**PROIECT CALITATE ȘI TESTARE SOFTWARE**

Nume: Zamfirescu Irina Maria

Grupa: 1054, Seria B, Anul 3

Cuprins

[Definirea și detalierea pattern-urilor folosite 1](#_Toc420422567)

[Definirea și detalierea metodelor testate prin Unit Testing 2](#_Toc420422568)

[Definirea și descrierea Test Case-urilor 3](#_Toc420422569)

[Definirea și descrierea Test Suite-ului 6](#_Toc420422570)

[Descrierea sumară a funcțiilor aplicației cu referire la pattern-uri și la metodele testate 8](#_Toc420422571)

[BIBLIOGRAFIE 9](#_Toc420422572)

# Definirea și detalierea pattern-urilor folosite

Pentru realizarea aplicației am utilizat următoarele pattern-uri:

* Builder;
* Façade;
* Template;
* Singletone.

Clasa Attractions are următoarele atribute: denumire, descriere, latitudine, longitudine, vechime și popular. Pentru a afisa obiectivele turistice într-o listă este nevoie doar de denumirea și descrierea obiectivului, motiv pentru care am folosit ***Builder*** ca și design pattern. Pentru asta am creat interfața IAttractionBuilder care conține metoda *build()* și clasa AttractionBuilder care conține aceleași atribute ca și clasa inițiala Attraction. În plus, clasa AttractionBuilder mai conține câte o metodă pentru fiecare atribut al clasei.

Pentru ca harta să fie afișata pe ecran, se parcurg de fiecare dată aceeași pași, o rutină (***Façade***). În primul rând se inițializează harta, în al doilea rând se setează anumite limite de încadrare a hărții (să fie focalizată doar harta României, nu întregul Glob), apoi se adaugă marcatorii pe hartă (atât marcatorii pentru orașe cât și cei pentru obiectivele turistice), iar în final se pot accesa informații despre fiecare marcator în parte printr-un simplu click.

Metoda ***Template*** este utilizată pentru a standardiza scheletul unui algoritm. Implementarea comună este definită în clasa de bază.

**Singletone** este o tehnică care permite accesul la o singură resursă în orice moment al execuției aplicației. Am folosit Singletone în clasa DataBaseHelper pentru a crea o singură instanță de acest tip în clasa MapActivity.

# Definirea și detalierea metodelor testate prin Unit Testing

Pentru testare am folosit următoarele clase: Attraction, City, DataBaseHelper și MapActivity.

Pentru clasa ***Attraction*** am făcut teste pe următoarele metode:

* constructorul cu toți parametrii – am verificat dacă în constructor este trimis ca parametru pentru descriere un string null sau gol; același lucru l-am verificat și pentru denumire (să nu fie null sau gol);
* constructorul cu un singur parametru – am verificat ca singurul parametru transmis (denumirea) să nu fie null sau gol;
* am testat toate metodele de set să îndeplinească anumite condiții;
* am testat toate metodele de get (ceea ce am setat eu să fie același cu ceea ce returnează metoda set).

Pentru clasa ***DataBaseHelper*** am făcut teste pe următoarele metode:

* metoda getDb()– returnează baza de date;
* metoda checkDataBase() – verifică dacă baza de date există;
* metoda openDataBaseVerify() – verifică dacă baza de date s-a deschis.

Pentru clasa ***MapActivity*** am facut teste pe următoarele metode:

* metoda distanceBeetween(loc1, loc2) – determină distanța dintre două coordonate în kilometrii.

Pentru clasa ***City*** am făcut teste pe următoarele metode:

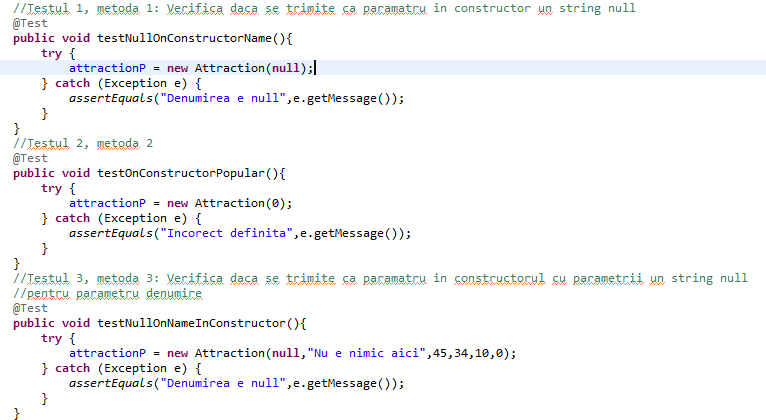
* constructorul cu parametrii;
* metodele de set;
* metodele de get.

# Definirea și descrierea Test Case-urilor

Pentru a face teste pe metodele descrise mai sus am folosit mai multe clase de test*: TestSetAttributes, TestGetAttributes, TestMapActivity, TestConstructors, TestDataBaseHelper*.

**TestConstructors**

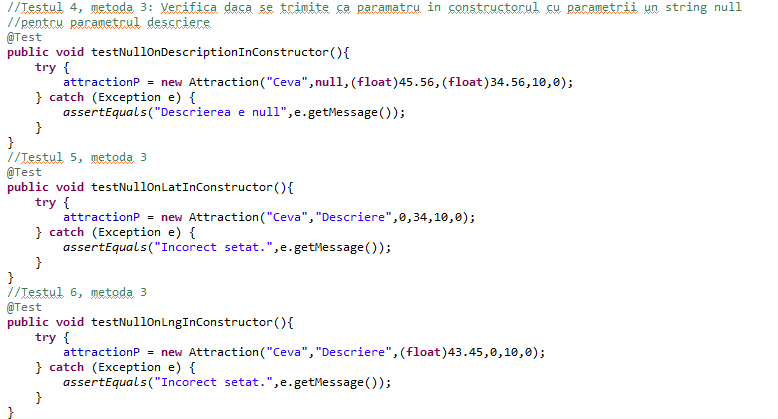
În această clasă de teste am folosit metoda *setUp()* pentru a instanția cele două obiecte din clasa Attraction, respectiv City.



**Testul 1** este folosit pentru a verifica dacă *denumirea* trimisă ca parametru în *constructorul cu un singur parametru* este null sau este un string gol.

**Testul 2** este folosit pentru a se verifica daca câmpul *popular* (care poate lua doar două valori: 1 – popular, 0 – nepopular)transmis ca parametru în *constructorul cu parametrii* este corect definit.

**Testul 3** este folosit pentru a se verifica dacă *denumirea* transmisă ca parametru în *constructorul cu parametrii* este un string null sau gol.

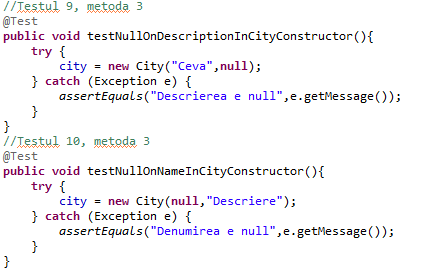


**Testul 4** este folosit pentru a verifica dacă descrierea transmisă ca parametru în constructorul cu parametrii este null. Dacă da, metoda va arunca o excepție.

**Testul 5** este folosit pentru a testa dacă parametrul latitudine este corect setat.

**Testul 6,** ca și testul 5, verifică dacă valoarea pentru parametrul longitudine este corect setat.

**Testul 7** și **Testul 8** verifică dacă parametrii vechime, respectiv popular sunt corect setați (vechimea trebuie să fie mai mare decât 0, iar parametrul *popular* să aibă valoarea 0 sau 1).



**Testul 9** și **Testul 10** verifică, pe rând, cei doi parametrii din clasa City. Aceștia trebuie să fie diferiți de valoarea null, altfel metodele vor arunca excepția nou creată.

**TestSetAttributes**

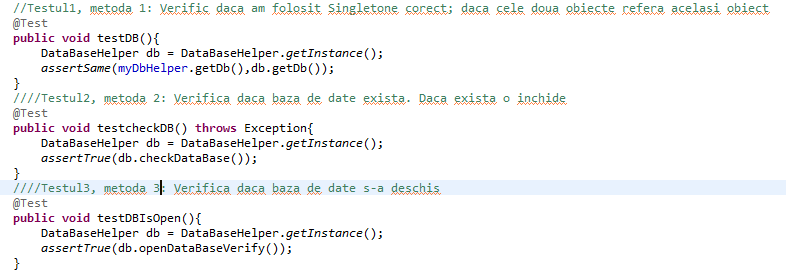
Această clasă de teste conține metoda *setup()* în care se instanțiază două obiecte ale clasei Attraction, respectiv City.

Toate testele din această clasă verifică ca parametrii setați pentru orice obiect să fie setați corect (denumirea și descrierea obiectivului turistic să nu fie null, latitudinea să fie cuprinsă între 1 și 90, longitudinea să fie cuprinsă între 1 și 180, vechimea să fie mai mare decât 0, iar câmpul popular să fie 0 – atracție turistică nepopulară sau 1- atracție turistică populară).

**TestGetAttributes**

Această clasă de teste conține metoda *setup()* în care se fac initializări. Toate testele din această clasă verifică dacă valoarea setată pentru obiect cu *set* este egală cu valoarea returnată de *get*. S-au făcut teste pentru toate atributele din clasa Attraction, respectiv clasa City.

**TestDataBaseHelper**



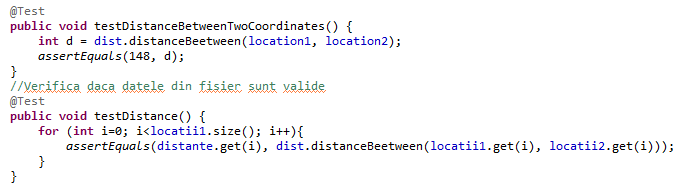
**Testul 1** este folosit pentru a testa dacă design pattern-ul folosit (Singletone) a fost bine implementat. Dacă da, două instanțe ale aceleiași clase (DataBaseHelper) trebuie să pointeze către același obiect (să aibe același id).

**Testul 2** verifică dacă baza de date există.

**Testul 3** verifică dacă baza de date s-a deschis. Dacă da, returnează true, altfel false.

**TestMapActivity**

Această clasă de teste are implementate metodele *setUp()* și *tearDown().* În metoda setUp() se află citirea din fișier (citesc linie cu linie din fișier și creez 3 liste: o listă pentru coordonatele primei locații, o listă pentru coordonatele celei de a doua locații și o listă pentru distanța dintre cele două locații).



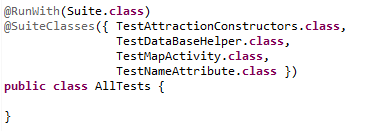
Testul *testDistance()*verifică dacă datele din fișier sunt valide (dacă distanța dintre cele două coordonate coincide cu distanța dată la sfârșitul fiecărei linii din fișier). O linie conține: 2 coordonate pentru punctul A (prima locație), 2 coordonate pentru punctul B (a doua locație) și o valoare întreagă care reprezintă distanța așteptată dintre A și B.

Metoda *tearDown()* conține închiderea fișierului. Testul *testDistanceBetweenTwoCoordinates()* verifică dacă distanța dintre cele două locații inițializate în *setUp()* este aceeași cu cea așteptată.

# Definirea și descrierea Test Suite-ului

Am folosit două clase de tip Test Suite: AllTests și TestSuiteOnNonActivityClass.

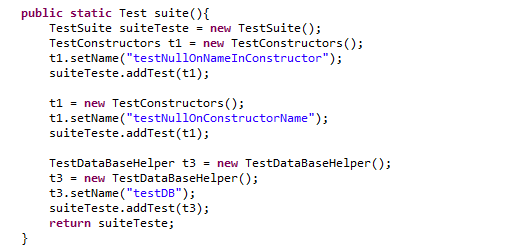
Clasa ***AllTests*** cuprinde toate clasele de teste:



Am folosit această clasă pentru a testa toată testele odată, fără a mai fi necesarp rularea lor separată.

Cealaltă clasă, ***TestSuiteOnNonActivityClass*** conține doar o parte din clasele de teste definite și doar o parte din metodele folosite. Numele este sugestiv, deoarece această clasă testează doar acele metode care fac parte din clase care nu sunt activități. Aici am definit un obiect de tip TestSuite, am creat câte un obiect din fiecare clasă și am adăugat la suita de teste doar acele teste de care aveam nevoie:

* testNullOnNameInConstructor;
* testNullOnConstructorName;
* testDB.



# Descrierea sumară a funcțiilor aplicației cu referire la pattern-uri și la metodele testate

Aplicația este concepută pentru a ajuta turiștii să identifice obiectivele turistice aflate în apropierea locului unde aceștia își petrec vacanța sau excursiile din România. Este o aplicație care prezintă exclusiv locuri din România, precum orașele, bisericile, mânăstirile, parcurile etc.

Aplicația va utiliza o hartă pentru o vizualizare mai bună a obiectivelor, folosind și o legendă care arată tipul obiectivului turistic: mânăstire, parc, oraș, castel etc. Aceasta are in spate o baza de date care stocheaza informatii (denumire, descriere, locatie) cu privire la obiectivele turistice.

Conexiunea la baza de date se face folosind pattern-ul Singletone. Acesta are avantajul că se va crea o nouă instanță pentru baza de date numai atunci când nu există deja una creată. Altfel spus, crearea unei noi instanțe a clasei DataBaseHelper va indica de fapt spre același obiect. Pentru a verifica că acest lucru se întamplă, am folosit testul *testDB()* care verifică prin metoda *assertSame()* că două instanțe ale aceleiași clase ocupă același spațiu în memorie și au același id.

Clasa Attraction are mai multe atribute (unele din ele sunt folosite, altele nu). Acesta este motivul pentru care am folosit pattern-ul Builder.

# BIBLIOGRAFIE

* **acs.ase.ro**
* **seminarii**