napomena, ja sam zbog vezbe testirao I imena metoda, vi to ne morate da radite):

*@Test*

public void izracunajCenuTest() {

try {

*assertEquals*("izracunajCenuTest",name.getMethodName());

} catch (Throwable t) {

ec.addError(t);

}

double ocekivaniRezultat = 15000.00;

try {

double stvarniRezultat = S.izracunajCenu();

*assertEquals*(ocekivaniRezultat,stvarniRezultat,0.1);

}catch (Throwable t) {

ec.addError(t);

}

}

*@Test*

public void povoljanStanTest1() {

try {

*assertEquals*("povoljanStanTest1",name.getMethodName());

} catch (Throwable t) {

ec.addError(t);

}

boolean ocekivaniRezultat = true;

try {

boolean stvarniRezultat = S.povoljanStan();

*assertEquals*(ocekivaniRezultat,stvarniRezultat);

}catch (Throwable t) {

ec.addError(t);

}

}

*@Test*

public void povoljanStanTest2() {

try {

*assertEquals*("povoljanStanTest2",name.getMethodName());

} catch (Throwable t) {

ec.addError(t);

}

boolean ocekivaniRezultat = false;

S.setCenaPoKvadratu(4000);

S.setKvadratura(100000);

try {

boolean stvarniRezultat = S.povoljanStan();

*assertEquals*(ocekivaniRezultat,stvarniRezultat);

}catch (Throwable t) {

ec.addError(t);

}

}

*@Test*

public void toStringTest() {

try {

*assertEquals*("toStringTest",name.getMethodName());

} catch (Throwable t) {

ec.addError(t);

}

String ocekivaniRezultat = "Stan [adresa=Nemanjina, kvadratura=150, cena po kvadratu=100]";

try {

String stvarniRezultat = S.toString();

*assertEquals*(ocekivaniRezultat,stvarniRezultat);

}catch (Throwable t) {

ec.addError(t);

}

}

}

Sada treba kreirati parametrizovane testove.Sledeci deo koda se mora nauciti napamet I napisati u oba testa.

private Stan S;

private AgencijaProdajeStanova APS;

*@BeforeClass*

public static void tProveriOperativniSistem() {

Assume.*assumeTrue*(System.*getProperty*("os.name").contains("Windows"));

}

*@Rule*

public final TestRule timeout = Timeout.*seconds*(5);

public AgencijaProdajeStanovaDodajStanParametrizedTest(Stan S) {

this.S = S;

}

*@Parameters*

public static Collection<Object[]> stan(){

return Arrays.*asList*(new Object [][] {

{new Stan("Nemanjina",150,100)},

{new Stan("Nemanjina",150,100)},

{new Stan("Nemanjina",150,100)},

{new Stan("Nemanjina",250,200)}

});

}

*@Before*

public void init() {

APS = new AgencijaProdajeStanova();

}

Sada pisete testove, a posle testova dodate

*@After*

public void destroy() {

APS =null;

}

Podsetnik u delu parameters treba kreirati podatke u formatu klase koju vi imate.

Napomena!

Da bi program znao da radite parametrizoane testove morate da iznad naziva klase dodate liniju kao sto je prikazano ovde

*@RunWith*(Parameterized.class)

public class AgencijaProdajeStanovaDodajStanParametrizedTest {

Sto se samih testova tice ja sam ih ucio na pamet I na tu foru su mi slicni I pali samo sam ih malo izmenjao tako da cu samo da navedem kako izgledaju testovi za ovaj konkretan primer.

**AgencijaProdajeStanovaDodajStanTest**

*@Test*(expected = NullPointerException.class)

public void dodajStanTest() {

Stan S = null;

APS.dodajStan(S);

}

*@Test*(expected = RuntimeException.class)

public void dodajStanTest2() {

APS.dodajStan(S);

APS.dodajStan(S);

}

*@Test*

public void dodajStanTest3() {

*assertFalse*(APS.stan.contains(S));

APS.dodajStan(S);

*assertTrue*(APS.stan.contains(S));

}

**AgencijaProdajeStanovaPronadjiStanTest**

*@Test*

public void pronadjiStanTest1() {

String adresa = null;

APS.pronadjiStan(adresa);

}

*@Test*

public void pronadjiStanTest2() {

*assertFalse*(APS.stan.contains(S));

APS.dodajStan(S);

LinkedList<Stan> stan = new LinkedList<Stan>();

stan.add(S);

*assertEquals*(stan,APS.pronadjiStan("Nemanjina"));

}

*@Test*

public void pronadjiStanTest3() {

*assertFalse*(APS.stan.contains(S));

APS.dodajStan(S);

LinkedList<Stan> stan = new LinkedList<Stan>();

stan.add(S);

*assertNotEquals*(stan,APS.pronadjiStan("Svetog Save - NotExctingModule"));}

**6. TestRunner**

Ovo bukvalno celmo morate da naucite na pamet jer se sve uvek isto pise, jedino sto menjate je naziv test suitova koje ste ranije kreirali.

Ovako izgleda TestRunner za ovaj primer:

class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.*runClasses*(StanTests.class, APSParametrizedTests.class);

Logger l = Logger.*getLogger*(TestRunner.class.toString());

for(Failure f: result.getFailures()) {

l.info(f.toString());

}

l.info("Vreme"+result.getRunTime());

l.info("Ukupno Testova"+result.getRunCount());

l.info("Ukuono Proslih"+(result.getRunCount()-result.getFailureCount()-result.getIgnoreCount()-result.getAssumptionFailureCount()));

l.info("Ukupno palih"+result.getFailureCount());

l.info("Ukupno odbijenih"+result.getIgnoreCount());

l.info("Ukupno dinamicki odbijenih"+result.getAssumptionFailureCount());

if(result.wasSuccessful())

l.info("Svi testovi su prosli");

else

l.warning("Postoji test koji je pao");

}

}

Isti ce ovakav biti I vas samo je potrebno da u liniju

Result result = JUnitCore.*runClasses*(StanTests.class, APSParametrizedTests.class);

U zagradi navedete nazive test suitova koje ste vi dali kao sto je prikazano gore(format je isti NazivTestSuita.class).