1. Conventii de nume

-Numele variabilelor nu respecta conventia Java (ex: valoare\_imprumut -> valoareImprumut, tip\_cont -> tipCont)

2. Conventii de nume

-Numele clasei este mult prea generic, pot exista mai multe tipuri de cont, asadar o voi redenumi

3. Conventii de nume

-Numele pachetului nu este sugestiv, asadar trebuie redenumit in ro.ase.csie.cts.tema.(clase,enumerari,exceptii etc.) pt a fi mai ordonat

4. Lizibilitatea codului si accesul mai usor la metode si functii

-Eliminare valori hardcodate

5. Formatare – Asigurarea unui cod formatat, organizat

-Codul nu este formatat, in unele locuri exista spatii, in altele nu, etc.

6. Folosirea enumerarilor in loc de constante int

-Variabilia tip\_cont nu ar trebui sa fie de tip int, deoarece s-ar potrivi mai bine pe un enum, cu valori predefinite;

7. Folosirea enumerarilor in loc de constante int

-STANDARD, BUGET, PREMIUM, SUPER\_PREMIUM trebuie mutate intr-un enum pentru ca se refera la aceeasi entitate si codul va fi mult mai ordonat

8.Regula de logica a codului

-Istoricul imprumuturilor nu se salveaza nicaieri

9.Formatare – Codul trebuie sa aiba in vedere o ordine a ceea ce se implementeaza

-Construtorul trebuie realizat dupa definirea variabilelor

10.Do one thing

-Metoda getRata() are ca scop returnarea unei valori, nu si afisarea ei, deci trebuie eliminat printul

11. Comentarii – Explain yourself in code

-Comentariile sunt irelevante pentru cod; nu se folosesc comentarii pentru a justifica actiunile

12.Conventii de nume

-Metoda getRataLunara() nu calculeaza ceea ce numele sugereaza

13. Exceptii – Utilizarea exceptiilor verificate pentru conditii

-Metoda setValoare ar trebui sa aiba o exceptie custom si un mesaj, pentru a sti motivul erorii

14. Folosirea lui override toString

-Metoda to\_string nu suprascrie metoda corecta, si anume toString din Object

15. Regula ce tine de inconsistenta lingvistica

Trebuie stabilit de la inceput ce limba se foloseste pentru scrierea codului; unele variabile sau metode sunt scrise in lb engleza, restul codului fiind precdominant in romana

16.Concatenarea unui numar mare de stringuri (ex: return "Loan: "+this.valoare\_imprumut+"; rate: "+this.rata+"; days active:"+zileActiv+"; Type: "+tip\_cont+";";) este recomandat sa se faca cu StringBuffer

17. Minimizarea metodelor nefolositoare

-Metoda print() nu realizeaza nimic, este inutila

18.Programare generala

-Variabila vb nu este folosita nicaieri si i se atribuie o valoare hardcodata

19. Conventii de nume

-Numele metodei calculeaza nu sugereaza ceea ce face ea de fapt; acesta trebuie schimbat

20. Minimizarea accesului la clase si membrii

-Metoda calculeaza() este independenta fata de clasa si nu are sens in interiorul clasei, din cauza acelui static

21.Conventii de nume

-Valorile hardcodate trebuie puse in constante cu nume sugestiv

22. Conventii de nume

-Variabila temp nu este folosit, iar numele nu este sugestiv

23. Programare generala – minimizarea scopului variabilelor

-Variabila cont nu este necesara, deoarece se poate accesa elementul din lista direct

23. Conventii de nume

-Se face confuzie intre Cont si Account; Account nu exista, clasa mea fiind denumita Cont

24. Comentarii

-Comentariul: "// 1.25% broker's fee" nu are legatura cu codul

24. Programare generala – minimizarea scopului variabilei locale

-Valoarea 0.125 ar trebui pusa intr-o variabila constanta

25.D.R.Y.

-Eliminarea valorii 365, ea fiind declarata mai sus in variabila temp

26.Comentarii

-Comentariul "// dobanda-principala" este un comentariu inutil

27. Functiile - Fac un singur lucru

-Functia calculeaza() are prea multe responsabilitati, asadar voi face alte functii de care sa usurez intelegerea codului

-Voi reface calculul functiei cu ajutorul unei formule gasite pe acest site: <https://www.investopedia.com/terms/e/effectiveinterest.asp>

28. Functiile - Fac un singur lucru

-Functia calculeaza() trebuie sa efectueze operatia numai pentru contul curent, nu pentru o lista de conturi

29. Minimizarea accesului la clase si membrii

Variabilele trebuie declarate private, pentru a se respecta incapsularea

30. Regula ce tine de logica

-Constructorul nu are ca parametru membrul clasei zileActiv

31. Reguli fundamentale – Duplicarea codului

-Metoda setValoare() este duplicata in constructor

32. Reguli fundamentale

-Inconsistenta in denumiri: constructorul trebuie sa respecte o regula de organizare a denumirilor de la bun inceput (membrii trebuie accesati ori cu this ori sa se stabileasca conventii de nume in constructor)

33. Conditie ce tine de logica

-Conditia pentru imprumut este gresita. Imprumutul este mai mare decat 0