Laborator 2

1. **Ce e Graal? Ce e Poliglotul?**

   - Polyglot = folosim mai multe limbaje in acelasi proiect
   - Truffle = interpretator de cod = framework
   - Graal = masina virtuala -> ne ajuta pentru portabilitate de pe o platforma pe alta

2. **Care sunt limbajele suportate de Graal?**

   -Python
   -Java Script
   -R
   -C/C++
   -Kotlin
   -Scala
   -Ruby
   -Node
   -OpenJDK
   -JS
   -Oracle
   -Standalone

3. **Ce ar fi o functie lambda?**

-

-Lambda este o functie fara nume (anonima).
- Curry este o functie care proceseaza cate un argument (e important sa precizam recursivitatea)

4.a Ce este functia Curry?

-functie care are argumente multiple

4. **Care e diferenta intre tipurile statice si dinamice de date?**

   -cele statice au nevoie mai intai de o declaratie si isi vor pastra tipul pe parcursul programului pe cand cele dinamice isi pot schimba tipul pe parcursul programului

5. **Ce limbaje suporta tipurile statice?**

-C/C++

-Java

6. Ce limbaje au cele dinamice?

-Java Script
-Python
-Ruby

7. Ce este AST?

-transforma instructiunile indiferent de limbajul in care sunt scrise intr-un arbore de sintaxa abstract si este interpretat de catre Truffle si Graal

8. Diferenta intre modelul Charch si modelul Turing?

-in modelul Charch toate problemele sunt rezolvate printr-o serie de functii care se evalueaza un ape alta la nesfarsit
-modelul Turing totul e bazat pe un state machine si codul e rulat secvential

9. Care sunt problemele masinii Java?

-masina e veche,lenta si am avea problem cu interpretorul masinii virtuale
-foloseste mai multe resurse de partea de masina virtuala iar pe partea de containere resurse diminuate

10. Ce este limbajul gazda si limbajul tinta la Polyglot?

limbajul gada=limbajul pt care initiem contextul

limbajul tinta=limbajul apelat de limbajul gazda

11. Care ar fi comanda GU ca sa vede care sunt limbajele pe care o suporta versiune de Graal pe care o avem instalata?

Gu available

12. Ce inseamna functie de tip curry?

Functiile cu argumente multiple sunt premise prin intoarcerea functiilor ca rezultat

Evaluam partial fiecare functie

13. Ce inseamna evaluare partial?

reprezinta un algoritm care are ca scop optimizarea procesului de compilare

14. Ce este evaluatorul partial?

Inlocuieste o valoare cu o valoare particulare la un moment dat

15. Ce inseamna constant folding?

Expresia cu operanzi constant pot fi evaluate in timpul compilarii si va avea ca rezultat cresterea performantei de executie si reducerea dimensiunii codului de aplicatie

16. Ce se intampla cand avem o implementare cross compiler? Cum se va comporta programul daca avem o masina tinta si masina gazda?

Va fi afectata dpdv a modului in care se face calculul pe una si se executa pe cealalta.

17. La ce se refera dumping-ul?

Furnizeaza si suporta generarea de informatii detaliate despre anumite structuri . El functioneaza eronat

-se vor ingheta informatii privind executia in anumite subzone ale memorie care vor contine date sau functii care au avut disfunctionalitati la executie si deasemenea se vor mai stoca starile registrilor procesorului

18.Teoria s-m-n a lui Kleene

-teorema spune că pentru un limbaj de programare dat și numere întregi pozitive *m* și *n*, există un anume algoritm care acceptă ca intrare cod sursa a unui program cu *m* + *n* variabile libere, impreuna

cu *m* valori. Acest algoritm generează cod sursă care substituie efectiv valorile pentru prima *m* variabile libere, lăsând restul variabilelor libere.

19.Deoptimizarea

Truffel permite acestei operatii care reface codul masina optimizat in forma sa originala de tip ADT pt a putea avea un feedback continu necesar profilului aplicatiei

Laborator 3

1. **Ce este descompunerea functionala dpdv al programarii orientate pe obiect?**
   1.Identifica lista de forme in baza de date
   2.deschide lista de forme din baza de date
   3.ordoneaza aceasta lista conform unui set de reguli
   4.afiseaza formele pe monitor(identifica tipul formei, obtine locatia acesteia,apeleaza functia care poate sa afiseze aceasta forma furnizandu-I locatia formei)

2.Care sunt principalele probleme ale specificatiilor primate de la client si care ar fi principalele motive ale aparitiei acestora?

Specificatiile sunt:

-incomplete

-de multe ori gresite partial sau total

-inselatoare(din cauza diferentelor de limbaj si reprezentare a unor concept dintre proiectant si beneficiar)

-niciodata nu surprind toate aspectele implicatiilor

Motivele modifcarilor de specificatii:

-beneficiarul observa noi posibilitati de imbunatatire a softului in urma discutiilor cu dezvoltatorul

-dezvoltatorul devine mai familiar cu domeniul problemei specific utilizatorului

-mediul pt care este dezvoltata aplicatia se schimba

3.Ce efecte au coeziunea scazuta si cuplarea stransa?

Coeziune: cat de apropiate sunt operatiile dintr-o metoda

Cuplarea: cat de stransa este legatura dintre 2 metode

Telul ar fi:

Sa creeze functii:

-cu integralitate interna(coeziune stransa)

-care au relatii directe,vizibile,flexibile si directe(nu prin intermediare/cuplare stransa)

4. Care sunt pasii din proiectarea unei aplicatii oop?

-izolareza obiectele din lumea reala care interactioneaza in cadrul problemei

-abstactizeaza -obiectele care au proprietati si comportamente similare sunt grupate in multimi/grupuri numite clase

-determina-reponsabilitatile grupului dpdv al interactiunii cu alte grupuri

5.Explicati procesul de Brainstorming?

Etepele: stabilim o problema,avem un grup care este condus de un moderator, din membrii grupului cativa sunt cu experienta iar altii familiarizati cu domeniul problemei, se pune accentul pe generarea de idei cu focalizarea pe cantitatea de idei generate, acesta conducand la niste idei de calitate, se vor elimina criticile a.i. sa nu influenteze descoperirea de idei noi,fiecare idee e bine venita, se vor analiza toate aceste idei generate, se vor combina si se vor imbunatati ideile si se poate repeat iterative pana cand se ajunge la o forma decenta pt construirea specificatiilor problemei

6.Ce pasi implica Metodologia oop?

Brainstorming-pt a localiza clasele posibile

Filtrarea-claselor pt a gasi duplicatele si a elimina pe cele in plus

Scenarii- sunt necesare pt a fi siguri ca s-a inteles colaborarea intre obiecte

Algoritmi-sunt proiectati pt a define toate actiunile pe care clasele trebuie sa le poata efectua

7.La ce se folosesc fisierele crc?

-sunt utilizate pt a verifica diverse erori sau integritatea unor fisiere

8.Cum creaza Kotlin obiectele asociate unei clase? Ce insemna instant, ce insemna clasa? Pe unde sunt stocate in memorie aceste informatii?

Instant=un obiect variabil

Clasa=un tip de referinta

In instanta nu este stocat obiectul in sine , vom avea o referinta spre o locatie de memorie care va stoca informatiile despre instanta, locatia va fi pe heap iar partea de referinta pe stiva

9.Care este rolul lui this si care este rolul lui scope(kotlin)?

-this (obiect current)-acceseaza informatiile dintr-un scope prin intermediul operatorului @ : this@scopeaferent

-scope o sa reprezinte o regine de cod care este delimitate sintactic si in care vom avea un anumit context pt anumite unitati sau nume

10.Care este fow-ul general pt rezolvarea si implementarea unei problem in paradigm orientata obiect?

Mai intai se rezolva problema iar apoi se implementeaza.

Initial trebuie sa obtinem niste obiecte,sa le creem reprezentari abstacte sis a obtinem clase. Apoi sa trecem la analiza sau sa recreem aplicatia pt cazuri concrete.

11.Proiectarea orientate obiect

O metodologie orientate pe rezolvarea problemei care produce o solutie a problemei in termini de entitati incapsulate numite obiecte

12.Un obiect=o entitate sau un lucru care are sens in contextual problemei

13. Problemele sunt rezolvate prin:

-izolarea obiectelor implicate

-determinarea proprietatilor si actiunilor

-descrierea colaborarii intre obiecte cu scopul rezolvarii problemei

14**. O interfață** =este o listă de metode care trebuie definite de către orice clasă care implementează interfața respectivă. O interfață poate defini şi constante (public static final).

Laboratorul 4

1.Cum definim proiectarea aplicatiilor software?

Proiectarea,in contexul aplicatiilor software reprezinta un process de rezolvare a unor problem,obiectivul fiind sa se gaseasca si sa se descrie o cale de a implementa necesitatile functionale ale sistemului tinand cont si de constrangerile impuse de:

-nivelul de calitate asteptat

-specificatiile platformei tinta

-de necesitatile specific ale procesorului care trebuie modelat

-aspectele de business cum ar fi: termenul de realizare si fondurile dispuse pt respectivul proiect

2.Care este diferenta intre must have si nice to have?

must have=criticalitatea cerintelor pt functionarea aplicatiilor

Nice to have=ce isi doreste utilizatorul dar nu ii solutioneaza nicio problema critica ca si functionalitate in cadrul aplicatiei

3.Definiti componentele

Este un modul sau o clasa care are anumite proprietati care o fac reutilizabila intr- arhitectura software

4.Din ce este alcatuit un sistem dpdv al paradigmei modulare?

Un sistem este alcatuit din:

-subsisteme

-componente

-module

5.Ce este UML-ul?

Unified Modeling Language reprezinta un limbaj grafic pentru vizualizarea,specificarea,dezvoltarea si documentarea componentelor unui system software de dimensiuni medii sau mari

Sau

UML ofera o maniera standard pt a crea schema unui system pornind de la aspect concrete cum ar fi blocuri de cod,scheme pentru bazele de date,componente reutilizabile si ajungand la aspect abstracte pecum capturarea fluxului de desfasurare a unei afaceri sau functii ale sistemului

6.Ce este modelul domeniului?

Modelul de domeniu este o schiță a entităților elementare ale sistemului și a relațiilor dintre ele. Este independent de platformă (nu este destinat unui limbaj de programare specific) și atributele nu au tipuri de date.

clasele modelului de domeniu sunt foarte simplificate. Au doar atribute importante și nu au metode.

În modelul de domeniu, vom folosi doar o notație de clasă simplificată cu numele clasei și atributele acesteia.

(NU stiu daca e ok)

7.Ce ne ofera UML?

UML ofera semantici si notatii pentru:

-user interaction/use case model

-interaction/communication model

-state/dynamic model

-logical/class model

-physical component model

-pysical deployment model

-mai defineste si solutii pentru extinderea sau dezvoltarea dupa nevoi a unor mecanisme existente

8.Ce este modelul de proiectare,implementare in cascada(modelul Waterford?

-este un model in care intoarcerea se face in functie de testarea pe care o avem la fiecare etapa

-are pasi inapoi interactivi doar intre 2 etape consecutive

9.Enumerati modelele sistem

-modelul obiect

-modelul functional

-modelul dynamic

10.Enumerati modelele task-urilor

-harta pert: care sunt legaturile dintre taskuri?

-planificarea: cum poate fi aceasta realizata in durata de timp rezervata?

-harta organizational: care sunt rolurile in proiect sau organizatie?

11.Cum se defineste un model dpdv UML?

Un model este o descriere complete a unui system dintr-o perspectiva particulara.

Relatiile modelului:

-modelul sistemului: Structura,Functionalitate,comportament dinamic

-problemele modelului: propuneri,argumentare,rezolvare

-modelul taskurilor: organizare,activitati,planificare

12.La ce se refera principiul de baza in gestiunea abstractiilor?

principiul abstractizării (sau principiul abstractizării) este un dictum de bază care are ca scop reducerea duplicării informațiilor într-un program (de obicei, cu accent pe

duplicarea codurilor) ori de câte ori este practic prin utilizarea abstractizărilor furnizate de limbaj de programare sau biblioteci de software [este necesară citarea].

recomandă evitarea duplicării informațiilor în general și, de asemenea, evitarea duplicării efortului uman implicat în procesul de dezvoltare software.

(nu stiu sigur cum vine)

13.Care sunt principiile elementare reutilizarii?La ce se refera proiectarea pentru flexibilitate?

-anticiparea activa a schimbarilor pe care un proiect le poate suporta in viitor si planificarea adecvata a arhitecturii sistemului in functie de acestea

14.Cum gestionati problemele de inlocuire sau problema disparitiei unor componente care ne sunt puse la dispozitie in momentul in care dezvoltam aplicatia?

15.Ce inseamna TDD?

Test Driven Development (TDD) este o abordare de dezvoltare software în care sunt dezvoltate cazuri de testare pentru a specifica și a valida ce va face codul.

16.Care sunt principiile SOLID?

Principiul Responsabilității unice (Single responsibility principle)

O clasă trebuie să aibă o singură rațiune pentru a se schimba. Cu alte cuvinte, clasele complicate trebuie divizate în clase mai mici care au responsabilități explicite.

Principiul Închis/Deschis (Open/closed principle)

Entitățile software trebuie să fie deschise pentru extindere, dar închise pentru modificare. Acest lucru înseamnă că fiecare entitate software trebuie scrisă astfel încât să poată fi extinsă fără a necesita modificări explicite în interiorul lor.

Principiul substituției Liskov (Liskov substitution principle)

Clasele derivate trebuie să fie substituibile pentru clasele lor de bază. Adică funcțiile care folosesc referințe la clase de bază, trebuie să poată manipula într-un mod transparent instanțele claselor derivate din acestea. Așadar clasele derivate trebuie doar să adauge funcționalități noi la clasele de bază, nu să le înlocuiască pe cele existente.

Principiul separării Interfețelor (Interface segregation principle)

Mai multe interfețe specifice sunt mai bune decât o interfață generală. Nici un client nu trebuie forțat să depindă de metode pe care nu le folosește. Numărul de membri din interfață care este vizibil pentru

clasele dependente trebuie minimizat. Clasele mari vor implementa mai multe interfețe mai mici care grupează funcțiile după maniera lor de utilizare.

Principiul Dependenței inverse (Dependency inversion principle)

Modulele de nivel arhitectural superior nu trebuie să depindă de cele de nivel inferior. Ambele trebuie să depindă de abstracții care, la rândul lor nu trebuie să depindă de detalii (implementări concrete). Practic detaliile depind de abstracții, nu invers. Dacă această dependență nu este vizibilă în faza de proiectare, atunci ea se construiește

17.Descrie principiul separarii interfetelor

Principiul separării Interfețelor (Interface segregation principle)

Mai multe interfețe specifice sunt mai bune decât o interfață generală. Nici un client nu trebuie forțat să depindă de metode pe care nu le folosește. Numărul de membri din interfață care este vizibil pentru clasele dependente trebuie minimizat. Clasele mari vor implementa mai multe interfețe mai mici care grupează funcțiile după maniera lor de utilizare.

18.Procesul dezvoltare a unei aplicatii

-specificarea cererilor

-analiza sistemului

-proiectarea sistemului

-implementarea sistemului

-testarea sistemului

-instalarea sistemului

-mentenanta sistemului

19.Tipuri de proiectare specific:

-proiectarea arhitecturala

-proiectarea claselor

-proiectarea interfetei vizuale

-proiectarea bazelor de date

-proiectarea algoritmilor

-proiectarea protocoalelor

20.Bune practice in proiectare(BPP)-devide and conquer

-este mai usor sa gestionam lucrurile mici decat unul mai mare

20.BPP-increse cohesion where possible

-coeziune: “aplica D&c dar pastreaza cat mai “

-tipuri de coeziune posibile:

*functionala

*la nivel de strat architectural

*la nivel de comunicatii

*secventiala

*procedural

*temporala

21. BPP-reduce coupling where possible

-cuplarea apare atunci cand sunt interdependente intre un model sau altul

Tipuri de cuplare:

-Continut

-comun

-control

-eticheta

-date

-apel de functie

-utilizare de tip

-incluziune/import

-externa

22.BPP-keep the level of abstraction as high as possible

-ascundem dataliile pt a putea pastra dpdv logic urmarirea coerenta a cee ace se intampla in designul de proiectare propus

23.BPP-increase reusability where possible

Exista 2 principii:

-proiectare pt reutilizare- se refera la reutilizarea diverselor componente a sistemelor in cadrul aceluiasi sistem sau in cadrul altui sistem similar

-**Proiecteaza prin reutilizare**

24 BPP-reuse existing designs and code where possible

-clonarea nu art r sa fie vazuta ca o forma eficienta de utilizare deoarece nu prea se inteleg algoritmii de gestiune a coerentei

25.BPP-design for flexibility

-anticiparea activa a schimbarilor pe care un proiect le poate avea in viitor si pregatirea din start a sectiunilor pt ele

26.BPP-anticipate obsolenscence

-priectarea a. i. la aparitia schimbarilor atat compatibilitatea cu sistemul gazda cat si modificarea unor componente prin rescriere vor induce un impact minim

27.BPP-design for portability

-obiectiv principal:supravietuirea pe termen lung a softwareului dezvoltat

28.BPP-design for testability

-trebuie luate masuri pt a usura procesul de testare care poate fi manual sau automat

29.Principii OOP pt proiectarea claselor

-coeziunea claselor

-inchisa vs deschisa-o clasa tr sa fie deschise le extindere dar inchise pt modificare

-raspundere unica-o clasa tr sa se ocupe de o singura problema

-separarea interfetelor-(daca clenti diferinti depend de metode diferite ale aceleasi clase at o modificare intr-o metoda ar putea conduce la recompilarea si reinstalarea aplicatiilor tuturor clientilor afectati)

-crearea unei interfete separate pt fiecare tip de client care sa contina numai setul de metode necesare pt clientul x

-dependenta inversa-

-substitutia liskov

-legea lui demeter-

-reutilizarea abstractiilor

30.Matrici pt masurarea coeziunii claselor:

-clasa are un nr de A attribute

-clasa are un nr de M metode

-fiecare atribut Ai este accesat de R(A) metode

31.Legea lui Demeter

Legea lui Demeter este un caz specific de cuplare liberă

-Fiecare unitate ar trebui să aibă doar cunoștințe limitate despre alte unități: numai unități „strâns” legate de unitatea curentă.

-fiecare unitate ar trebui să vorbească numai cu prietenii săi; nu vorbi cu străini.

- Vorbește doar cu prietenii tăi apropiați

Laborator 5

1.Ce este un eveniment?

Un eveniment poate fi definit ca un tip de semnal catre program.Acesta ii indica programatorului ca ceva s-a intamplat

2.Explicati o maniera de gestionare a unui eveniment(incluzand si sistemul de operare)

- la nivel de sistem de operare se refera la crearea unei cozi in care se vor pune acele evenimente cu scopul de a trata evenimentele in ordinea in care sunt scoase(motivul: daca avem mai multe evenimente de cat putem trata atunci sa nu pierdem niciunul din ele)

3.Cum sunt gestionate dpdv architectural aceste evenimente?

-evenimentul intra intr-o coada,ajunge la un distrubuitor si in functie de biblioteci(de niste reguli descries de programator) ,distribuitorul le organizeaza pe diferite canale de evenimente unde sunt procesate de mai multe module

4.Care este diferenta intre un model sincron si unul asincron?

eveniment sincron=apar inainte de o anumita actiune

eveniment asincron=sunt generate de niste actiuni

5.Maniera de tartare a unui eveniment dpdv a programatorului

Polling-citirea secventei intr- bucla infinita

Interrupt-driven-tratarea intreruperilor

Event-driven-evenimentul este asteptat intr-o bucla infinita

6.Care sunt avantajele procesarii orientate pe eveniment?

-sunt mai portabile

-permit tratarea mai rapida

-se pot folosi in time-slicing

-incurajeaza reutilizarea codului

-se potrivesc cu oop

7.La ce se refera EDP ca acronym si care sunt componentele principale implicate?

EDP= **event-driven programming**

= este o paradigmă de programare în care fluxul programului este determinat de evenimente precum acțiuni ale utilizatorului (clicuri de mouse, apăsări de taste), ieșiri ale senzorilor sau mesaje care trec de la alte programe sau fire.(wiki)

Componente:

-generator de evenimente

-sursa evenimentelor

-bucla de evenimente

-gestionari de evenimente

-event mapper

-inregistarea evenimentelor

8.Diagrama de secventa EDP-explicatii

-in aplicatii se efectueaza niste actiuni asupra unor elemente din interfata cu utilizatorul iar pentru obiectele asociate interfetei se vor generera evenimentele respective ca urmarea a acestor actiuni

9. Ce este Dispatcherul?

-este o entitate care se ocupa de distribuirea evenimentelor catre niste gestionari de evenimente

10.Care este specificarea acronimului WYSIWYG si care este legatura lui cu EDP?

-se refera la fapul ca avea niste instrumente pe care le folosim la crearea interfetei grafice cu utilizatorul si ca aceste instrumente permit sa automatizam anumite taskuri care se impugn in procesul de dezvoltare a aplicatiei

11.Explicati maniera LOOK AND FEEL?

-look se refera la maniera in care este perceputa sau vazuta interfata grafica

-feel este maniera in care interfata raspunde la evenimentele utilizatorului

12.Ce semnificatie are codul asociat pentru look si codul asociat pentru feel?

Codul asociat pentru look se numeste resursa

Codul asociat pentru feel se numeste sursa

13.Care sunt criteriile minimale care trebuie indeplinite de o interfata grafica cu utilizatatorul care este proiectata correct?

O interfata este buna daca are urm caracteristici:

-eleganta

-il ghideaza pe looser

-ofera informatii ajutatoare

-foloseste o ierarhie de interfete

-permite ca utilizatorul sa faca greseli

14.Ce este un widgets?

sunt obiecte din cadrul unui GUI (graphic user interface) orientat obiect

15.Explicatii patternul de model view controller(mvc)

-arhitectura este formata din controller,view si model cu scopul de:

Controlerul-primeste niste intrari de la utilizator si va actiona asupra unor obiecte ale modelului pt a efectua prelucrarile aferente

View-afiseaza modelul intr-o anumita forma care este specific apt utilizator

Model-se refera la date,la starea aplicatiei si logica aplicatiei

16.Ce este SDL si unde se foloseste?

SDL=simple directmedia layer

- era folosit pt jocuri video

- SDL este, de asemenea, adesea folosit pentru porturi ulterioare pe platforme noi cu cod vechi.

17.Care sunt modelele SDL si care sunt echivalentele DIRECTE?

Video=directDraw –ascunde accesul native la ecran

Gestiunea evenimentelor=directinput-ascunde accesul native la evenimente

Joystick=directinput-ascunde accesul native la maneta de joc

Audio=directsound-ascunde accesul native la placa audio

cd-rom-fara echivalent-acceseaza direct CD audio

fire de executie=fara echivalent-functii helper peste cele native de gestiune a firelor de executie

timere=fara echivalent-functii helper peste cele native de gestiune a timerelor

18.Ce inseamna list comprehation?

-se refera la definirea si initializarea simplificata a unei liste

19.Ce inseamna eveniment virtual si cum este tratat el in tkinter?

Puteți crea propriile dvs. noi tipuri de evenimente numite evenimente virtuale. Puteți să le dați orice nume doriți, atâta timp cât este inclus în perechi duble de << ... >>.

20.Care este maniera aborescienta de creare a meniurilor in tkinter?

-se face un meniu ierarhizat si prima data creem obiectele care pot fi legate la un process

21. Ce este paradigm orientate evenimet?

Un program bazat pe evenimente este unul care răspunde în mare măsură la evenimentele utilizatorilor sau la alte informații similare.

22. Pick Correlation

-procesul de selectie a unei ferestre sau aplucatii care trebuie sa trateze un eveniment oarecare se numeste corelatoe de selectie(pick correlation)

Laborator 6

1.Care sunt principalele domenii in care s-ar putea utiliza python?

-aplicatii WEB

-fronted

-backend

-hardward low level(support native pe unele microcontrolere)

-arhitecturi complexe bazate pe microservicii

-inteligenta artificiala

-securitate(pentest(mai mult automatizare),criptografie dar nu treburi seriaoase)

2.Care sunt paradigmele de programare suportate de Python?

-functionala

-imperativa

-orientata obiect

-procedurala

3.Ce tipuri de conversii explicite ale tipurilor de date sunt suportate in Python?

Integer,floating point,long si complex

4.Explicati instructiunea try except?

este o intructiune de tipul incerca si daca nu reuseste sa execute intructiunea va emite un cod de eroare

tipuri de erori: intrare,iesire,import,index,cheie,nume,tip

5.Ce este o exceptie?

Exceptiile reprezinta situatii care apar in timpul rularii unui program si care determina oprirea acestuia (de pe net)

6.La ce este utilizata intructiunea pass?

Utilizata cand tine locul unei instructiuni vide

7.Ce diferente sunt fata de C in cazul operatorilor simplii?

Not,and,or,in,not in,is,not is(sa mai cauti raspunsul la intrebarea asta)

Repetitia de siruri,concatenarile,asignarile simultane

8.Ce operatori de atribuire sunt suportati in Python?

=

+=

-=

*=

/=

%=

**=

//=

9.Care este diferenta intre operatorii de apartenenta si operatorii de identitate?

Operatori de apartenenta:

In-verifica daca un obiect este intr-o secventa

Not in-verifica daca un obiect nu este in secventa

Operatori de identitate:

Is-verifica daca 2 variabile au aceeasi referinta(adresa) de memorie

Is not-verifica daca 2 variabile nu au aceeasi referinta(adresa) de memorie

10. Ce standar de caractere este implicit pt Python?

Strandard Unicode

11.De ce se poate executa un cod in interiorul unui sir?

-permite evaluarea unei expresii,primeste ca intreare un sir de caractere sau poate sa primeasca o secventa de cod complilata

Etapele parcurse de eval:

Analizați expresia

Compilați expresia în bytecode

Evaluează bytecode-ul

Returnează rezultatul

12.Care este diferenta intre o tupla si un dictionar?

La o tuple nu se pot modifica valorile la dictionar se pot modifica datele

La dictionar retinem perechi de date care au relatie intre ele

13.Care sunt atributele unui obiect de tip fisier?

Dupa ce s-a deschis un fisier se obtine o referina care are mai multe attribute.

-file.closed-returneaza true daca fisierul este inchis,false altfel

-file.mode-returneaza modul de acces in care a fost deschis fisierul

-file.name-returneaza numele fisierului

-file.softspace-returneaza false daca este necesar spatiu explicit pentru afisare(print),true altfel

14.Care este metoda corecta de tartare a operatiilor cu fisiere?

-facem un try ,o deschidere,o procesare,o inchidere si o exceptie in cazul in care s-au produs erori la executie

15.Cum se trateaza apelul unei functii in Python?

-parametrii formali se inlocuiesc cu parametrii actuali

-se respecta nr de argumente,ordinea de argumente ,daca exista parametrii default

16.De ce se pot intoarce valori multiple in Python?

-pt ca avem tuple si dictionary,avem unpacked iterabil si putem returna tuple

17.Cum putem simula transferal prin referinta la iesirea dintr-o functii in python?

Când treceți argumentele funcționale prin referință, aceste argumente sunt doar referințe la valorile existente. În schimb, atunci când treceți argumente după valoare, aceste argumente devin copii independente ale valorilor originale**.**

18.La ce ne folosesc cuvintele cheie atunci cand le utilizam ca argumente?

Când apelați funcții în Python, va trebui adesea să alegeți între utilizarea argumentelor cuvintelor cheie sau a argumentelor poziționale. Argumentele cuvintelor cheie pot fi adesea folosite pentru a face apelurile funcționale mai explicite.

Cand folosim cuvintele cheie:

De multe ori putem omite argumentele care au valori implicite

Putem rearanja argumentele într-un mod care le face cele mai lizibile

Numim argumente după numele lor pentru a face mai clar ce reprezintă

19.Ce sunt tuplele?

Similiare cu listele, sunt colectii indexate de obiecte

20.Ce este un dictionar?

Asocierea unui grup de valori si a cheilor valorilor la o variabila

Laborator 7

1.Ce sunt functiile parametrizate?

sunt functii la care nu conteaza ce tip de date vor fi transmise deoarece sunt convertite la ele iar la prelucrarea rezultatului el poate fi particularizat cu un caz explicit

scop: permite selectia unei game extinse de tipuri pt functia respective

2.Ce sunt tipurile parametrizate?

sunt tipuri container datorita asocierii cu colectiile si deoarece contin 1 sau mai multi parametrii

-pt declarare se folosesc <>

-cele mai cunoscute tipuri parametrizate sunt colectiile

3.La ce se refera polimorfismul limitat superior? Ce este polimorfismul limitat?

-polimorfismul limitat este abilitatea unei functii de a lucre pe mai multe tipuri

-polimorfismul limitat superior reprezinta restrictionarea tipurilor la cele existente in subclasele corespunzatoare limitarii

4.Ce inseamna limitare superioara multipla?

-marginirile superioare se declara ca clause superioare where

Utila/in ce sens dorim sa extindem o functie ca sa poata sa lucreze prin ce fel de valori: ca sa poate lucre cu valori serializabile

5.Ce inseamna declaration site variance?

Varianta la momentul declararii-se foloseste un modificator de tip …. ca sa

este varianța care este specificată în punctul în care genericul este definit - de exemplu, în definiția clasei, interfeței, funcției sau proprietății de extensie.

(de pe net)

6.Ce esre variance annotation?

O adnotare a variației este un modificator aplicat unui parametru de tip sau argument de tip al unui generic, pentru a declara varianța acestuia.

(de pe net)

7.Ce inseamna type projection?

O proiecție de tip este un tip care a fost limitat în anumite moduri pentru a obține caracteristici de variață.

(de pe net)

8.Care sunt regulile pt type projection?

Declarație-site variață - puteți specifica varianța în definiția clasei sau a interfeței.

Utilizați varianța site-ului - puteți specifica varianța în locurile din codul dvs. în care utilizați o instață a unui generic.

9.Ce sunt functiile generice?

Functii care folosesc variabile generice

-valoarea lor de return este una generic(nespecificata)

(incomplete raspuns)

10.Ce este o constrangere generic?

Sunt folosite pentru a stabili in mod clar toate tipurile care pot fi substituite unui tip parametrizat de date

Vom avea doar un tip de date intre paranteze

Daca sunt mai multe tipuri folosim clauza where

11.Ce inseamna type erasure?

Mecanismul de verificare a sigurantei pt un tip de date,verifica declaratiile generice numai in timpul complilarii,la runtime instantele tipurilor generice nu retin nicio informative cu privire la tipurile lor

12.Ce sunt tipurile de date algebrice?

-se refera la un set inchis de tipuri composite impreuna cu functiile necesare procesarii lor

13.Ce sunt listele in Kotlin?

-O colectie ordonata de elemente

14.Ce sunt colectiile?

O colectie este o entotate care grupeaza mai multe elemente si permite tratarea lor unitara(echivalent mathematic: multimea)

15.Ce sunt clasele sigilate?

-restrictia este la nivel de ierarhie ,in sensul ca toate subclasele sunt cunoscute la momentul complilarii iar dup ace este complilat modulul nu mai pot fi adaugate alte subclase

16.Ce operatii ne ofera colectiile de tip set?

-operatii inspectie date

```
public interface Set<out E> : Collection<E>  //ex2
{ //Operatii inspecție date
override val size: Int
override fun isEmpty(): Boolean
override fun contains(element: @UnsafeVariance E): Boolean override fun iterator(): Iterator<E>
  //Operatii de masă
override fun containsAll(elements: Collection<@UnsafeVariance E>): Boolean }
public interface MutableCollection<E> : Collection<E>, MutableIterable<E> //ex2
{  //Operatii inspecție date
override fun iterator(): MutableIterator<E>
   //Operații de modificare
public fun add(element: E): Boolean
public fun remove(element: E): Boolean
   //Operatii de masă
public fun addAll(elements: Collection<E>): Boolean
public fun removeAll(elements: Collection<E>): Boolean
public fun retainAll(elements: Collection<E>): Boolean
public fun clear(): Unit }
```

17.Ce inseamna HashSet?

-este o varientate de set

-creaza o colectie care foloseste o tabela de dispersie sa memorize variabile sau date

18.Ce inseamna LinkHashSet?

-este o versiune ordonata a HashSetului care foloseste o lista dublu inlantuita intre elemente pt a le mentine ordinea

Laborator 8

1.Ce este un model de proiectare?

-este Solutia unei problem intr-un context particular

În ingineria software, un model de proiectare este o soluție generală repetabilă la o problemă care apare frecvent în proiectarea software-ului.

Un model de design este o descriere sau un șablon pentru modul de rezolvare a unei probleme care poate fi utilizat în multe situații diferite.

2.Cum a aparut termenul de model de proiectare?

A aparut in urma studiilor facute de Cristofor Alexander care era architect si cauta noi mainiere pt proiectarea cladirilor,zonelor urbane.El spunea ca fiecare forma poate fi vazuta in acel context simplist ca o regula cu 3 parti care explica relatiile existente in cadrul unui anumit context,problem sau solutie.

3.Cum a evoluat conceptul de design pattern?

1987-Cunningham si Back au realizat limbajul

1990 Gasca celor 4 au realizat catalogul

1995 Gasca celor 4 au realizat o carte

4.Explicati cele 3 tipuri de modele software?

Modelul conceptual-descriere in termini si conceptul domeniul aplicatiei

Modelul de proiectare-descrie proiectarea software folosind obiecte,mostenirea clasei

Modelul de programare-se folosesc constructii de limbaj pt a descrie forme,folosind intructiunile pt utilizator se poate obs o problema legata din faza conceptuala

5.Unde sunt utile aceste modele de proiectare?

-cand avem mai multe niveluri de abstractizare(cand vrem sa reutilizam anumite clase,cand avem nevoie de un nivel complex pt o aplicatie mai complicate)

6.Care sunt elementele unui model de proiectare si la ce se refera fiecare?

-nume-folosit pt identificarea acelui model

-problema-se descrie contextual sic and ar trebui folosit acel model

-solutia-descrie elementele care sunt prezente in model,relatiile dintre ele si responsabilitatile acestora

-consecintele-sune implicarea modelului respective,care sunt costurile,beneficiile

7.Ce este o arhitectura inchisa?

Subsistemul decide maniera de accesare

8.Ce este o arhitectura deschisa?

Orice sistem din dealear din managementul so poate apela orice component sau clasa din dealear de persistenta

9.Ce sunt modelele creationale?

-se refera la abstractizarea instantierii unui process si au 2 caracteristici principale:

-incapsuleaza informatiile despre clasele concrete care sunt folosite

-ascund maniera in care sunt create clasele si cum coopereaza ele intre ele

10.Ce este Fabrica de obiecte?

-avem o interfata pt mai multe produse de diverse tipuri care vine implementata in clasele concrete asociata produselor ,avem un creator si niste creatori concreti care ne asigura crearea acestor produse

11.Ce inseamna design patterns in singleton?

permite să vă asigurați că o clasă are o singură instanță, oferind în același timp un punct de acces global la această instanță.

12.Cum poate fi implementat singleton in Kotlin?

Declaram un obiect separate a carei unicitate o sa fie garantata

13.Modelul builder-explicat

-permite crearea de obiecte complexe pornind de la obiecte simple folosind o maniera incrementara

Utilizare: cand un obiect nu se poate crea intr-un singur pas

14.Care sunt principalele tipuri de sabloane?

-creational

-structural

-comportamental

15.Explicati modelul state?

-scopul sau este de a permite unui obisect sa isi modifice comportamentul atunci cand starea sa interna se schimba

Utilizare: atunci cand un obiect depinde de starea sa si tr sa isi schimbe comportamentul in timpul executiei in functie de starea respective, cand operatiile au declaratii conditionale compuse mari care depind de starea obiectului

16.Explicati modelul prototip

permite copierea obiectelor existente fără a face codul dependent de clasele lor.
.

17.Explicati modelul composite

-se refera la un grup de obiecte care sunt tratate in acelasi fel, ca si cum ar fi o singura instanta a unui aceluiasi obiect

-- **permite să compuneți obiecte în structuri tree și apoi să lucrați cu aceste structuri ca și cum ar fi obiecte individuale.**

18.Explicati modelul momento

Capturați și restaurați starea internă a unui obiect

19.Explicati modelul observer

-face o relatie de dependent,cand un obiect isi schimba starea si celelalte obiecte care depinde de el se schimba

20.Explicati modelul lant de responsabilitati

permite să transmitem cereri de-a lungul unui lanț de manipulatori. La primirea unei cereri, fiecare gestionar decide fie să proceseze cererea, fie să o transmită următorului gestionar din lanț.

Laborator9:

1.Ce sunt modelele creationale?

-se refera la abstractizarea instantierii unui proces si au 2 caracteristici principale:

-incapsuleaza informatiile despre clasele concrete care sunt folosite

-ascund maniera in care sunt create clasele si cum coopereaza ele intre ele

2.Explicati fabrica de obiecte

-avem o interfata pt mai multe produse de diverse tipuri care vine implementata in clasele concrete asociata produselor ,avem un creator si niste creatori concreti care ne asigura crearea acestor produse

3.Explicati fabrica de fabrica de obiecte

Creează o instanță din mai multe familii de clase

4. Ce este singleton?

O clasa care permite o instant

5. Ce face modelul constructor?

Utilizare:cand nu poti face un constructor dintr-un singur pas,cand nu putem sa creem mai multi constructori intr-un anumit obiect

6.Explicati modelul prototip

Modelul creaza o copie a unui obiect existent prin intermediul clonarii

Utilizare: cand crearea unui obiect este costisitoare si avem deja un obiect creat

7.Cu ce se ocupa modelele structurale?

Modelele structurale explică cum să asamblați obiecte și clase în structuri mai mari, păstrând în același timp aceste structuri flexibile și eficiente.

8.Explicati modelul bridge

-permite să împărțiți o clasă mare în două ierarhii separate - abstractizare și implementare - care pot fi dezvoltate independent unul de celălalt.

9.Explicati modelul composite

permite să compuneți obiecte în structuri tip arbore și apoi să lucrați cu aceste structuri ca și cum ar fi obiecte individuale.

-utilizat acolo unde aplicatia trebuie sa manipuleze/prelucreze o colectie ierarhica de obiecte primitive si composite

10.Explicati modelul decorator

permite să atașați comportamente noi la obiecte, plasând aceste obiecte în obiecte speciale care conțin comportamente.

11.Explicati modelul proxy

- **Un obiect care reprezintă un alt obiect**

- accesul la obiectul original, permițându-vă să efectuați ceva înainte sau după ce solicitarea ajunge la obiectul original.

Ce tipuri de proxy exista: remote,virtual,de protective,smart-referance

12.Ce sunt modelele comportamentale?

- **modelele de proiectare comportamentală sunt modele de proiectare prepocupate de algoritmi și de atribuirea responsabilităților între obiecte**

13.La ce este folosit generatorul?

Generatoarele reprezintă o modalitate simplă de a crea iteratori, toate cerințele anterioare fiind gestionate în mod automat de generator.

Definirea unui generator presupune utilizarea cuvântului cheie *yield* în loc de *return*. Acesta trebuie să conțină cel puțin un *yield* (poate conține mai multe alte yield-uri, return-uri).

Diferența între *return* și *yield*:

- *return* - încheie complet execuția funcției
- *yield* - pune pauză funcției, salvând stările și continuând mai târziu de unde a rămas.

Avantajele utilizării unui *generator* în loc de *iterator:*

- ușor de implementat
- eficient din punct de vedere al memoriei
- permite reprezentarea unui ***stream*** **infinit**

- generatoarele pot fi folosite pentru a realiza un **pipeline** cu o serie de operații.

-folosit pt a controla un iterator si va gestiona in mod automat parcurgerea

14.La ce sunt folositi iteratorii?

-ca sa itereze printr-un set de obiecte

-cu iteratorii facem niste pargurgeri care pot fi dupa anumite criterii

15.Care sunt functiile generatorului?

Yeld-pune pauza functiei,salvand starile si continuand de unde a ramas mai tarziu

Laborator10:

1.Ce este calculul paralel?

Executia in paralel pe mai multe procesoare a acelorasi instructiuni sau a unor instructiuni diferite cu scopul rezolvarii mai rapide a unor problem

2.Ce este concurenta?

-se refera la faptul ca mai multe procese trebuie sa fie executate in acelsi timp pe mai multe resurse

3.Care este ierarhia de control la nivelul Kotlinului?

1corutine-uri la nivel user : planificare nivel user

2thred-uri la nivel interpretor: planificare la nivelul masinii virtuale

3thred/process la nivel OS: planificare la nivel Kernel OS

4thred/process la nivel Up: planificare la nivel Miroprocesorului

4.Ce inseamna memorie comuna?

-mai multe procese/threduri/corutine impart un spatiu de memorie si comunica prin a schimba valori din spatial comun de memorie

5.Ce inseamna transfer de mesaje?

Prin intermediul unei cozi de mesaje mai multe threduri pot comunica intre ele

6.Explicati concurenta la nivel de date

Putem procesa acelasi tip de date pornind de la aceeazi zona de date

Ce processor avem: ……..

7.Ce inseamna parallelism la nivelul datelor?

8.Ce inseamna granularitate?

-se refera la legatura dintre partea software si hardware a unui program,sa avem continuitate

9.Care sunt nivelurile de granularitate?

-granularitate cod: entitate cod,granularitate mare(nivel task),program

-granularitate medie: (nivel control),functie(corutina/thred)

-granularitate fina: (nivel date),bucla

-granularitate foarte fina: (alegeri multiple),cu suport hard

10.Ce inseamna acronimul CPS?

Continuation Passing Style

-in mod normal o functie ar trebui sa returneze un rezultat dar in cazul CPS functia ar trebui sa primeasca un parametru nou care sa faca cumva sa continue programul

11.Cum s-a ajuns la cuvantul de corutine in kotlin?

12.Ce este stilul transferului continuarii in kotlin?

Fiecarei functii I se adauga un argument suplimentar utilizat pt continuarea, I se modifica corpul functiei si ca sa intoarca o valoare, ea o transfera catre argumentul de continuare

13.Ce face automatul pt analiza de cod?

14.Ce inseamna terminare sincrona?

15.Ce este o corutina?

-kotlin are o abordare hibrida,au fost prevazute mecanisme automate de planificare dar se da si libertatea de a modifica algoritmul,avem concurenta flexibila

16.Care sunt starile din ciclul de viata a unei corutine?

17.Ce stil de programare functionala suporta kotlin?

Recursivitatea coada(tail recursion)

18.Cum se executa apelurile recursive in kotlin in cazul rezursivitatii coada?

Calculele sunt executate primele,apoi apelurile recursive(apelul recursive trimite rezultatul pasului current catre urmatorul apel recursive)

Laborator 11

1.Care este rolul jurnalizarii?

Putem pastra informatii despre cum functioneaza un program , de ex putem pastra informatii despre exceptii pt a depana programul.

2.Cum se face un logging mai serios?

E bine sa se jurnalizeze si exceptiile aparute care nu sunt tratate.

3.Ce inseamna acronimul GIL?

4.Ce functionalitati ne ofera modulul trading din python?

-ofera functionalitatile primare si exista niste probleme la nivelul acestor functionalitati primare.

Probleme care se refera la:

5.Cum arata un fir de executie in python?

-este o clasa a carei instante are urmatoarele proprietati: grupul,tinta,numele,argumentele si k argumentele(dictionar cu argumentele folosita de functia tinta)

6.Cum determinam treadul curent?

7.Ce inseamna lock?

8.Ce inseamna rlock?

9.Ce este un semafor?

Este un tip de data abstract care este gestionat de sistemul de operare si este utilizat pt a se furniza accesarea sigura a mai multor treaduri la resurse sau la date

10.Ce este un fir cu conditie?

Un mecanism de sincronizare care face un fir sa astepte indeplinirea unei conditii specific iar un alt fir il anunta ca acesta a fost indeplinit

11.Ce este un eveniment?

Evenimentele sunt obiecte care sunt utilizate pt comunicarea intre fire

12.Ce inseamna fir cu eveniment?

Un fir asteapta un semnal in timp ce celalalt fir il genereaza

13.Ce proprietati are un obiect de tip eveniment?

Un obiect de tip gestioneaza variabila care poate fi initializata cu true cu ajutorul lui set sau cu false

14.Cum utilizam cozile pt comunicarea intreprocese?

15.Cum realizam sincronizarea proceselor?

-lock-are 2 stari de blocare si eliberare

-event-ne permite o comunicare simpla intre procese

-condition-folosit pt a asigura o ordine de executie secventiala sau parallel proceselor

-semaphore-utilizat in versiunea unor resurse commune

-rlock-identic cu cel de treading

-barrier-puncte de sincronizare commune la nivelul tuturor treadingurilot de cate ori e necesar

16.Ce este event si cum functioneaza?

event-ne permite o comunicare simpla intre procese

17.Ce este barrier si cum functioneaza?

-barrier-puncte de sincronizare commune la nivelul tuturor treadingurilot de cate ori e necesar

18.Ce rol are condition si cum functioneaza pt procese?

-condition-folosit pt a asigura o ordine de executie secventiala sau parallel proceselor

Laborator 12

1.Care este ipoteza Church?

Fiecare functie poate fi real calculate sau orice predicat decisional este recursive in general

2.Care este ipoteza Turing?

Fiecare functie poate fi privita ca fiind calculabila poate fi calculate cu ajutorul unei masini Turing(functie recursive ~1930)

3.Ce este functia anonima?

Este o functie definite si apelata fara sa fie legata de un identificator

4.Ce clasa de obiecte matematice descriu ipotezele descries de Turing si Church?

clasa funcțiilor definibile lambda (a numerelor întregi pozitive) este identică cu clasa a funcțiilor recursive (a numerelor întregi pozitive) și la clasa funcțiilor calculabile (a numerelor întregi pozitive)

5.Care este seminificatia lui lambda?

Termen introdu de Church in 1930 pt a define cee ace numeste el functie

Utilizata: pt scrierea mai compacta a codului

Lambda-un model universal de calcul care poate fi folosit pt a simula orice masina Turing

6 .Care sunt limitarile java in cazul calculului functional?

-se evita o serie de operatiuni clasice:

***crearea unei instante a interfetei**

***suprascrierea metodei**

7.Ce tip de date au functiile in kotlin?

8.Ce este o functie de nivel superior in kotlin?

Accepta alte functii ca parametru sau intoarce alte functii ca rezultat sau ambele situatii simultan

9.Ce este paramestrul alias?

Aliasurile de tip oferă nume alternative pentru tipurile existente. Dacă numele tipului este prea lung, puteți introduce un alt nume mai scurt și îl puteți folosi pe cel nou.

10.Ce inseamna vararg?

Putem sa pasam pt o functie un nr variabil de argumente

11.Ce este o functie pura?

functie la care valoarea rezultata depinde de parametrii acesteia

Laborator 13

1.Care este ipoteza Turing?

Fiecare functie poate fi privita ca fiind calculabila poate fi calculate cu ajutorul unei masini Turing(functie recursive ~1930)

2.Care este ipoteza Church?

Fiecare functie poate fi real calculate sau orice predicat decisional este recursive in general

3.Care este seminificatia lui lambda?

Termen introdu de Church in 1930 pt a define cee ace numeste el functie

Utilizata: pt scrierea mai compacta a codului

Lambda-un model universal de calcul care poate fi folosit pt a simula orice masina Turing

4.Ce se intampla in urm secventa?

```
true=lambda x,y : x
false=lambdax, y:y
iff = lambda p, x, y : p(x,y)
```

True-intoarce mereu prima variabila

False-intoarce mereu a doua variabila

5.Ce fel de functii avem in general in programarea functionala?

Functii de prim nivel

Functii de nivel superior

6.Ce inseamna functie de prim nivel in python?

Pot fi vazute ca un obiect , pot fi utilizate ca obiecte de alte functii

7.Ce inseamna functie pura in python?

functii la care valoarea rezultata depinde in totalitate numai de argumentele sau parametrii ei si ea nu va avea efecte laterale

8.Ce inseamna functie de nivel superior?

Functii care accepta alte functii ca argumente si/sau intorc alte functii

9. Ce inseamna date inutemant in python?

Dupa ce au fost create nu pot fi schimbate

10.Ex de impacheteaza, proceseaza si despacheteaza in python?

1.Ce face functia map?

Functia map(din kotlin) este o functie a colectiilor. Aceasta primeste a argument o colectie(colectia prin care e apelata metoda map, o primeste ca this) si o functie lambda returneaza o lista cu fiecare dintre elementele colectiei originale transformate de functia lambda.

2.Ce sunt generatorii recursive?

Sunt functii ce creeaza iteratori intr-un mod recursiv. Acestia au forme identice cu cele ale functiilor recursive, diferenta fiind ca se foloseste yield in loc de return. La fel ca la generatorii nerecursivi, starea functiei este salvata de la un apel la altul. Functiile creeaza toate valorile de-o data pe cand generatorii creeaza cate o valoare, pe rand, astfel fiind mai eficienti din punctul de vedere al consumului de memorie.

3.Cum arata un when si ce face?

When (in Kotlin) este similar cu switch din Java, C, C++ etc. Se foloseste cam asa:

```
when(variabila){
        valoare1->{instruct}
        valoare2->{instruc}
        …
        else->{instruct}
}
```

4.Ce inseamna efect lateral?

Efectele laterale(sau secundare) constau in modificarea datelor de intrare/datelor globale prin utilizarea de functii lambda.

5.Deseneaza UML-ul pentru sablonul builder si ce face.

Sablonul builder contine o interfata Builder care are metodele build_part(...) si get_result():ObiectConstruit. Aceasta este implementata de minim 1 builder concret care defineste cele 2 metode. Optional ar mai putea exista si o clasa Director ce se afla intr-o relatie de agregare cu interfata Builder si, prin mai multe instante de builderi concreti, creeaza un obiect. Sablonul builder se foloseste cand un obiect e prea complicat de construit dintr-o data si se prefera construirea lui etapizata, incrementala.

6.Ce este coerenta datelor?

Coerenta datelor reprezinta faptul ca datele au relevanta si nu au fost corupte prin scrierea mai multor thread-uri/procese asupra lor, in acelasi timp. Tot pierderea coerentei datelor se intampla si cand un proces/thread citeste date in acelasi timp cu altul care scrie peste acele date. Coerenta datelor se pastreaza prin metode de sincronizare: semafoare, bariere, monitoare, tipuri de date atomice(pe care le poate accesa doar 1 proces/thread la un moment dat), mutex-uri, conditii, actori(in Kotlin).

7.Ce este list comprehension?

Este o metoda de creare a listelor din alte obiecte iterabile(in Python). Exempl2:

lista1=[x for x in obiect_iterabil if x%2==0 and x>33]

lista2=[(x,y) for x in obiect_iterabil_1 if x%2==0 for y in obiect_iterabil_2 if y%2!=0]

8.Explica modelul Strategy.

Modelul Strategy incapsuleaza o serie de algoritmi ce au o interfata comuna. Acesta are o interfata numita Strategy. Aceasta este implementata de mai multe “strategii” concrete ce incapsuleaza diferiti algoritmi pentru rezolvarea uneia sau mai multor probleme. O clasa primi anumite task-uri iar, in functie de tipul acestora, ele sa fie delegate unui obiect de tip Strategy.

9.De ce se folosesc exceptiile custom?

Exceptiile custom se folosesc pentru incapsularea mai buna atat a cauzelor cat si a contextului aparitiei exceptiei. Acestea sunt astfel mai specifice si se mareste sansa ca ele sa fie tratate in locul inchiderii bruste a programului.

10.Ce este o functie extensie?

O functie extensie este o functie ce se poate adauga unei clase fara modificarea ei/crearea unei subclase ce contine acea functie/utilizarea delegatior(din Kotlin)/utilizarea de modele de proiectare precum Visitor si Decorator. Acestea sunt utile in cazul in care nu avem acces la codul sursa a claselor pe care vrem sa le modificam. Cu ajutorul lor putem evita complet utilizarea modelului de proiectare Visitor, scapand astfel de scaderea incapsularii si cresterea cuplarii aduse de el.

11.Explica dependenta dintre 2 clase.

Dependenta dintre 2 clase consta in modul in care acestea interactioneaza. Aceasta se mai numeste si cuplare(defapt cuplarea este o metrica ce cuantifica dependenta tuturor claselor de o singura clasa dar vorbind mai putin general se poate reduce la dependenta dintre 2 clase). Cu cat o clasa apeleaza mai multe metode/se foloseste de mai multi membri ai unei alte clase, cu atat dependenta dintre cele 2 clase este mai mare. Dependenta dintre clase ar trebui sa fie cat mai mica deoarece, in cazul in care este mare, schimbarile dintr-o clasa se propaga si in clasele ce depind de aceasta.

12.Ce este principiul lui Demeter?

Principiul lui Demeter spune ca un obiect nu ar trebui sa cunoasca detalii despre membrii altor obiecte. Prin intermediul lui se reduce cuplarea(obiectele nu pot sa depinda de lucruri pe care nu le cunosc) si se imbunatateste incapsularea(membrii unui obiect sunt folositi doar de el insusi, astfel el ii incapsuleaza perfect).

13.Ce este EDP(Event Driven Pattern)?(Event driven programing probabil)

EDP este prescurtarea de la Event Driven Programing. Aceasta este o paradigma de programare, de cele mai multe ori folosita la aplicatiile cu interfata grafica. Principiul dupa care functioneaza este: se asteapta un eveniment si apoi acesta se trateaza.

14.Modelul Observer.

Modelul Observer este un model de proiectare comportamental care creeaza un mecanism prin care anumite obiecte(“subiecti”) notifica faptul ca si-au schimbat starea unor alte obiecte(“observers”). Modelul Observer are o interfata numita Observer ce are metoda Update. Aceasta interfata este implementata de minim 1 Observer concret. Mai exista si o clasa abstracta numita Subject care detine o colectie de Observeri si are metodele add(o:Observer), remove(o:Observer) si notifyAll(). Modelul Observer poate fi de 2 feluri: de tip Push sau Pull. La cel de tip Push, metoda Update primeste ca argumente datele de interes pentru observers

iar in cazul Pull, metoda Update nu primeste argumete ci doar semnaleaza observatorilor ca subiectul si-a schimbat starea.

15.Modelul Composite + diagrama UML.

Modelul Composite este un model structural ce permite compunerea obiectelor in ierarhii arborescente iar toate obiectele din ierarhie putand fi manipulate uniform. Se porneste de la o interfata Component ce incapsuleaza numele metodelor comune(comportamentul comun) al obiectelor din ierarhie. Din aceasta se deriveaza o clasa Frunza(care ar putea fi la fel de bine o abstractiune) care este un obiect din ierarhie ce nu are descendenti. Interfrata Component mai este implementata si de o clasa(de cele mai multe ori abstracta) numita InternalNode ce contine o colectie de obiecte de tip Component. Clasa InternalNode/subclasele ei sunt obiecte ce au descendenti in ierarhie.

16.Diferenta dintre tipurile generice si tipurile algebrice.

Tipurile generice sunt tipuri parametrizabile ce pot gestiona anumite tipuri de date. De exemplu List poate gestiona orice tip de data, SortedSet poate gestiona date de tipul Date:Comparable<Date>(daca nu se primeste in constructor un comparator).

17.Granularitatea firelor de executie.

Granularitatea firelor de executie reprezinta cantitatea de calcule realizata de un fir de executie. Aceasta poate fi de 2 tipuri: mica/fina si mare. In cazul celei mici un program este impartit foarte multe task-uri mici, acestea fiind distribuite unui numar mare de thread-uri. In cazul celei mari, un program este impartit in cateva task-uri de dimensiuni mari.

18.Model Prototype +_UML.

Modelul Prototype este util cand un obiect este greu de construit si se doreste construirea unui prim obiect(prototip) care mai apoi sa fie clonat iar clona sa fie particularizata dupa nevoi. Se porneste de la o interfata Prototype care are metoda clone():Prototype. Toate clasele ce implementeaza acea interfata au posibilitatea de a se clona, putand fi folosite ca obiecte de tip prototip.

19.Ce este UML?

UML este prescurtare de la Unified Modeling Language. Acesta consta intr-un set de tipuri de diagrama ce ajuta la proiectarea, intelegerea, vizualizarea, documentarea si evaluarea sistemelor software. Contine mai multe tipuri de diagrame: de clase(prezinta ierarhia de clase), de obiect(arata un snapshot a cum sunt instantiate obiectele la un anumit moment de timp in executia programului), de secventa(prezinta cum interactioneaza obiectele in cazul anumitor inputuri utilizator), de pachete(arata cum interactioneaza pachetele) etc.

20.Ce librarie din python pentru calcul paralel duce la cel mai bun timp de executie?

Din cate am vazut in lab 11/exemplul 1, modulul ar fi multiprocessing

21.Ce intelegi prin reutilizarea abstractizarii?

Prin reutilizarea abstractizarii se intelege utilizarea claselor abstracte ca superclase pentru clasele ce au comportament comun, ne copiind aceleasi metode/membrii in fiecare clasa. O alta insemnatate a reutilizarii abstractizarii ar fi segregarea interfetelor claselor in

interfete mai mici si implementarea mai multora din aceste interfete de catre noile clase.

22.Ce sunt corutinele? Ce difera in raport cuun thread? De ce pot apela 1mil de corutine, dar nu si 1mil de threaduri?

Corutinele seamana cu thread-urile numai ca sunt mult mai light-weight. Acestea sunt secvente de instructiuni ce se executa pe thread-uri separate, in mod concurent. Acestea pot porni pe un thread si a se termina pe un altul. Ele difere de threaduri prin modul in care sunt puse in asteptare: cand o corutina asteapta(face un simplu sleep, asteapta sa intre intr-o regiune critica etc) aceasta este scoasa din thread-ul pe care rula, este pusa pe stiva si lasa liber thread-ul pentru alte corutine care sunt disponibile sa ruleze in acel moment(sunt "ready"). Threadurile cand sunt puse in asteptare se blocheaza complet, ele nefiind utile pana la iesirea din starea de asteptare si consumand mai multa memorie. Nu stiu sigur de ce poti crea 1kk corutine dar nu si threaduri in afara de faptul ca threadurile consuma mult mai multa memorie decat corutinele.

23.Ce este principiul dependentei inverse?

Principiul dependentei inverse spune ca clasele din etajele superioare ale ierarhiei nu trebuie sa depinda de cele din etajele inferioare, ci de abstractiuni. Practic se spune ca abstractiunile trebuie sa depinda de alte abstractiuni iar clasele concrete trebuie sa depinda tot de abstractiuni.

Abstractiuni=clase abstracte si interfete

24.Principiul Deschis/Inchis. (SOLID).

Principiul inchis/deschis spune ca o clasa trebuie sa fie inchisa pentru modificari si deschisa pentru extindere. Cu alte cuvinte, o clasa nu trebuie sa fie modificata. Daca se doreste modificarea ei totusi, e mai bine sa o derivam iar clasa derivata sa aduca modificarile dorite. Astfel nu o sa se propage modificari in codul ce depindea de clasa initialia, cea fara modificari.

25.Dependente aciclice.

Dependenta aciclica este o dependenta intre pachete/clase(in curs am vazut doar despre pachete) in care nu se gasesc bucle. Astfel, printr-o sortare topologica a pachetelor din graful de dependenta se poate creea o ordine de proiectare, implementare si testare a pachetelor(se vor creea intai pachetele care nu depind de alte pachete, apoi cele care depind de cele deja creeate si tot asa).

26.Deadlock.

Deadlock-ul (situatie de blocaj) este atunci cand unele procese/thread-uri detin anumite resurse critice si asteapta eliberarea altor resurse critice intr-o maniera circuluara. De exemplu thread-ul A are resursa X si vrea resursa Y iar thread-u B are resursa Y si vrea resursa X(o sa se astepte unul pe altul la infinit). Astfel de situatii se previn utilizand mijloace de sincronizare sau asigurarea ca un thread/proces detine la un moment dat cel mult 1 resursa critica.

27.Ce este bariera?

Bariera este o metoda de sincronizare prin care se asigura oprirea mai multor thread-uri intr-un punct pana la sosirea tuturor thread-urilor in acel punct, dupa care se permite

avansarea taskurilor fiecarui thread. Acest lucru este necesar cand continuarea task-urilor din acel punct necesita terminarea unor calcule(a tuturor calculelor realizate de fiecare thread pana in acel punct).

28.Ce este o functie lambda?

O functie lambda este o functie anonima care este caracterizata prin tipul parametrilor sai si tipul sau de return. Acestea retin contextul in care sunt creeate. Ele au acces la variabilele si metodele din contextul in care au fost creeate. Acestea le pot folosi chiar daca sunt apelate inafara acelui context, loc de unde acele metode/membri nu s-ar putea folosi.

29.Desenare Model View Controller+ explicare.

Model View Controller este un model de proiectare ce are 3 componente: Model, View si Controller. Model contine starea interna a programului, primeste informatii de la Controller(evenimente utilizator) si updateaza View(interfata grafica). Controller este componenta ce primeste input de la utilizator si il transmite lui Model. View este componenta ce se ocupa de afisare. Aceasta contine Look and Feel-ul aplicatiei. Look se refera la cum arata aplicatia iar Feel la functionalitatile aplicatiei.

30.Ce este coesiunea dintre clase?

Coeziunea este o metrica aplicabila oricarei clase. Aceasta arata daca o clasa realizeaza/incapsuleaza prea multe notiuni sau nu. Coeziunea cuantifica puterea legaturii dintre metodele unei clase si membrii acesteia. Este in curs o formula pentru calculul ei. Coeziunea se doreste a fi cat mai mare.

31.Ce face itertools.tee si ce erori poate provoca

Itertools.tee cloneaza un iterator. Nu stiu ce erori poate provoca dar stiu ca rezolva problema consumarii unui iterator prin clonarea starii initiala a acestuia. Astfel se obtin 2(sau mai multi iteratori) ce pot fi parcursi separat. Consumul de memorie este mai mic decat existenta unei liste care sa fie parcursa de mai multe ori deoarece un iterator returneaza cate 1 element, pe rand.

32.Functia pura.

Functiile pure sunt functii ce nu modifica parametrii de apel, nici variabilele globale. Despre functiile lambda care sunt pure se poate spune ca sunt functii fara efecte secundare.

33.Liste algebrice.

???

34.Explicatie pentru any si all.

Functiile cele 2 sunt specifice colectiilor.

Kotlin:

-Functia any returneaza true daca exista minim un element care satisface functia lambda data ca parametru(functie care trebuie sa returneze Boolean si sa primeasca ca parametru un element al colectiei), false in rest

-Functia all returneaza true daca toate elementele colectiei satisfac predicatul dat ca parametru(functia lambda)

Python:Cele doua functii se aplica colectiilor iterabile de Boolean

-Functia any returneaza True daca exista cel putin un element True in colectie, False in rest

-Functia all returneaza True daca toate elementele din colectie sunt True, false in rest

35.Substitutia Liskov.

Subclasele trebuie sa poata fi manipulate la fel ca superclasele de catre clienti. Subclasele nu trebuie sa modifice comportamentul mostenit de la superclase ci trebuie sa adauge noi functionalitati.

36.Ce face un semafor

Semaforul permite accesul intr-o regiune critica a unuia sau mai multor procese/thread-uri. Acesta are o stare initiala caracterizata printr-un intreg notat de Dijkstra cu s. Semaforul permite unui numar de procese sa intre in regiunea critica egal cu starea sa initiala. Daca un proces/thread incearca sa intre in regiunea critica si semaforul nu ii da voie, acesta ramane in asteptare pana la iesirea unui thread/proces din regiunea critica.

37.Este immutabilul garantat in kotlin

Din cate am vazut in curs, nu.

38.Proxy

Proxy este un model de proiectare structural ce vine cu un substitut al unui obiect. Substitutul poate reglementa accesul la obiect, poate creea obiectul doar in cazul in care e nevoie de el sau poate modifica modul in care clientul comunica cu obiectul intermediat. Se pleaca de la o interfata comuna obiectului intermediar(proxy) si obiectului intermediat. Apoi se creeaza clasa ce reprezinta obiectul intermediat. Dupa, se creeaza clasa obiectului intermediar, acesta contine o referinta la un obiect de tip intermediat caruia ii deleaga sarcinile.

39.Sa fol un when intr-un case

Cred ca vrea sa ceara sa folosesti un when ca un switch-case in Kotlin, este asa ceva mai sus.

40.Ce e un thread cu o conditie

Un thread cu o conditie este un thread care, pentru a-si putea continua task-urile, o data sau de mai multe ori pe parcursul executiei sale, acesta asteapta ca obiectul de tip conditie pe care il detine sa fie semnalat(adica face niste treaba, dupa apeleaza condition.acquire() pentru a astepta sa i se spuna ca isi poate continua treaba)

41.Ce este o bariera

Mai sus...

42.Arhitectura inchisa-deschisa

Arhitectura deschisa este o arhitectura care poate fi extinsa, al carei cod sursa ar putea fi cunoscut. Arhitectura inchisa este o arhitectura ce nu mai poate fi dezvoltata. Componentele

ei sunt de cele mai multe ori necunoscute si se lucreaza in mare parte cu interfete ale lor, ba chiar cu Fatade a subsistemelor din arhitectura. Acestea simplifica utilizarea componentelor si reduc riscul de a le utiliza in mod eronat.

43.Prototype diagrama

Mai sus...

44.Procesul impacheteaza /despacheteaza

Procesul functioneaza in felul urmator: datele se impacheteaza intr-un mod convenabil, se proceseaza iar apoi se despacheteaza in vederea obtinerii rezultatului final. Impachetarea are rolul de a schimba structura de organizare a datelor, facandu-le mai usor de procesat. De exemplu in Python, folosind modului Functional:

```
lista=[1,2,3,4]

new_list=seq(lista).map(lambda it:it*it).filter(lambda it: it>=7).list()
```

seq(lista)=impachetarea listei intr-o Sequence

.map, .filter=procesarea

.list=despachetarea

(sincer nu stiu sigur sigur daca la asta se refera procesul dar eu asa am inteles)

45.Coeziunea claselor

Mai sus...

46.Cu ce seamana decoratorul

Decoratorul seamana atat cu Composite cat si cu Visitor. Decorator este un Composite in care nodurile interne au un singur fiu. Cu Visitor seamana deoarece imbogateste functionalitatile clasei. Totusi, Visitor adauga noi functionalitati pe cand Decorator le “decoreaza” pe cele deja existente.

47.Generator recursiv

Mai sus...

48.Ce e un actor

Actorii sunt entitati utilizate in calculul concurent in Kotlin. Acestia sunt caracterizati de un canal de mesaje si de tipurile de mesaj pe care le poate primi pe canal. Acesta primeste in mod asincron mesaje prin canalul de comunicatii si le trateaza in ordinea primirii.

49.DIFERENTA INTRE MAP SI FLATMAP

Map este o functie a colectiilor ce returneaza o colectie obtinuta prin aplicarea functiei date ca parametru peste fiecare din elementele colectiei initiale. Flat map face acelasi lucru doar ca, la final, concateneaza colectiile din colectia rezultata. Adica, daca din map ar fi rezultat List<List<Int>>, flat map ar concatena listele de intregi din lista finala si ar rezulta List<Int>. Totusi, flatmap nu functioneaza chiar cum ne-am dori in cazul colectiilor arborescente. De exemplu, daca la final ar rezulta List<List<List<Int>>>, flatmap ar face ca rezultatul sa fie List<List<Int>>. Daca se doreste ca rezultatul sa fie

List<Int>, ar trebui sa se foloseasca functii precum compress din Python(nu stiu daca are echivalent in Kotlin)

50.Fatada

Modelul fatada este un model de proiectare structural care asigura o interfata simplificata unui sistem. Aceasta ascunde implementarea interna a sistemului(componente, metode globale etc) si face ca arhitectura sistemului reprezentat sa fie inchisa. Fatada simplifica operarea sistemului si reduce riscul de a fi folosit gresit. Totusi, impiedica dezvoltarea sistemului.

51.Coerenta datelor

Mai sus...

52.Pick corelation

Reprezinta procesul prin care o aplicatie selecteaza ce fereastra gestionata de aceasta sa trateze un anumit eveniment.

53.Diferenta dintre python si c++

Prea multe… Ar fi fost fain sa fie mai specific:))

54.Caz utilizare proxy

-Cand se doreste controlat accesul la obiectul intermediat

-Cand s-ar putea sa nu fie nevoie de obiectul intermediat si astfel acesta sa fie creeat doar la nevoie

-Cand Proxyul este mic sau usor de folosit si obiectul intermediat este fix invers

-Cand nu exista acces direct la obiectul intermediat, acesta fiind la distanta. Comunicarea complexa intre client si obiect intermediat este asigurata de Proxy

55.Diagrama builder

Mai sus...

56.Blocare

Probabil acelasi lucru cu intrebarea 26.(daca nu, pe scurt, procesul prin care un thread/proces asteapta ca unele date sa fie disponibile)

57.Interblocare

Aceeasi cu 26

58.S de la solid

S din SOLID vine de la principiul responsabilitatii unice. Acesta spune ca o clasa trebuie sa aiba o singura responsabilitate, un singur motiv pentru care sa se schimbe. Clasele ce au mai multe responsabilitati trebuie sparte in clase mai mici, fiecare din ele cu cate 1 singura responsabilitate.