2-MA’RUZA.

OBYEKT MODELI VA UNING AFZALLIKLARI.

OBYEKTGA YO’NALTIRILGAN DASTURLASH PRINSIPLARI.

Obyekt modeli va uning afzaliklari

Obyektga yo‘naltirilgan dasturlashning asosiy afzalligi shundaki, ularni boshqarish uchun ishlatiladigan ma'lumotlar ham, amallar (kod) ham bitta obyektga joylashtirilgan. Masalan, obyekt tarmoq bo‘ylab harakatlansa, u ma'lumotlar va xatti -harakatlarni o‘z ichiga olgan holda, to‘liq uzatiladi.

Obyekt. Obyektlar – obyektga yo‘naltirilgan dasturlarning qurilish bloklari hisoblanadi. Obyektga yo‘naltirilgan texnologiyadan foydalanadigan dastur asosan obyektlar to‘plamidir.

Obyekt ma'lumotlari. Obyektdagi ma'lumotlar uning holatini ko‘rsatadi. Obyektga yo‘naltirilgan dasturlash terminologiyasida bu ma'lumotlar ***atributlar*** deb ataladi.

Obyektlarning xatti-harakatlari. Obyektning xatti-harakati u bajara oladigan narsani ifodalaydi. Protsedurali tillarda xatti-harakatlar protseduralar, funksiyalar va qismiy dasturlar bilan belgilanadi. Obyektga yo‘naltirilgan dasturlash terminologiyasida obyektlarning xatti-harakatlari metodlarda mavjud bo‘lib, unga xabar yuborish orqali metod chaqiriladi.

Obyektga yo‘naltirilgan texnologiya ***obyektlar modeli*** deb ataladi.

Uning asosiy tamoyillari: abstraksiya, inkapsulyatsiya, modullik, iyerarxiya, tiplashtirish, parallellik va butunlilik. Bu tamoyillarning har biri haqiqatan ham yangi emas, lekin obyekt modelida ular birinchi marta birgalikda qo‘llaniladi. Birinchi to‘rtta tushuncha majburiydir, chunki ularning har birisiz model obyektga yo‘naltirilgan bo‘lmaydi. Boshqalar ixtiyoriy, ya'ni ular obyekt modelida foydali, lekin majburiy emas.

Obyekt modelining afzalliklari. Obyekt modeli strukturaviy tahlil, dizayn va dasturlashning an'anaviy usullari bilan bog‘liq bo‘lgan modellardan tubdan farq qiladi. Bu obyekt modeli ilgari topilgan va vaqt sinovidan o‘tgan barcha metodlardan voz kechishni talab qiladi degani emas. Aksincha, u oldingi tajribaga qo‘shadigan ba'zi yangi elementlarni taqdim etadi. Obyekt yondashuvi boshqa modellar bermagan bir qator muhim qulayliklarni ta'minlaydi. Eng muhimi, obyektga asoslangan yondashuv yaxshi tuzilgan murakkab tizimlarning xususiyatlarini rivojlantiradigan tizimlarni yaratishga imkon beradi. Obyekt modelining yana beshta afzalligi bor.

1) Obyekt modeli sizga obyektning imkoniyatlaridan to‘liq foydalanish imkonini beradi va dasturlash kabi obyektga yo‘naltirilgan.

2) Obyekt yondashuvidan foydalanish rivojlanishning birlashish darajasini va nafaqat dasturlarni, balki loyihalarni qayta ishlatish darajasini sezilarli darajada oshiradi, bu esa oxir-oqibat rivojlanish muhitini yaratishga olib keladi. Obyektga yo‘naltirilgan tizimlar, odatda, obyektga yo‘naltirilgan bo‘lmaganlarga qaraganda ancha ixchamdir. Va bu nafaqat dastur kodi miqdorini, balki avvalgi ishlanmalardan foydalangan holda, loyiha narxining pasayishini ham anglatadi, bu esa vaqt va xarajatdan foyda keltiradi.

3) Obyekt modelidan foydalanish barqaror oraliq tavsiflarga asoslangan tizimlar qurilishiga olib keladi, bu esa o‘zgarishlarni kiritish jarayonini osonlashtiradi.

4) Obyekt modeli murakkab tizimlarni ishlab chiqish xavfini kamaytiradi, chunki, birinchi navbatda, integratsiya jarayoni butun rivojlanish vaqtiga cho‘ziladi va bir martalik hodisaga aylanmaydi.

5) Obyekt modeli insonning dunyoni idrok etishiga qaratilgan yoki Robsonning so‘zlari bilan aytganda, "kompyuter qanday ishlashini bilmaydigan ko‘p odamlar tizimlarga obyektga yo‘naltirilgan yondashuvni mutlaqo tabiiy"[1] topadi.

Sinflar va obyektlarning aloqasi va o‘zaro ta'siri. Obyekt – bu xususiyatlarga ega bo‘lgan va uning xatti-harakatlarini ko‘rsatadigan haqiqiy nomli mohiyat.

Obyektga yo‘naltirilgan dasturlash tillariga qo‘llanilganda, obyekt va sinf tushunchasi aniqlanadi:

Obyekt – bu kompyuter xotirasida fizik jihatdan joylashtirilgan va ularga kirish imkoniyatiga ega bo‘lgan ma'lumotlar to‘plami (obyektlar maydonlari). Ism obyektni tashkil etuvchi maydon va metodlarga kirish uchun ishlatiladi. Ba’zi holatlarda, obyekt xossalar yoki metodlarni o‘z ichiga olmaydi va nomga ega bo‘lmasligi mumkin.

[1] Robson, D. August 1981. Object-oriented Software Systems, Byte vol.6(8), p.74.

Har qanday obyekt ma'lum bir sinfga tegishli. Sinf ma'lumotlarning tavsifi va ulardagi amallarni o‘z ichiga oladi. Sinf ma'lum bir -biriga bog‘liq, aslida mavjud obyektlarning umumiy tavsifini beradi. Obyekt - bu sinfning aniq nusxasi.

Misol. Ikki o‘lchovli fazoda geometrik shakl doirasini ifodalovchi oddiy **Circle** sinfini ko‘rib chiqaylik. Bu sinfning atributlarini quyidagicha ta'riflash mumkin:

x - koordinata, doira markazini belgilash uchun OX o‘qi bo‘yicha

y - koordinata, doira markazini belgilash uchun OY o‘qi bo‘yicha

R - aylananing radiusini ko‘rsatish uchun

Uning ba'zi amallarini quyidagicha ta'riflash mumkin:

findArea() – yuzani hisoblash metodi

findCircumference() – aylanani uzunligini hisoblash metodi

RadiusInc() – radiusni oshirish yoki kamaytirish metodi

O‘zlashtirish (ta’minlash) paytida qiymatlar atributlarning kamida bir qismiga beriladi. Agar biz my_circle obyektini yaratadigan bo‘lsak, uning holatini ko‘rsatish uchun x-koordinatalar: 2, y koordinatalari: 3 va R: 4 kabi qiymatlarni belgilashimiz mumkin. Endi, agar RadiusInc() metodiga ikki qiymati bilan murojaat qilsak, R o‘zgaruvchining qiymati 8 ga aylanadi. Bu operatsiya my_circle holatini o‘zgartiradi, ya'ni obyekt ma'lum xatti-harakatlarni bajaradi.

3. Obyektga yo‘naltirilgan dasturlash prinsiplari

Obyektga yo‘natirilgan dasturlash quyidagi prinsiplarga asoslanadi:

- Abstraksiya;
- Inkapsulyatsiya;
- Merosxo‘rlik;
- Polimorfizm;

Inkapsulyatsiya. Har bir obyekt shunday tuzilganki, unga kerakli ma'lumotlar dasturning tashqarisida emas, balki shu obyekt ichida yashaydi. Masalan, agar bizda "Foydalanuvchi" obyekti bo‘lsa, unda foydalanuvchi haqidagi barcha ma'lumotlar bo‘ladi: ism, manzil va boshqalar. Shuningdek, u "manzilni tekshirish" yoki "Pochta ro‘yxatiga obuna bo‘lish" metodlariga ega bo‘ladi.

Inkapsulyatsiya - bu tizimda ishlaydigan ma'lumotlar va metodlarni sinfda birlashtirishga va foydalanuvchidan amalga oshirish tafsilotlarini yashirishga imkon beruvchi tizimning xususiyati.

Inkapsulyatsiya - bu ma'lumotlarni manipulyatsiya qiladigan va kodni birlashtirgan, shuningdek, birinchi navbatda ma'lumotlarga to‘g‘ridan -to‘g‘ri tashqi kirishdan va noto‘g‘ri ishlatishdan himoya qiluvchi tamoyil. Boshqacha qilib aytganda, sinf ma'lumotlariga kirish faqat bir xil sinf metodlari yordamida amalga oshirilishi mumkin.

Inkapsulyatsiya sinf interfeysi tushunchasi bilan uzviy bog‘liq. Aslida, interfeysga kirmagan hamma narsa sinfga kiritilgan.

Inkapsulyatsiya va ma'lumotlarni yashirish. *Inkapsulyatsiya* - bu sinfda atributlar va metodlarni bir - biriga bog‘lash jarayoni. Inkapsulyatsiya orqali, sinfning ichki tafsilotlarini tashqaridan yashirish mumkin. Bu sinf a'zolariga tashqi tomondan faqat sinf tomonidan taqdim etilgan interfeys orqali kirishga imkon beradi.

Ma'lumotlarni yashirish. Qoida tariqasida, sinf shunday tuzilganki, uning ma'lumotlariga (atributlariga) faqat uning sinf metodlari yordamida kirsa bo‘ladi va tashqi tomondan to‘g‘ridan-to‘g‘ri kirishdan ajratiladi. Obyekt ma'lumotlarini ajratish jarayoni *ma'lumotni yashirish* deb ataladi.

Misol. Yuqoridagi Circle sinfida siz atributlarni sinfdan tashqarida ko‘rinmas holga keltirish va sinf ma'lumotlariga kirish uchun sinfga yana ikkita metod qo‘shish orqali ma'lumotlarni yashirishingiz mumkin:

setValues(), x va y-koordinatalarga qiymat tayinlash

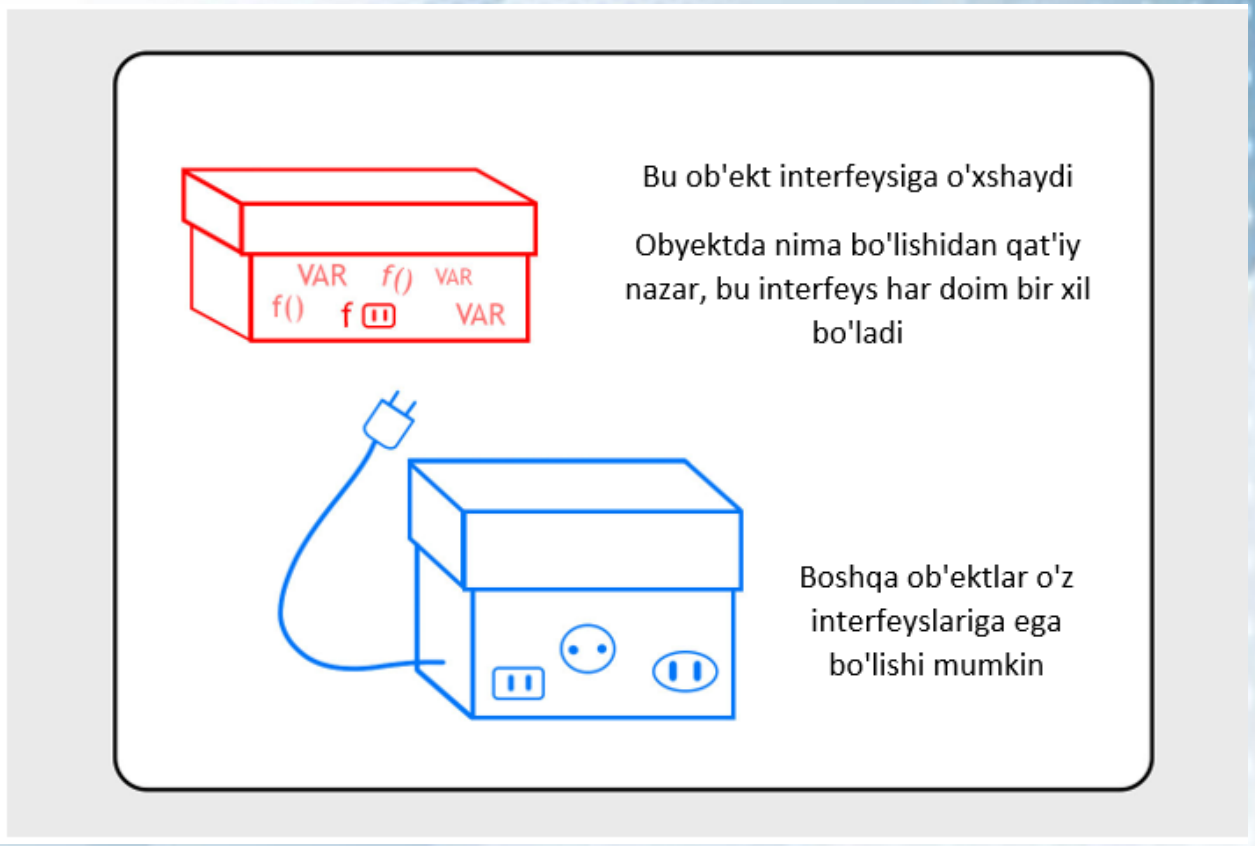
getValues (), x va y koordinatasini olish qiymatlarni olish metodi.

Bu yerda **my_circle** obyektining shaxsiy ma'lumotlariga to‘g‘ridan -to‘g‘ri Circle sinfiga kiritilmagan har qanday metod bilan kirish mumkin emas. Buning o‘rniga, **setValues()** va **getValues()** metodlari orqali kirish kerak.

Abstraksiya. Obyekt biz obyektga tashqaridan kira oladigan metod va xususiyatlarga ega. Xuddi qurilmadagi biror tugmani bosishimiz mumkin bo‘lganidek. Qurilmada juda ko‘p narsalar bor, bu uning ishlashini ta'minlaydi, lekin asosiy panelda faqat tugma bor. Bu tugma mavhum interfeysdir.

Tizimda biz "foydalanuvchini o‘chirish" deyishimiz mumkin. OYD tilida bu "user.delete()" bo‘ladi - ya'ni biz "user" obyektiga murojaat qilamiz va "delete" metodini chaqiramiz. Qiziq tomoni shundaki, o‘chirish qanday sodir bo‘lishi biz uchun unchalik muhim emas: OYD murojaat paytida bu haqda o‘ylamaslikka imkon beradi.

Masalan, do‘konda ikkita dasturchi ishlaydi: biri buyurtma modulini, ikkinchisi yetkazib berish modulini yozadi. "Buyurtma" obyektida birinchi dasturchi "bekor qilish" metodiga ega, ikkinchi dasturchi esa yetkazib berish tufayli buyurtmani bekor qilishi kerak. Ikkinchi dasturni osongina "order.cancel()" metodi orqali buyurtmani bekor qilish mumkin. Birinchi dasturchiga bekor qilishni qanday amalga oshirishi unga qiziq emas: u qanday xatlar yuboradi, ma'lumotlar bazasiga nima yozadi, qanday ogohlantirishlarni ko‘rsatadi, bu “cancel()” metodining ishidir.



1-rasm.

Merosxo‘rlik. Merosxo‘rlik – nusxa ko‘chirish qobiliyati. OYD boshqa obyektning tasviri va o‘xshashligida ko‘plab obyektlarni yaratishga imkon beradi. Bu sizga kodni ikki yuz

marta nusxalash va joylashtirishga emas, balki odatdagidek bir marta yozib, keyin ko‘p marta ishlatishga imkon beradi.

Meros - bu bitta obyekt boshqasining xususiyatlarini olish jarayonidir. Aniqroq aytganda, obyekt boshqa obyektning asosiy xususiyatlarini meros qilib olishi va unga o‘ziga xos xususiyat va metodlarni qo‘shishi mumkin.

Meros - bu mavjud sinflardan yangi sinflar yaratish, uning imkoniyatlarini kengaytirish va takomillashtirish imkonini beradigan mexanizm. Mavjud sinflar asosiy sinflar (ajdod, supersinflar), yangi sinflar esa bola (avlod) sinflari deb nomlanadi.

Masalan, sizda "Foydalanuvchi" ideal obyekt bo‘lishi mumkin: unda siz foydalanuvchi bilan sodir bo‘lishi mumkin bo‘lgan hamma narsani yozasiz. Sizda xossa bo‘lishi mumkin: ism, yosh, manzil, karta raqami. Va "chegirma berish", "buyurtmani tekshirish", "buyurtmalarni topish", "qo‘ng‘iroq qilish" usullari bo‘lishi mumkin.

Meros – bu tizimning xususiyatidir, bu sizga mavjud sinfga asoslangan yoki qisman yoki to‘liq olingan funksiyalarni tavsiflashga imkon beradi. Meros qoldiriladigan sinfga asosiy yoki ajdod deyiladi. Yangi sinf - avlod, merosxo‘r yoki hosil qilingan sinf deb ataladi.

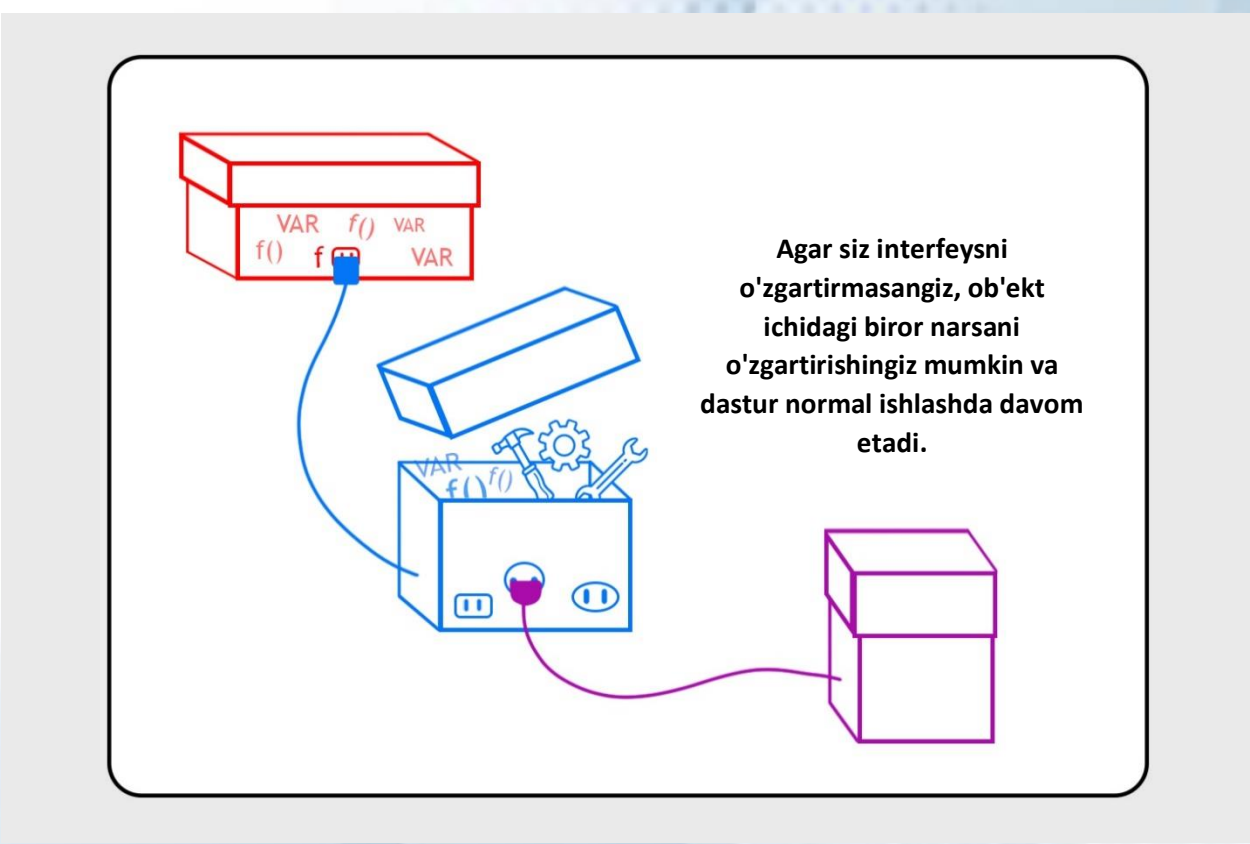
Shuni ta'kidlash kerakki, hosil qilingan sinf ajdodning spetsifikatsiyasiga to‘liq mos keladi, lekin qo‘shimcha funksiyalarga ega bo‘lishi mumkin. Interfeyslar nuqtai nazaridan, har bir olingan sinf ajdod inf interfeysini to‘liq amalga oshiradi. Buning aksi to‘g‘ri emas.

Polimorfizm. Polimorfizm – yunoncha so‘z bo‘lib, u turli shakllarga ega bo‘lish qobiliyatini bildiradi. Obyektga yo‘naltirilgan paradigmada polimorfizm amallarni ular bajaradigan holatiga qarab har xil usulda qo‘llashni nazarda tutadi. Polimorfizm har xil ichki tuzilishga ega bo‘lgan obyektlarga umumiy tashqi interfeysga ega bo‘lishga imkon

beradi. Polimorfizm, ayniqsa, merosni amalga oshirishda samaralidir.

Polimorfizm - umumiy muloqot tili. OYDda barcha obyektlar bir-biri bilan ular tushunadigan tilda muloqot qilishi muhim. Agar har xil obyektlarda "delete" metodi bo‘lsa, u aynan shunday ishni bajarishi va hamma joyda xuddi shunday yozilishi kerak. Bir obyektda "delete", ikkinchisida "clear" bo‘lishi mumkin emas.

Shu bilan birga, obyekt ichida metodlar turli yo‘llar bilan amalga oshirilishi mumkin. Masalan, biror narsani o‘chirish - bu ogohlantirish, keyin ma'lumotlar bazasidagi elementni o‘chirilgan deb belgilash va foydalanuvchini o‘chirish uning xaridlarini bekor qilishni, pochta ro‘yxatidan obunani bekor qilishni va sotib olish tarixini arxivlashni bildiradi. Voqealar boshqacha, lekin dasturchi uchun bu muhim emas. U faqat delete() metodiga ega va unga ishonadi.



2-rasm

Polimorfizm – bu ikki yoki undan ortiq o‘xshash, lekin biroz boshqacha muammolarni yechishda bir xil metod nomidan foydalanadigan mexanizm.

Polimorfizmning maqsadi sinf uchun umumiy harakatlarni aniqlash maqsadida bitta nomdan foydalanish. Umuman olganda, polimorfizm tushunchasi "bitta interfeys, ko‘p usullar" g‘oyasidir.

Misol. Keling, har biri **findArea()** metodi bilan ikkita sinfni - "Circle" va "Square" sinflarini ko‘rib chiqaylik. Sinflardagi metodlarning nomi va maqsadi bir xil bo‘lsa-da, ichki amalga oshirish, ya'ni maydonni hisoblash tartibi har bir sinf uchun turlichadir. Circle sinfining obyekti **findArea()** metodini chaqirganda, amal Square sinfining **findArea()** metodi bilan ziddiyatsiz doira yuzasini topadi.

Modullik. Modullik - bu tizimning ichki ulangan, lekin erkin bog‘langan modullarga bo‘linadigan xususiyati.

Modullik - bu tizimning bir-biri bilan chambarchas bog‘liq bo‘lgan qismlarga (modullarga) bo‘linish qobiliyati bilan bog‘liq xususiyati. Modullik boshqa obyektlarga va umuman

tizimga ta'sir qilmasdan yangilanishi yoki almashtirilishi mumkin bo‘lgan obyektlarni diskret dasturlashga asoslangan.

Smalltalk kabi ba'zi dasturlash tillarida modul yo‘q va sinflar bo’linishning yagona fizik asosidir. Boshqa tillarda, shu jumladan Object Pascal, C++, Javada modul mustaqil til tuzilishi hisoblanadi. Bu tillarda sinflar va obyektlar tizimning mantiqiy tuzilishini tashkil qiladi; ular tizimning fizik tuzilishini tashkil etuvchi modullarga joylashtirilgan. Bu xususiyat, agar tizim yuzlab sinflardan iborat bo‘lsa, ayniqsa foydali bo‘ladi.

Shunday qilib, modullik va inkapsulyatsiya bir-biri bilan chambarchas bog‘liq. Modullik turli dasturlash tillarida har xil yo‘llar bilan qo‘llab-quvvatlanadi. Masalan, C++ da modullar alohida kompilyatsiya qilingan fayllardir. C/C++ uchun modullarning oldingi qismini .h kengaytmasi bo‘lgan alohida fayllarga joylashtirish odatiy holdir (sarlavha fayllari deb ataladi). Amalga oshirish, ya'ni modul matni .c

kengaytmasi bo‘lgan fayllarda saqlanadi (C++ dasturlarida ko‘pincha cp va .cpp kengaytmalari ishlatiladi). Fayllar orasidagi bog‘lanish #include makroprotsessor ko‘rsatmasi bilan e'lon qilinadi. Bu yondashuv faqat konvensiyaga asoslangan va tilning o‘ziga xos qat'iy talabi emas. Object Pascal tilida modullik prinsipi biroz qat'iyroq rasmiylashtirilgan. Bu til birlikning interfeysi va bajarilishining o‘ziga xos sintaksisini belgilaydi. Javada paket deb ataladigan tushuncha mavjud. Har bir to‘plamda ba'zi mantiqiy atributlar bo‘yicha guruhlangan bir nechta sinflar mavjud.

Modullik, kerakli tavsifni topishni osonlashtirishdan tashqari, loyihani qurish jarayonini sezilarli darajada tezlashtirishga imkon beradi (albatta, alohida kompilyatsiyani qo‘llab-quvvatlaydigan kompilyatorlar uchun).

Tabiiyki, bularning barchasi interfeyslarning barqarorligiga juda qattiq cheklovlar qo‘yadi, lekin barqaror interfeyslarni shakllantirish vazifasi umuman dizayn vazifasidir.

Iyerarxiya. Abstraksiya - foydali narsa, lekin har doim, eng oddiy vaziyatlardan tashqari, tizimdagi mavhumliklar soni bizning aqliy imkoniyatlarimizdan ancha oshib ketadi. Inkapsulyatsiya abstraksiyalarning ichki mazmunini ko‘rish maydonidan olib tashlash orqali ma'lum darajada bu to‘siqni olib tashlashga imkon beradi. Modullik, shuningdek, mantiqiy bog‘liq abstraksiyalarni guruhlarga ajratish orqali vazifani soddalashtiradi. Lekin bu yetarli emas.

Iyerarxiyalarning mavjudligi - bu tizim obyektlarining ba'zi qoidalariga muvofiq tartiblash.

Abstraksiyalardan iyerarxik tuzilmaning shakllanishi tufayli murakkab muammolarni tushunishda sezilarli soddalashtirishga erishiladi. Iyerarxiyani quyidagicha ta'riflaylik:

Iyerarxiya - bu mavhumliklarning tartiblanishi, ularning darajadagi joylashuvi.

Murakkab tizimlarga nisbatan iyerarxik tuzilmalarning asosiy turlari sinf tuzilishi (*"is-a" ierarxiyasi*) va obyekt strukturasi (“*part-of” iyerarxiyasi*) hisoblanadi.

1) "is-a" ierarxiyasi. Obyektga yo‘naltirilgan tizimlarning muhim elementi va "is-a" ierarxiyasining asosiy turi-yuqorida aytib o‘tilgan meros tushunchasi. Meros - bu sinflar o‘rtasidagi munosabatni (ajdod / avlod munosabatlari), bir sinf bir yoki bir nechta boshqa sinflarning strukturaviy yoki funksional qismini oladi (mos ravishda bitta va ko‘p meros). Boshqacha qilib aytganda, meros mavhumliklar iyerarxiyasini yaratadi, bunda kichik sinflar bir yoki bir nechta yuqori

sinflardan tuzilmani meros qilib oladi. Koʻpincha kichik sinf ajdod komponentlarini yaratadi yoki qayta yozadi.

Semantik jihatdan, meros "is-a" munosabatini tavsiflaydi. Masalan, ayiq - sutemizuvchi, uy-koʻchmas mulk, "Quick sort " - saralash algoritmi. Shunday qilib, meros umumlashtirish-ixtisoslashuv iyerarxiyasini vujudga keltiradi.

2) “part of” iyerarxiyasi. Agar "is a" iyerarxiyasi umumlashtirish / ixtisoslashuv munosabatlarini aniqlasa, u holda "part of" munosabatlar yigʻindisi iyerarxiyasini kiritadi.

Tiplashtirish. Tiplashtirish – bu barcha obyektlar turlarining tavsifi;

Tur tushunchasi mavhum ma'lumotlar turlari nazariyasidan olingan. Bizning maqsadlarimiz uchun atamalar turi va sinfi bir-birining oʻrnini bosadi deb taxmin qilish kifoya. (Aslida, tur va sinf bir xil emas; ba'zi tillarda ular farqlanadi. Masalan, Trellis / Owl tilining dastlabki versiyalari obyektga ham sinf, ham turga ega boʻlishga ruxsat bergan.

Hatto Smalltalkda ham SmallInteger, LargeNegativeInteger, LargePositiveInteger sinflari bir xil turdagi Integerga tegishli bo‘lsa-da, har xil sinflarga tegishli).

Tiplashtirish – bu boshqa sinf o‘rniga bir sinf obyektlarini ishlatishdan himoya qilish usuli (kuchli tiplashtirish) yoki hech bo‘lmaganda bunday foydalanishni nazorat qilishdir (zaif tiplashtirish).

Tiplashtirish bizni mavhumliklarimizni shunday ifoda etishga majbur qiladi, uni amalga oshirishda ishlatiladigan dasturlash tili dizayn qarorlariga sodiq qoladi. Yozish konsepsiyasida turni moslashtirish g‘oyasi markaziy o‘rinni egallaydi. Masalan, fizik birliklarni olaylik. Vaqtni masofaga bo‘lish orqali biz og‘irlikni emas, tezlikni olamiz. Haroratni kuchga ko‘paytirishning ma'nosi yo‘q, lekin masofani kuchga ko‘paytirishda – ma’no mavjud. Bularning barchasi kuchli tiplashtirish misollaridir, bu yerda dastur sohasi abstraksiyalarni ishlatish va kombinatsiyasiga qoidalar va

cheklovlar qo‘yadi. Zaif tiplashtirish bilan ishlar biroz murakkablashadi. Zaif tiplashtirish polimorfizm tushunchasi bilan chambarchas bog‘liq.

Parallelizm. Parallelizm - bu obyektlarning faol yoki passiv holatda bo‘lish xususiyati. Ko‘p protsessorli arxitektura uchun obyekt alohida boshqaruv kanali bo‘lishi mumkin (jarayon abstraksiyasi), bu parallellik masalalarini hal qilishni soddalashtiradi (tupik, blokirovka va boshqalar). Bir protsessorsiz arxitektura uchun u minimal shaklda amalga oshiriladi. Misol sifatida Windows ko‘p oynali interfeysi.

Avtomatik tizimlar bir vaqtning o‘zida ko‘plab hodisalarni qayta ishlashi kerak bo‘lgan vazifalar mavjud. Boshqa hollarda, qayta ishlash quvvatiga bo‘lgan ehtiyoj bitta protsessor resurslaridan oshadi. Bu holatlarning har birida muammoni hal qilish uchun bir nechta kompyuterlardan foydalanish yoki ko‘p protsessorli kompyuterda ko‘p vazifalarni bajarish tabiiy jarayondir.

Jarayon (boshqaruv oqimi) - bu tizimdagi asosiy harakat birligi. Har bir dasturda kamida bitta nazorat oqimi bor, parallel tizimda bunday oqimlar koʻp: ba'zilari qisqa muddatli, boshqalari esa butun tizim davomida yashaydi. Haqiqiy parallelizmga faqat koʻp protsessorli tizimlarda erishiladi va bitta protsessorli tizimlar vaqtni taqsimlash algoritmlari orqali parallellikni simulyatsiya qiladi.

Bu "apparat" farqidan tashqari, biz resurs talablari nuqtai nazaridan "ogʻir" va "yengil" parallellikni ajratamiz. "Ogʻir" jarayonlar boshqalardan mustaqil ravishda operatsion tizim tomonidan boshqariladi va ular uchun alohida himoyalangan manzillar maydoni ajratiladi.

Koʻpgina zamonaviy operatsion tizimlar parallellikni toʻgʻridan-toʻgʻri qoʻllab-quvvatlaydi va bu holat obyektga yoʻnaltirilgan tizimlarda parallellikni ta'minlash qobiliyati uchun juda foydali. Masalan, Zamonaviy Windows tizimlari

ko‘p funksiyali; ular shuningdek jarayonlarni yaratish va boshqarish uchun dasturiy interfeyslarni ham ta'minlaydilar.

Butunlilik – obyektlarning o‘z holatini saqlab qolish va ma'lum sinfga mansubligi.