**Tehnicile de creare a unui cod de programare calitativ**

Java este unul dintre cele mai utilizate limbaje de programare. În vremurile actuale de progrese rapide când multe alte limbaje de programare care în trecut au avut o prezență puternică au dispărut, Java este încă foarte utilizat și crește rapid odată cu tendințele. Conform statisticilor furnizate de Oracle, din decembrie 2019, Java a fost clasat pe primul loc în top-ul limbajelor de programare pentru aplicații recente precum DevOps, Inteligența Artificială, Realitatea Virtuală, Big Data, Mobile și dezvoltarea de microservicii.

Datorită concurenței imense dintre programatorii care cunosc limbajul de programare Java, pentru a ieși în evidență munca noastră trebuie să îndeplinească anumite standarde pe care le putem atinge urmând unele bune practici de codare în Java.

Un cod executabil, fără erori, ar putea părea mai mult decât suficient, însă nu este chiar așa. Un cod trebuie scris urmând setul de reguli și standarde de codare. Prin aplicarea celor mai bune practici de codare Java, putem îmbunătăți drastic calitatea codului nostru, făcându-l mult mai ușor de înțeles atât pentru noi, cât și pentru ceilalți programatori, reutilizabil, flexibil și ușor de întreținut. Unele bune practici Java pot chiar să crească performanța codului, făcându-l semnificativ mai eficient.

În acest Assignment am enumerat și descris câteva din cele mai bune practici Java pe care ar trebui să le urmăm pentru a scrie un cod de programare calitativ.

Mai întâi, înainte de a începe să scriem codul, trebuie să setăm o convenție de denumire adecvată pentru proiectul nostru Java. Trebuie să decidem în prealabil numele pentru fiecare clasă, interfață, metode și variabile. Dacă la proiectul nostru lucrează și alți dezvoltatori, aceștia ar trebui să respecte aceste reguli, pentru a menține uniformitatea. O convenție de denumire semnificativă este extrem de importantă, deoarece totul, de la clase la interfețe, este identificat din numele lor în cod.

Nu este recomandabil să atribuim nume aleatorii doar pentru a satisface compilatorul, ci trebuie să utilizăm nume semnificative și care se explică de la sine, astfel încât să fie lizibil și să poată fi înțeles ulterior de către noi, colegii de echipă, inginerii de asigurare a calității și de către cei ce se vor ocupa de întreținerea proiectului.

Este considerată o practică bună Java să menținem accesibilitatea câmpurilor de clasă cât mai inaccesibilă posibil, iar asta în ideea de a proteja câmpurile. Pentru a realiza acest lucru, modificatorul de acces privat este alegerea cea mai bună. Această practică este recomandată pentru a menține încapsularea, unul dintre conceptele fundamentale ale OOP. Acesta este un concept de bază al programării orientate pe obiecte, însă mulți dezvoltatori noi nu sunt conștienți de el și nu atribuie corect modificatorul de acces claselor și preferă să-l mențină public pentru ușurința lor.

Spre exemplu, să luăm în calcul această clasă în care câmpurile sunt făcute publice:

public class Employee{

public String name;

public String department;

}

Încapsularea este compromisă în această situație, deoarece oricine poate schimba aceste valori în felul următor:

Employee E01 = new Employee();

Employee.name = “George”;

Employee.department = “IT”;

Utilizarea modificatorului de acces privat cu membrii clasei va păstra câmpurile ascunse, împiedicând utilizatorul să modifice datele, cu excepția metodelor setter. Spre exemplu, am putea folosi modificatorul de acces privat astfel:

public class Employee{

private String name;

private String department;

public void setName(String name){

this.name = name;

}

public void setDepartment(String department){

this.department = department;

}

}

O altă bună practică pentru crearea unui cod de programare calitativ este utilizarea caracterelor de subliniere (Underscore) “\_” în cazul valorilor numerice lungi. Caracteristica aceasta ajută la scrierea valorilor numerice lungi într-un mod mult mai lizibil. Astfel, în loc să scriem în felul obișnuit astfel:

int dob = 13111985;

Putem scrie acest număr într-o manieră ce respectă buna practică Java astfel:

int dob = 13\_11\_1985;

Adoptarea acestei bune practici Java face ca codul nostru să fie lizibil, bine structurat și unic.

O altă bună practică Java o reprezintă faptul că este recomandabil să nu lăsăm blocul Catch gol. Aceasta este una dintre practicile preferate ale dezvoltatorilor profesioniști, deoarece permite scrierea unui mesaj adecvat și semnificativ în blocul Catch în timpul gestionării excepțiilor. Dezvoltatorii noi lasă adesea acest bloc Catch gol, deoarece inițial sunt singurii care lucrează la cod, însă atunci când excepția este capturată de blocul Catch gol, dacă programul primește o excepție, acesta nu arată nimic, ceea ce face debugging-ul mai dificil și consumă foarte mult timp.

Comentariile adecvate reprezintă o altă bună practică Java de scriere a unui cod de programare calitativ. Datorită faptului că codul nostru va fi citit de diverse persoane cu cunoștințe diferite despre Java, comentariile adecvate ar trebui folosite pentru a oferi o prezentare generală a codului nostru și pentru a oferi informații suplimentare care nu pot fi percepute din codul în sine. Comentariile ar trebui să descrie funcționarea codului nostru pentru a putea fi citite de către inginerul de asigurare a calității, testerul sau personalul care se ocupă cu mentenanța programului, care ar putea să nu aibă suficiente cunoștințe de programare în limbajul de programare Java.

O altă bună practică Java o reprezintă faptul că este recomandabil să evităm inițializarea variabilelor cu valori precum 0, false sau nul. Aceste valori sunt deja valorile implicite de inițializare ale variabilelor membre în Java. Astfel, cea mai bună practică Java este de a fi conștienți de valorile implicite de inițializare ale variabilelor și de a evita inițializarea explicită a acestor variabile cu astfel de valori.

Bucla „For” este utilizată cu o variabilă counter (contor), dar o bună practică Java este folosirea buclei „For” îmbunătățită (enhanced for loop), în loc de cea veche. În general, nu se va face nicio diferență în utilizarea acestora, dar în unele situații, variabila contor utilizată ar putea fi foarte predispusă la erori. Variabila contor poate fi modificată întâmplător, poate fi folosită mai târziu în cod sau se poate începe indexul de la 1 în loc de 0, ceea ce va duce la perturbarea codului în mai multe puncte. Pentru a elimina acest lucru, enhanced for loop este cea mai bună opțiune. De exemplu, în următorul bloc de cod:

String[] names = {“George”, “Andy”, “Mike”};

for (int i = 0; i < names.lenght; i++){

method1(names[i]);

}

Aici variabila “i" este folosită ca și contor, precum și ca index pentru numele array-ului. Acest lucru poate să devină o problemă ulterioară în cod, iar pentru a evita acest lucru vom utiliza o buclă for îmbunătățită astfel:

For (String Name1 : names){

Method1(Name1);

}

De cele mai multe ori, ca dezvoltator fără experiență nu vom face diferența dintre Double și Float. Cea mai bună practică Java este să folosim Float și Double conform cerințelor. Cele mai bune practici nu numai că îmbunătățesc lizibilitatea și înțelegerea codului, dar îmbunătățesc chiar și performanța acestuia. Majoritatea proceselor necesită aproape același timp în procesarea operațiunilor pe Float și Double, doar că Double oferă mult mai multă precizie decât Float, iar de aceea cea mai bună practică este să folosim Double atunci când precizia este importantă. Altfel, putem alege Float, deoarece necesită jumătate de spațiu de memorie decât Double.

Crearea obiectelor este una dintre operațiunile care consumă cel mai mult memorie, de aceea cea mai bună practică Java este de a evita crearea de obiecte inutile, iar acestea ar trebui să fie create numai atunci când este necesar.

În Java, dezvoltatorii nu au prea mult control asupra gestionării memoriei, deoarece Java gestionează memoria automat. Totuși, există unele dintre cele mai bune practici Java utilizate de dezvoltatorii de top pentru a preveni scurgerile de memorie, deoarece pot cauza o degradare semnificativă a performanței. Eliberarea întotdeauna a conexiunilor la baza de date după ce interogarea noastră a fost finalizată, utilizarea Finally Block cât mai des posibil și eliberarea instanțelor stocate în tabelele statice sunt câteva dintre cele mai bune practici de codare Java pe care trebuie să le adaptăm pentru a preveni scurgerea memoriei.

Manipularea șirurilor este foarte ușoară în Java, dar trebuie utilizată eficient pentru a preveni utilizarea memoriei de acces. De exemplu, dacă două șiruri sunt concatenate într-o buclă, un nou obiect String va fi creat la fiecare iterație. Dacă numărul de bucle este semnificativ, poate cauza multă pierdere de memorie și va crește și timpul de performanță. Un alt caz ar fi instanțierea unui obiect String, cea mai bună practică Java este de a evita utilizarea constructorilor pentru instanțiere, iar instanțierea ar trebui să fie făcută direct. Este mult mai rapid în comparație cu utilizarea unui constructor.

Manipularea elementelor nule necesită o muncă suplimentară, iar de aceea, dacă o metodă returnează o colecție care nu are nicio valoare, cea mai bună practică Java este să returnăm o colecție goală în loc de elementele Null. Aceasta omite elementul null, salvând eforturile necesare pentru testarea elementelor nule.

Împărțirea programului reprezintă o altă bună practică Java. Blocurile de cod trebuie scrise în așa fel încât să înțelegem ușor ce face fiecare fără a trebui să știm cum face. Bucățile trebuie să fie cât mai independente unele de altele, iar interacțiunea dintre ele trebuie să fie se efectueze în moduri cât se poate de clare și simple. Părțile logice trebuie să fie independente, funcțiunea fiecăreia trebuie să fie extrem de clară și cât se poate de limpede enunțată. De exemplu, o funcție care caută ceva într-un șir nu trebuie să facă și ordonarea șirului. Trebuie concepute două funcții independente una de cealaltă, una de căutare, iar alta de sortare. În Java fiecare clasă trebuie implementată într-un modul separat, cu toate metodele ei. Acest lucru va face ca pe viitor să știm unde să le căutăm și apoi o să le putem reutiliza cu ușurință.

Java este un limbaj de programare excelent și pentru a putea fi printe dezvoltatorii de top, aceste bune practice de codare Java vor juca un rol semnificativ în îmbunătățirea abilităților noastre ca programatori. Astfel, vom putea scrie un cod performant, lizibil și bine compus, acestea fiind atributele unui dezvoltator de top.