Java Homework Summary

Mentiuni: anumite anexe utilizate in rezolvarea cerintelor sunt atasate in platforma de laborator aferenta.

Homework.Java-1.1. Pentru diagrama UML din Figure 1. 1 Classes extended, utilizați într-un exemplu toate clasele. Pentru clasa C, utilizați metodele ei, ale clasei B, ale clasei A și metoda readX() din clasa A.

Homework.Java-1.2. Creați o clasă Car care are metodele: start(), accelerate(), stop(), getSpeed(), setSpeed(), startEngine(), stopEngine(). Care dintre ele sunt private/public/protected ? Care sunt atributele clasei ? Extindeți clasa Car la clasa MyCuteCar și adăugați-i acesteia o metodă startRocketEngine(). MyCuteCar are și un motor de rachetă, utilizat împreună cu cel normal!

Investigați utilizarea metodelor din Car in clasa MyCuteCar. Investigați ce metode din clasa Car și MyCuteCar pot fi accesibile in main().

Homework.Java-1.3. Suprascrieți toString() dintr-o clasă astfel încât să returneze toate atributele clasei, într-un format custom.

Homework.Java-1.4. Implementați/Extindeți interfața Animal din Exemplul 1.7 într-o altă interfață, o clasă abstractă respectiv o clasă. Apelați toate metodele.

Homework.Java-1.5. Investigați implementarea diagramei UML din Figure 1. 2 Wheel implements Mechanism. Instanțiați clasa Wheel și apelați metoda goTo().

Homework.Java-1.6. Proiectați diagramele UML si scrieți cod/eventual metode stub pentru Clasele Persoană, Adresă, Profesor, Student, Curs și următoarele metode: addStudentToCourse,

removeStudentFromCourse, addTeacherToCourse, RemoveTeacherFromCourse,

changeTeacherAddress, ChangeStudentAddress, createNewCourse from list of students and teacher.

Homework.Java-1.7. Proiectați diagramele UML si scrieți cod/eventual metode stub pentru Clasele Persoană, Companie, Angajat, Adresă.

Homework.Java-1.8. Implementați diagramă UML Figure 1. 5 Clasa Robot. Folosiți metode stub. Instanțiali clasa Robot și investigați apelarea tuturor metodele.

Homework.Java-1.9. **(Optional)** Scrieți o interfață Shape cu o metodă getArea(). Implementați interfața în clasa Cerc. Creați câteva instanțe ale clasei Cerc și adăugați-le într-o Listă/Vector/etc. Parcurgeți Lista/Vector/etc și calculați aria Cerc-urilor stocate. (Optional) Lucrați in echipa. Folosiți git.

Homework.Java-1.10. **(Optional)** Scrieți codul aferent următoarelor clase și generați o diagramă UML corespunzătoare. Creați o interfață Shape (atribut culoare și metodă getPerimetru). Extindeți interfața la clasa abstractă Shape2D (metodă abstractă getAria) respectiv Shape3D (metodă abstractă getVolum). Implementați Shape în clasa segmentDeDreapta (un segment de dreaptă din geometrie). Extindeți Shape2D și Shape3D la pătrat, cerc respectiv sferă și cub. Efectuați modificările necesare elementelor claselor pentru a afișa:

1. Lungimea segmentului de dreaptă
2. Perimetrul și aria pătratului, cercului, sferei și cubului
3. Volumul sferei și cubului.

Homework.Java-1.11. **(Optional)** Puneți codul de la Exemplul 1.11 într-o metodă care are ca parametru int x și returnează 10/x. Metoda trebuie sa returneze rezultatul operatiei si sa anunte situatia unei impartiti la 0. Apelați metoda. Metoda tratează excepția în corpul ei sau o aruncă, la declarare. Observați diferența de abordare. Care este mai flexibilă ?

Homework.Java-1.12. Citiți un număr x dintr-un fișier și calculați valoarea 2 * x. Pentru orice eroare posibilă, x are valoarea implicită 1. Afișati valoarea calculată. Organizați codul folosind excepții, metode, clase.

Homework.Java-1.13. Modificați Exemplul 1.14 astfel încât utilizatorul să repete citirea unei valori până aceasta este număr întreg.

Homework.Java-1.14. **(Optional)** Scrieți o aplicație care generează numere random intregi, notate cu x, și adaugă 100/x ca și elemente ale unui tablou. Returnați acest tablou. Organizați codul. Tratați cât mai multe dintre excepțiile ArithmeticException, ArrayIndexOutOfBoundsException, Exception.

Homework.Java-1.15. Rezolvați ecuația ax+b=0 în toate cazurile folosind clase, null, Exceptii, Excepții Custom.

Homework.Java-1.16. **(Optional)** Rezolvați ecuația ax2+bx+c=0 în toate cazurile folosind clase, null, Exceptii, Excepții Custom.

Homework.Java-1.17. Creați o Listă de numere întregi si ordonați-o folosind metode și/sau clase predefinite sau un API și afișati lista.

Homework.Java-1.18. Ordonați o list/vector/etc de Student (atributele nume și notă) după atributul nota.

Homework.Java-1.19. **(Optional)** Creați o clasă care gestionează o listă de clase Student, folosind ArrayList și Hashmap. Scrieți o metodă care are ca parametru o listă de studenți si returnează Student cu vârsta cea mai mare. Lucrați in echipa. Folosiți git.

Homework.Java-1.20. Calculați suma a două numere citite de la tastatură. Folosiți Excepții și NULL.

Homework.Java-1.21. **(Optional)** Scrieți o aplicație care citește trei numere a, b, c de la tastatura si rezolva ecuatia ax2 + bx + c = 0. Lucrați in echipa. Folosiți git.

Homework.Java-1.22. Considerați un fișier care poate conține numere întregi, float sau șiruri oarecare. Scrieți o clasă care calculează suma tuturor numerelor întregi din fișier. Tratați excepțiile.

Homework.Java-1.23. **(Opțional)** Proiectați o aplicație care încarcă pagina "https://www.ucv.ro " și extrage primul titlu din Știri.

Homework.Java-1.24. **(Optional)** Transferați o listă de Studenți între două aplicații Java folosind mecanismul de serializare.

Homework.Java-1.25. **(Optional)** Scrieți o clasă A1 și A2 cu aceleași atribute dar cu 1-2 metode diferite. Convertiți o instanță a1 a lui A1 în format JSON. Importați aceste date într-o instanță a clasei A2.

Homework.Java-1.26. **(Optional)** Transferați o listă de Studenți între două aplicații Java folosind JSON - uri salvate în fișiere. Lucrați in echipa. Folosit git.

Homework.Java-1.27. **(Optional)** Creați un proiect Maven. Adăugați GSON la Maven. Rescrieți Exemplul 1.27 folosind Maven pentru importul librăriei GSON.

Homework.Java-1.28. **(Optional)** Folositi un API public la alegere pentru a prelua date, a le prelucra si a le stoca in format JSON pe disc. Lucrati in echipa. Folositi git.

Homework.Java-1.29. **(Optional)** Generați 3 de numere aleatoare și scrieți fiecare număr în fișiere diferite, folosind Thread-uri. Cum pot gestiona un număr oarecare de numere aleatoare ?

Homework.Java-1.30. **(Optional)** Scrieți un serviciu (micro-serviciu) definit ca o aplicație web, SPRING based, care primește un JSON de forma {a :1, b :2} și returnează un JSON care conține suma a+b. Folosiți metodele GET și POST. Testați aplicația din browser sau dintr-un tool de genul POSTMAN. Investigați utilizarea GET și POST în cele două tool-uri.

Homework.Java-1.31. **(Optional)** Pentru aplicația web de la Homework.Java-1.30, scrieți cod Java care apelează serviciul.

Homework.Java-1.32. **(Optional)** Scrieți un serviciu (micro-serviciu) definit ca o aplicație web, SPRING based, care rezolvă ecuația ax+b=0 în toate cazurile folosind clase, null, Exceptii, Excepții Custom.