O exceptie - o solutie anormala ce apare in executii si care poate avea cauze hardware sau software. Exceptiile pot fi private ca evenimente previzibile ce pot aparea in anumite puncte din program si care afecteaza continuarea programului.

Pot provoca exceptii:

-constructorul lui FileReader

-read

-close

O metoda daca nu trateaza o exceptie, o arunca.

Tratarea exceptiilor:

- Try-catch-finally
- Throw(aruncarea explicita de exceptii)

Avantajele tratarii exceptiilor :

- Separa codul
- Propaga erorile
- Grupeaza

Exceptie personalizata: de obicei o tratam cu mesaj, erorile dupa tip cu metoda printStackTrace() din clasa Exception(sau cu Writer sau Printer)

Dependenta dintre doua clase:

- O relatie foarte clara intre doua clase(nu e implementata prin variabile membre)
- Poate fi implementata prin intermediul argumentelor unei metode

O extensie este interiorul sau tipul de date continute in fisier, este separata de numele fisierului printr-un punct .

Principiul sau legea lui Demeter: - principiul de cunostinte putine se refera la dezvoltarea de software, este un caz specific de cuplare libera(constiinte limitate).

Uml(Unified Modelling Language) reprezinta un limbaj graphic pentru vizualizarea, specificarea, dezvoltarea si documentarea componentelor unui sistem software de dimensiuni medii sau mari

4 entitati UML:

- Structurale
- Comportamentale
- Pentru grupare
- Pentru observatie

EDP(Event Driven Promming)

Un eveniment – un tip de semnal pentru program, ii spune programului ca s-a intamplat ceva.

Tratarea event:

- Polling
- Intrerrupt driven
- Event driven

Paradigma orientata eveniment – sistemul asteapta evenimentele(generate de utilizator (in principal)) si acestea vor declansa (trigger) metodele de tratarea a lor.

Avantaje:

- Sunt portabile
- Permit tratarea mai rapida
- Se pot folosi de suportul sistemului de operare
- Incurajeaza reutilizarea codului

Generatoare: tastat, mouse;

Generatorii de even: metodele necesitatii tratarii evenimentelor.

Sablonul Observator – un obiect observant care sufera diverse modificari si unul sau mai multe obiecte observator care trebuie anuntate(notificate) imediat de orice modificare in obiectul observant pentru a realiza anumite actiuni.

Obiectul observat contine o lista de referinte la obiectele observator si la producerea unui eveniment, apeleza o anumita metoda a obiectelor observator inregistrate de el.

Collection – modeleaza o colectie

Map – descrie structuri de date de tip cheie – valoare, asociaza fiecarui element o cheie unica, dupa care poate fi regasit.(o interfata ce descrie colectiile)

Tipuri de **recursivitate**: - pe stiva, pe coada, arborescenta

- Directa, indirecta

Recursivitatea este proprietatea unor notiuni de a se define printre ele insele

Efecte laterale – o functie/expresie produce efecte laterale daca pe langa valoarea pe care o genereaza, mai are si alte efecte, modificari asupra starii programului.

Exp: Modificarea unor variabile/argumente

Citirea/scrierea din/in fisier

Apelarea altor functii care produc efecte laterale

Sablonul constructor(builder) – descrie crearea unor obiecte in regim pas cu pas, in conditiile in care procesul de creare este independent de structura interna a obiectelor.

Scop: Sa creeze instante ale altei clase

Builder- interfata pt crearea partilor unui obiect

Concret B: implementeaza interfata ;

Construieste partile obiectelor;

Director : construieste un obiect folosind interfata builder

Product : reprezinta obiectul complex care se construieste

Incapsularea algoritmilor(Strategy)

Presupune incapsularea separata a fiecarui algoritm dintr-o familie, facand astfel ca algoritmi respectivi sa fie interschimbati.

Un algoritm incapsulat in acest mod s.n strategie.

Sablonul se mai numeste si politica.

Sablonul strategy se aplica cand:

- Mai mute clase inrudite difera doar prin comportament
- Sunt necesare mai multe variante ale unui algoritm, care difera intre ele(de exemplu prin compromisul spatiu-timp adaptat).
- Un algortim util, date pe care clientul agloritmului nu trebuie sa le cunoasca
- Intr-o clasa sunt definite mai multe actiuni care apar ca structuri conditionale multiple. In loc de aceasta, se recomanda plasarea ramurilor conditionale inrudite in cate o clasa strategy separata.

List comprehension(intelegerea listei) – un mod elegant de a defini si de a crea liste pe baza listelor existente.

Sablonul Compozit – se aplica structuri ierarhice, orice componenta este compusa din alte componente. O structura compozita este ca un arbore general: elementul ‘’Leaf’’ este un nod frunza, iar elementul ‘’Component’’ este nodul interior, care poate fi privit ca radacina unui subarbore. Fiecare componenta(nod) poate fi tratata ca un obiect separat sau ca un grup de obiecte, folosind aceeasi interfata.

Prototype

Scop: face mai usoara crearea dinamica a obiectelor prin definirea de clase ale caror obiecte pot crea duplicate ale lor. Sablonul se foloseste cand cream un obiect care este o copie a unuia existent. Are ca argument un obiect care este folosit ca baza pentru a crea o noua instanta cu aceeasi stare.

Sablonul Model View Controller - Un obiect ‘’model’’ este un obiect observat(ascultat), care genereaza evenimente pentru obiectele receptor inregistrate la model.

MVC foloseste 3 clase principale:

- Controller – rol de a comanda a unor modificari in model ca urmare a unor evenimente externe.
- Model – rol de obiect observant cu date si prelucrarile datelor
- View – redarea vizuala a modelului, e obiect observator

Principiul dependentelor aciclice ADP – graful dependentelor intre pachete nu trebuie sa aiba cicluri

Principiul dependentelor stabile SDP – dependentele trebuie realizate in directia cresterii stabilitatii.

Corutine - o forma de procesare secventiala : doar una se executa la un moment dat, **pe cand thread**(firele) – e o forma de procesare concurenta: se executa mai multe fire in orice moment.

Corutinele se apeleaza pe rand.

Threadurile se pot executa mai multe deodata.

O functie pura este un analog de calcul al unei functii matematice. Are proprietatea ca valoarea returnata este aceeasi pentru aceleasi argumente, evaluarea sa nu are efecte secundare(nu exista mutari). Exp: floor, max

Liste algebrice – liste in care specificam forma fiecarui element. Sunt create prin operatii algebrice(sume si produse)

Sume- e alternanta(ambele sau nu)

Produse- e combinatie(semnificatie)

Cuplarea – dependenta de ceilalti, un mod/pachet/clasa/metoda se bazeaza pe alte module. Daca doua module comunica, ele trebuie sa schimbe informatii cat mai putine.

Coeziunea – completitudinea cu sine, este o legatura stransa intre membrii. Toate codurile dintr-o rutina sau o clasa sustin un scop central.

Un generator e o rutina(secventa de instructiuni) ce controleaza o iteratie(repetarea unui proces) a unei bucle.

Iterator – un obiect care reprezinta un flux de date(schimb de date cu mediul extern)

Itertools(returneaza n iteratoare independente dintr-un sg. Iterabil) **.** tee -> am iterator 1,2..n- imparte un iterator in n.

Odata ce tee() a fost impartit, iterabilul original nu ar trebui folosit in alta parte; in caz contrar ar putea avansa fara ca obiectele tee sa fie informate.

Erori – cand se utilizeaza simultan iteratoare returnate de acelasi apel tee(), chiar daca iterabilul original este thread safe.

Reutilizarea abstractizarii – reutilizarea functionalitatii unei clase.

Libraria Python pentru **calculul paralel** – cel mai bun timp – **multiprocessing**

Generice – permit utilizrea mai multor tipuri de date

When – e de fapt switch din C – este o instuctiune care inlocuieste mai multe if-uri ca si switch-ul din C si care verifica o variabila si mai multe cazuri si depinde de cazul in care sunt ca sa execute instructiuni corespunzatoare acestuia.

Extensii – metode pe care le pot crea unor tipuri de date

Ex: la un string din Kotlin o functie de extensie poate fi removeFirstLetter() si apoi pot sa o apelez in cadrul oricarui obiect de tipul string.

Generatori recursivi – ajuta la a nu pierde valoarea anterioara, daca apelez o functie dupa ce face return valoarea e pierduta, in sensul ca functia nu si-o poate aminti. Dar generatorii astia recursivi pana nu intalnesc yield inca mai retin rezultatul functiei anteriare.

Procesele sau taskuri(Unix si in MicrosoftWindows) sunt activitati paralele ale nivelului sistemului de operare.

Fire de executie(threads) – nivel de parallelism care are loc in cadrul fiecarui program(aplicatie). In java un fir de executie e un obiect dintr-o subclasa Thread.

Granularitatea firelor de executie)multithreading) – fina si aspra

- G. fina – fiecare instructiune va fi preluata de un alt fir de executie, astfel neavand doua instructiuni din acelasi fir de executie prezente in acelasi timp in pipeline.
- G. aspra – gstioneaza in mod similar firele de executie din pipeline insa fiecare noua instructiune poate deriva din orice fir de executie, independent de instructiunile aflate in pipeline.
    - G. medie(nivel de control-functie de corutina/thread)
    - G. fina(nivel date-bucla)

- G. f. fina(alegeri multiple- cu support hard)
- G. mare(nivel task-program)

O colectie este un obiect care grupeaza mai multe elemente intr-o sigura unitate.

Principiul deschis-inchis – entitatile software(clasele, modulele, functiile etc.) trebuie sa fie deschise in cee ace priveste extinderea, dar inchise in cee ace priveste modificarea.

Principiul Subst. Liskov – functiile care utilizeaza pointeri sau referinte la clase de baza trebuie sa poata folosi instante ale claselor derivate fara sa isi dea seama de acest lucru.

Principiul Dependentei Inverse

- Modulele de pe nivelurile ierarhice superioare nu trebuie sa depinda de modulele de pe nivelurile ierarhice inferioare. Toare ar trebui sa depinda doar de module abstract.
- Abstractizarile nu trebuie sa depinda de detalii. Detaliile trebuie sa depinda de abstractizare.

Lambda este o functie definita si apelata fara a fi legata de un identificator. Permit accesul la variabilele din domeniul functiei in care sunt. Ne permite sa cream instante ale claselor cu o sigura metoda inr-un mod mult mai compact.

Deadlock(blocaj)- e o stare in care fiecare membru al unui grup asteapta ca un alt membru, chiar si el, sa ia masuri, sa transmita un mesaj sau sa elibereze un blocaj. Apare cand un proces sau thread intra intr-o stare de asteptare, in principal din cauza semnalelor pierdute sau corupte.

O baiera de sincronizare este un mecanism pentru sincronizarea globala (a tuturor proceselor), este introdusa cand un proces trebuie sa le astepte pe celelalte, iar executia se reia cand toate procesele au atins bariera. Implementarea unei bariere se face prin diverse tehnici: liniara(bazata pe un contor centralizat), arbore, future. Cand o pereche de procese transmit si receptioneaza date intre ele, atunci apare ‘’deadlock’’-ul sau impasul.