|  |
| --- |
| Programming |
| Fаrrux Аbduxаlilоv |

**4-qism: Programming**

**Guruh IDsi:**

**BTEC Learner Assessment Submission and Declaration**

Har bir o’quvchi vazifani baholash uchun taqdim etganda, yozilgan deklaratsiya o’ziniki ekanligini tasdiqlovchi imzo qoldirishi kerak.

|  |  |
| --- | --- |
| **O’quvchi (Talaba) IDsi:** |  |
| **Baholovchining ismi:** | Eshquvvatov Ravshanbek |
| **BTEC Dasturining mavzusi:** | Pearson BTEC Programming |
| **Qism yoki Komponent raqami va Mavzu:** | 4-qism: Programming |
| **Vazifani topshirish sanasi:** |  |

Iltimos, har bir topshiriq uchun taqdim etilgan dalillarni sanab o'ting. Dalillarni topish mumkin bo'lgan sahifa raqamlarini ko'rsating yoki dalillarning mohiyatini tavsiflang (masalan, video, rasm).

|  |
| --- |
| **O’quvchi deklaratsiyasi**  Ushbu vazifa uchun topshirilgan ish meniki ekanligini tasdiqlayman. Ishda foydalanilgan manbalarga aniqlik kiritdim va ko’rsatib o’tdim. Noto’g’ri deklaratsiya noto’g’ri ishlashning bir shakli ekanligini tushunaman.  **Talabaning imzosi:**    **Sana:** |

Mundarija

[1.0. Kirish 3](#_Toc129600062)

[2. Topshiriq 1 4](#_Toc129600063)

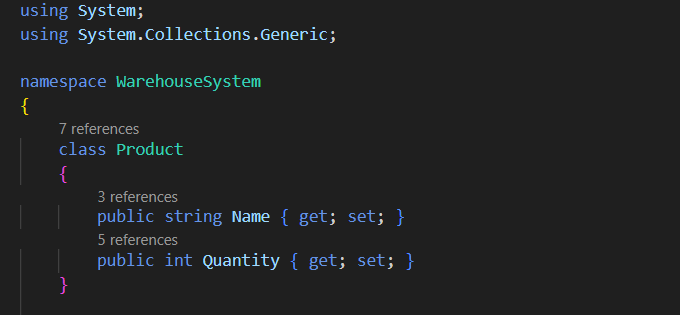
[3. Topshiriq 2 4](#_Toc129600064)

[4. Xulosa 5](#_Toc129600067)

[Manbalar ro’yxati 7](#_Toc129600068)

1.0. Kirish

**O’zingizni tanishtiring. Hisobot davomida nima qilishingizni yozing va tushuntiring.**



**Mahsulot klassi:**

Bu sinf ombordagi mahsulotni ifodalaydi.

U ikkita xususiyatga ega: mahsulot nomini saqlash uchun nom (string) va mahsulot miqdorini saqlash uchun Quantity (int).

**Ishlatilgan kutubxonalar:**

**Tizim:** Ko'p ishlatiladigan qiymat va mos yozuvlar ma'lumotlari turlarini, hodisalar va hodisalarni qayta ishlash vositalarini, interfeyslarni, atributlarni va ishlov berish istisnolarini belgilaydigan asosiy turlar va asosiy sinflarni taqdim etadi.

**System.Collections.Generic:** Ro'yxatlar, lug'atlar, navbatlar va steklar kabi umumiy to'plam sinflarini taqdim etadi.

**Amaldagi ma'lumotlar turlari:**

**string:** Belgilar ketma-ketligini ifodalaydi.

int: 32 bitli imzolangan butun sonni ifodalaydi.

**Sinf ta'rifi:**

Mahsulot: Ombordagi mahsulotni ifodalash uchun sinfni belgilaydi.

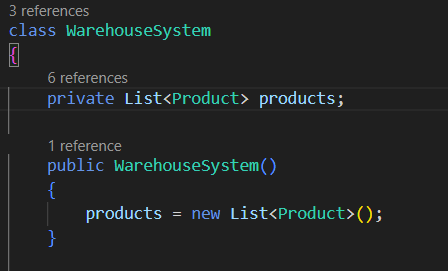
**Xususiyatlari:**

Ism: mahsulot nomini saqlash uchun string tipidagi xususiyat.

Miqdor: mahsulot miqdorini saqlash uchun int turidagi xususiyat.

csharp

Kodni nusxalash



**Ombor tizimi klassi:**

Bu sinf ombor operatsiyalarini boshqaradi.

Unda mahsulotlar ro'yxatini saqlash uchun List<Mahsulot> turidagi shaxsiy maydon mahsulotlari mavjud.

Konstruktor mahsulotlar ro'yxatini ishga tushiradi.

Amaldagi ma'lumotlar turlari:

**List<T>:** Belgilangan turdagi ob'ektlarning qattiq terilgan ro'yxatini ifodalaydi.

Mahsulot: Ro'yxatda saqlangan ob'ektlar turi.

Sinf ta'rifi:

**WarehouseSystem:** Ombor operatsiyalarini boshqarish uchun sinfni belgilaydi.

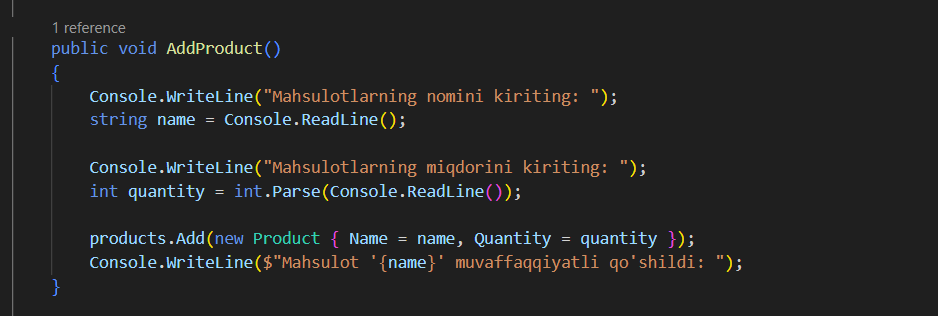
Maydon:

**private List<Mahsulot> mahsulotlar;:** Mahsulotlar roʻyxatini saqlash uchun shaxsiy maydonni eʼlon qiladi.

Konstruktor:

**public WarehouseSystem():** WarehouseSystem sinfining yangi nusxasini ishga tushiradi.

**mahsulotlar = new List<Mahsulot>();:** Mahsulotlar roʻyxatini ishga tushiradi.



**AddProduct usuli:**

Foydalanuvchidan yangi mahsulot nomi va miqdorini kiritishni taklif qiladi.

Kiritilgan nom va miqdor bilan yangi Mahsulot ob'ektini yaratadi va uni mahsulotlar ro'yxatiga qo'shadi.

Mahsulot qo'shilgandan so'ng muvaffaqiyatli xabarni ko'rsatadi.

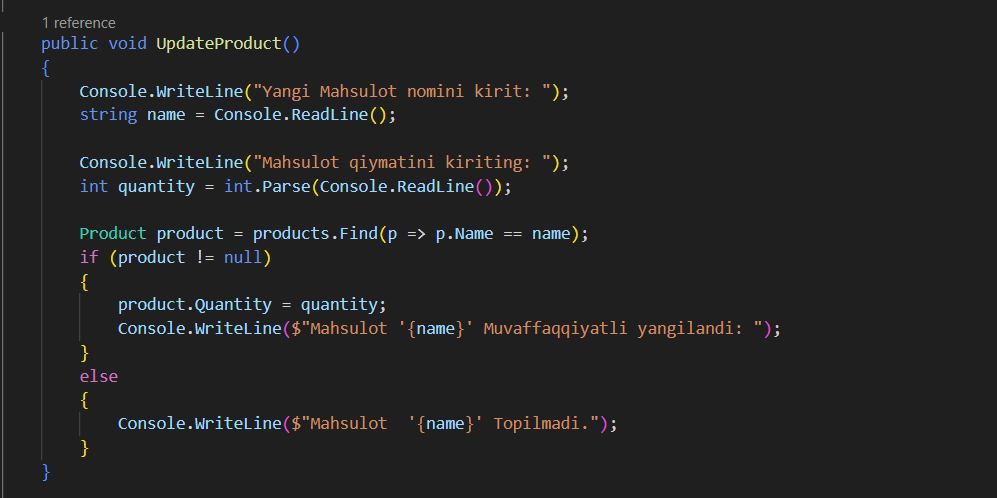
**Qo'llaniladigan usullar:**

**Console.WriteLine():** Belgilangan satr qiymatini standart chiqish oqimiga yozadi.

**Console.ReadLine():** standart kirish oqimidan keyingi belgilar qatorini o'qiydi.

**int.Parse():** Raqamning satr tasvirini uning 32-bitli imzolangan tamsayı ekvivalentiga aylantiradi.

**List<T>.Add():** Ob'ektni ro'yxat oxiriga qo'shadi.



**Mahsulotni yangilash usuli:**

Foydalanuvchidan mavjud mahsulot nomini va yangi miqdorini kiritishni taklif qiladi.

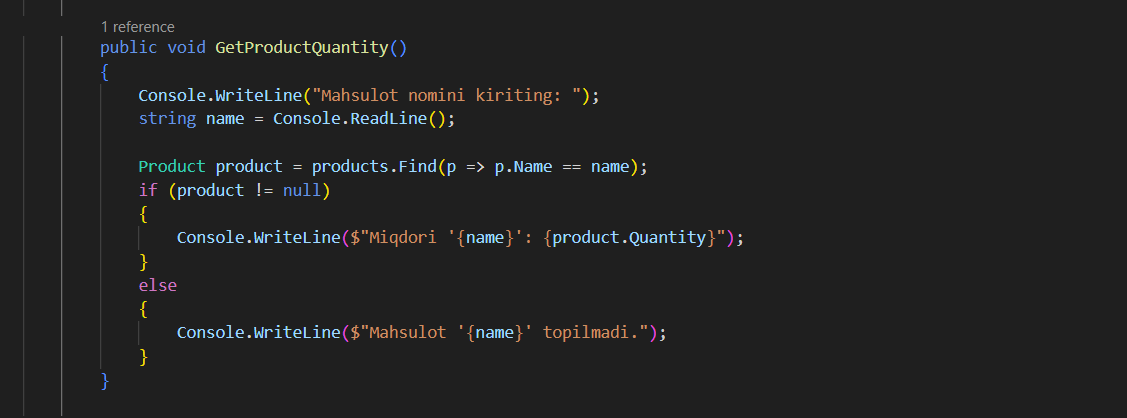
Mahsulotlar ro'yxatida kiritilgan nomi bilan mahsulotni topadi.

Agar mahsulot topilsa, uning miqdori yangi qiymat bilan yangilanadi.

Agar mahsulot topilmasa, mahsulot topilmaganligini bildiruvchi xabar paydo bo'ladi.

**Qo'llaniladigan usullar:**

***List<T>.Find():*** Belgilangan predikat bilan belgilangan shartlarga mos keladigan elementni qidiradi.



**GetProductQuantity usuli:**

Foydalanuvchiga mahsulot nomini kiritishni taklif qiladi.

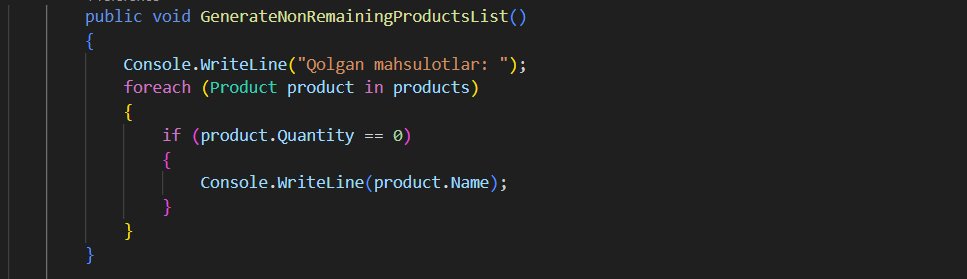
Mahsulotlar ro'yxatida kiritilgan nomi bilan mahsulotni topadi.

Agar mahsulot topilsa, uning miqdori ko'rsatiladi.

Agar mahsulot topilmasa, mahsulot topilmaganligini bildiruvchi xabar paydo bo'ladi.

Qo'llaniladigan usullar:

**List<T>.Find():** Belgilangan predikat bilan belgilangan shartlarga mos keladigan elementni qidiradi.



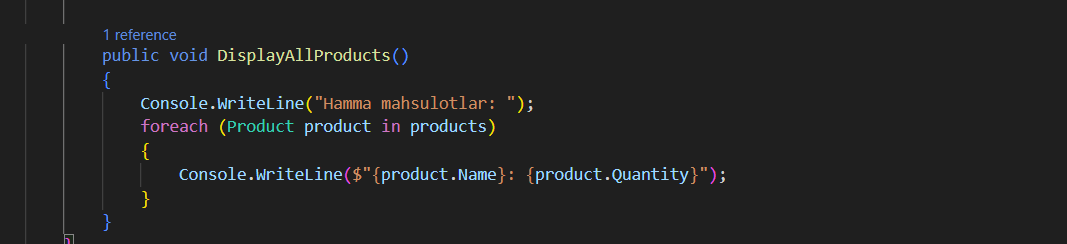
GenerateNonRemainingProductsList usuli:

Mahsulotlar ro'yxatidagi barcha mahsulotlarni takrorlaydi.

0 (qolgan mahsulotlar) miqdori bo'lgan mahsulotlarning nomlarini chop etadi.

**Takrorlash:**

***foreach tsikli:*** Mahsulotlar ro'yxatidagi har bir mahsulot ustidan takrorlanadi.

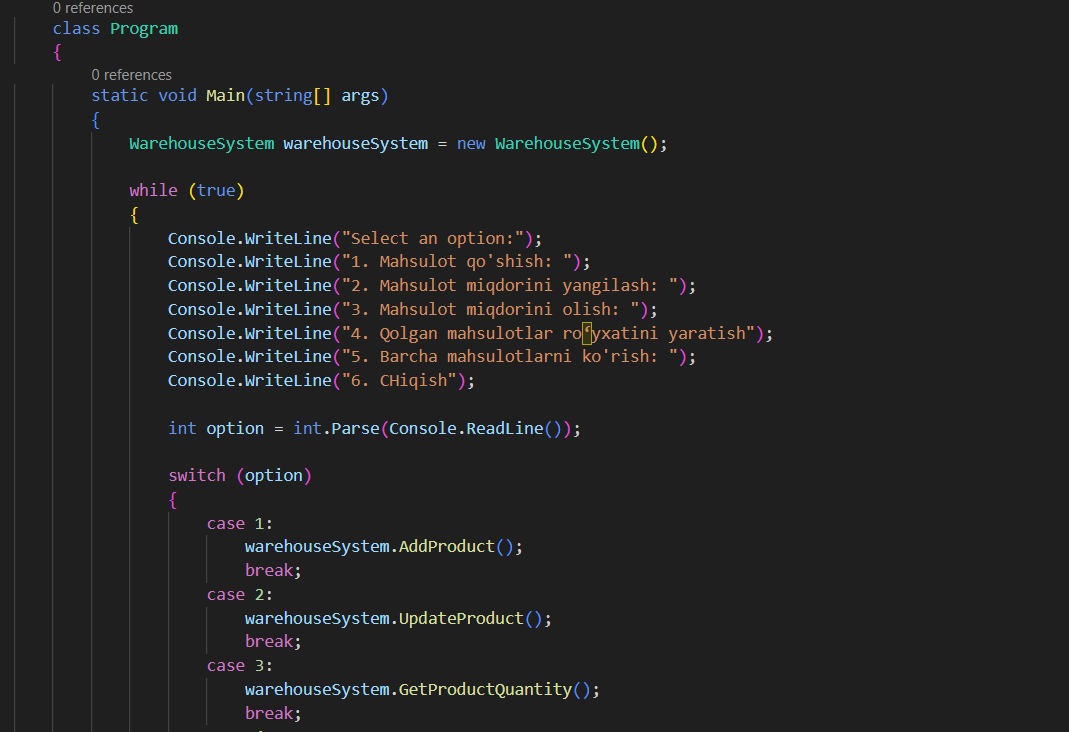


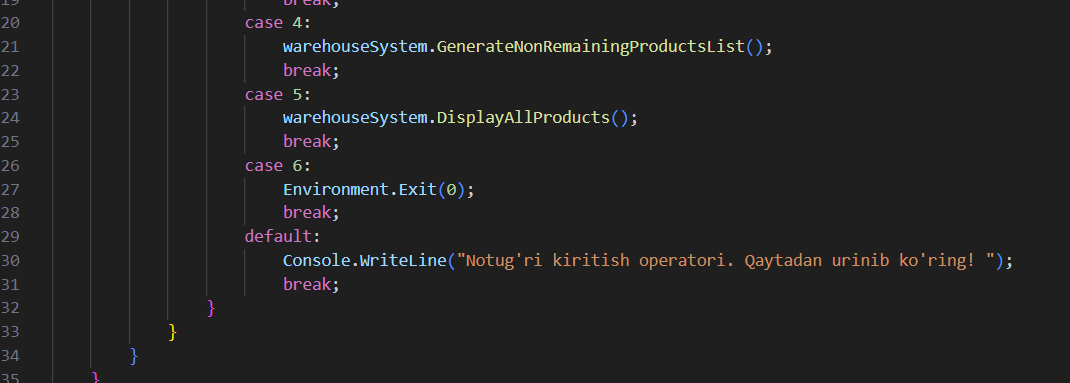
**DisplayAllProducts usuli:**

Barcha mahsulotlarni ularning miqdori bilan birga ko'rsatadi.

**Takrorlash:**

***foreach tsikli:*** Mahsulotlar ro'yxatidagi har bir mahsulot ustidan takrorlanadi.





**Dastur sinfi (Asosiy usul):**

Main usuli WarehouseSystem sinfining namunasini yaratadi.

U foydalanuvchiga variantlar menyusini ko'rsatish va foydalanuvchi kiritgan ma'lumotlar asosida tegishli operatsiyalarni bajarish uchun cheksiz tsiklni ishga tushiradi.

Variantlar:

1: Mahsulot qo'shing.

2: Mahsulot miqdorini yangilang.

3: Mahsulot miqdorini oling.

4: Qolgan mahsulotlar ro'yxatini yarating.

5: Barcha mahsulotlarni ko'rsatish.

6: Dasturdan chiqing.

**Qo'llaniladigan usullar:**

***Console.WriteLine():*** Belgilangan satr qiymatini standart chiqish oqimiga yozadi.

***Console.ReadLine():*** standart kirish oqimidan keyingi belgilar qatorini o'qiydi.

***int.Parse():*** Raqamning satr tasvirini uning 32-bitli imzolangan tamsayı ekvivalentiga aylantiradi.

Environment.Exit(): Belgilangan chiqish kodi bilan dasturdan chiqadi.

**Loop:**

***while loop:*** Foydalanuvchi dasturdan chiqishni tanlamaguncha menyu parametrlarini qayta-qayta bajaradi.

**Switch bayonoti:**

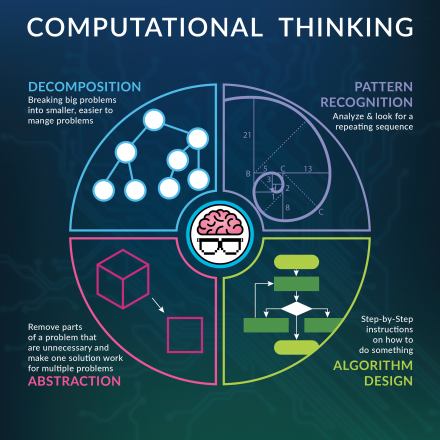
***switch bayonoti:*** Variant qiymatiga qarab bajariladigan ko'plab kod bloklaridan birini tanlaydi.

**Men Abduxalilov Farrux hozirda PDP universitetining 23-208-guruhdagi IT dasturiga o‘qishga kirganman. Mantiqiy fikrlashning ahamiyati haqida, xususan, dasturlash kontekstida men o'z yondashuvim va istiqbollarimni tahliliy va tanqidiy nuqtai nazardan tushuntiraman. Men mantiqiy fikrlash bilan bog'liq turli jihatlarni va o'zim ishlab chiqqan kutubxonani boshqarish dasturining ishlash tamoyillarini ko'rib chiqaman, har bir dastur segmenti kodida amalga oshirilishini ko'rsataman. Ushbu tekshiruv orqali biz ushbu kodlash amaliyotlarining aniqligi, tezligi va xatosizligini baholaymiz. Imtihon dasturga kiritilgan mantiqiy fikrlash jarayonlarini tahlil qilishni va har bir jarayonning turli bosqichlarida bajarilgan vazifalarni ko'rsatadigan misollarni o'z ichiga oladi. Ushbu tahlil davomida men har bir dastur segmentining sinchkovlik bilan bajarilishini tushuntirib beraman, bu operatsiyalarda aniqlik, samaradorlik va xatosiz ishlashning muhim ahamiyatini ta'kidlayman. Kodni ajratish mantiqiy fikrlash bilan bog'liq murakkab jarayonlarni yoritishga qaratilgan bo'lib, funktsional va samarali dasturni yaratish bilan bog'liq murakkabliklar haqida tushuncha beradi. Bundan tashqari, men ishlab chiqish bosqichida qilingan dizayn tanlovlari bo'yicha istiqbollarni taklif qilaman, bu qarorlar ortidagi sabablarga oydinlik kiritaman. Muayyan misollarni ta'kidlab, men har bir jarayonning turli bosqichlarida bajarilgan turli vazifalarni ko'rsatmoqchiman. Ushbu keng qamrovli tadqiqot nafaqat kodlash bo'yicha texnik mahoratni namoyish etadi, balki bardoshli va samarali dasturiy echimlarni yaratishda mantiqiy fikrlashning muhim rolini ham ta'kidlaydi.**

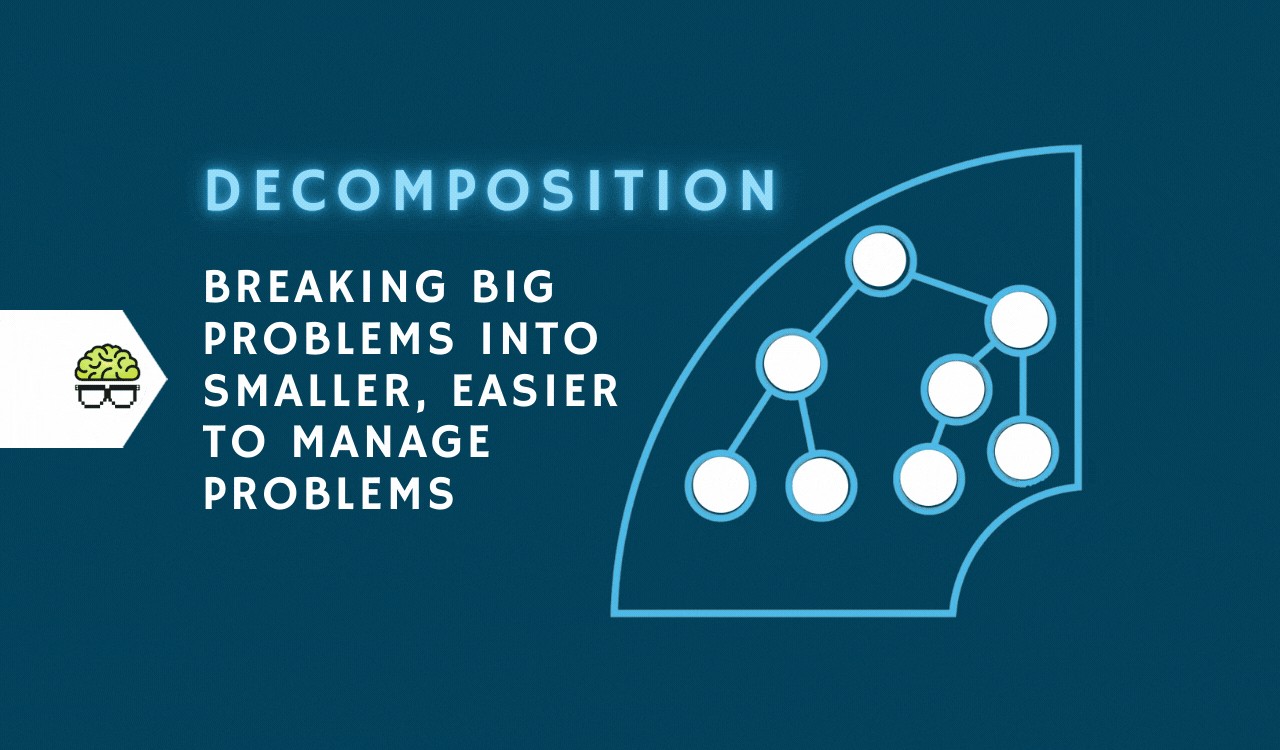
1. TOPSHIRIQ 1

A.P1. Hisoblash fikrlash ko'nikmalari qanday qilib dasturiy ilovalarga aylantirilishi mumkin bo'lgan yechimlarni topishda qo'llanilishi haqida tushuntirish.

Computational thinking bu murakkab muammolarni kichikroq, boshqariladigan qismlarga ajratish va ularni hal qilish uchun algoritmlarni ishlab chiqishni o'z ichiga olgan muammolarni hal qilish usuli. Dasturiy ta'minotni ishlab chiqish kontekstida Computational thinking qobiliyatlari samarali va samarali echimlarni yaratishda hal qiluvchi rol o'ynaydi. Dasturiy ta'minot dasturlarida talqin qilinishi mumkin bo'lgan echimlarni topishda Computational thinking qobiliyatlari qanday qo'llanilishini ko’rib chiqamiz.

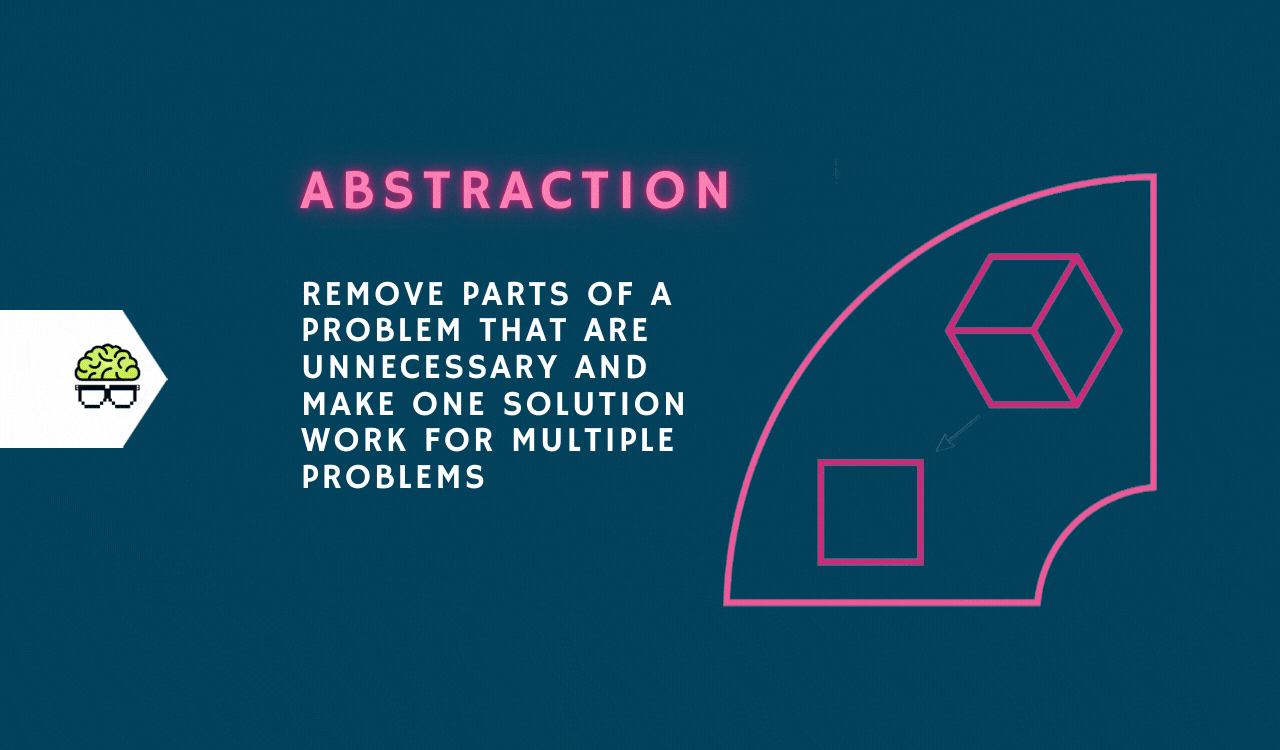


Decomposition: Computational thinking muammoni kichikroq, boshqariladigan kichik muammolarga ajratishni o'z ichiga oladi. Ushbu jarayon muammoni hal qilish uchun zarur bo'lgan asosiy komponentlar va vazifalarni aniqlashga yordam beradi. Dasturiy ta'minotni ishlab chiqishda dekompozitsiya dasturchilarga umumiy muammoni kichikroq modullar, funktsiyalar yoki sinflarga bo'lish imkonini beradi, bu esa dasturiy ta'minotni amalga oshirish va saqlashni osonlashtiradi.

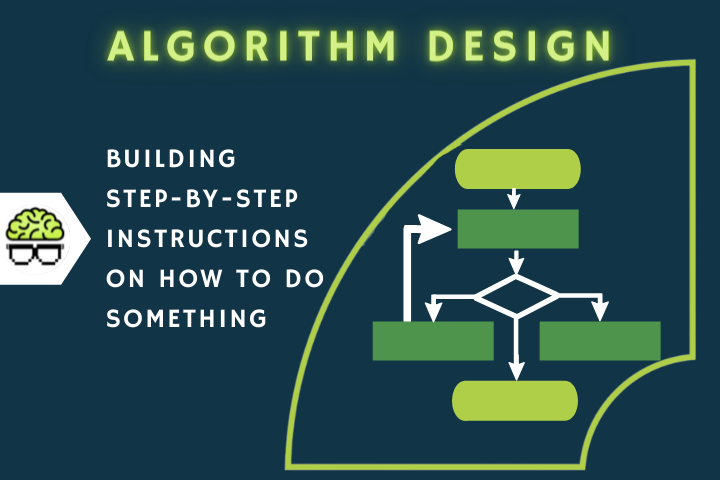


Pattern Recognition: Computational thinking muammo ichidagi naqshlarni yoki o'xshashliklarni tan olishni talab qiladi. Takroriy naqshlar yoki tuzilmalarni aniqlash orqali dasturchilar qayta foydalanish mumkin bo'lgan echimlar va algoritmlarni ishlab chiqishlari mumkin. Dasturiy ta'minotni ishlab chiqishda naqshlarni tanib olish dasturchilarga umumiy muammolarni samarali hal qilish uchun tasdiqlangan dizayn naqshlari, kutubxonalar va ramkalarni qo'llash imkonini beradi.

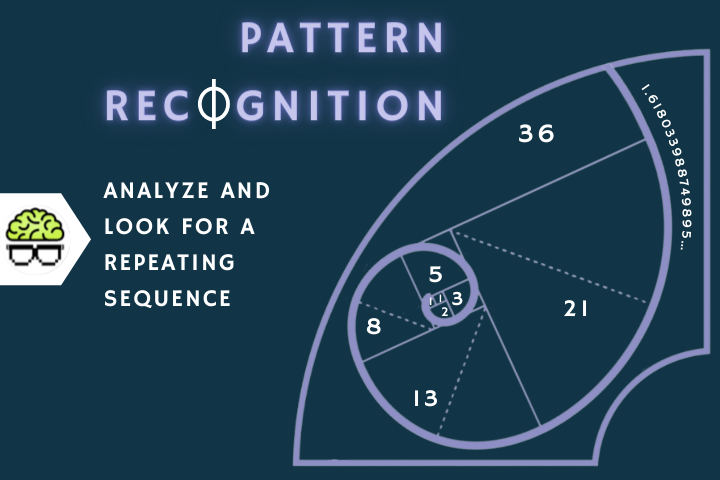
Abstraction: Abstraktsiya muammoning muhim tafsilotlariga e'tibor qaratish va ahamiyatsiz ma'lumotlarni e'tiborsiz qoldirishni o'z ichiga oladi Computational thinking muammoning muhim xususiyatlarini qamrab oluvchi mavhum modellar va tasvirlarni yaratishga yordam beradi. Dasturiy ta'minotni ishlab chiqishda mavhumlik dasturchilarga aniq belgilangan interfeyslar va APIlar orqasida murakkablikni yashirish orqali modulli, kengaytiriladigan va qo'llab-quvvatlanadigan kodni yaratishga imkon beradi.



Algorithm Design: Computational thinking muammolarni tizimli ravishda hal qilish uchun bosqichma-bosqich algoritmlarni yaratishga urg'u beradi. Dasturchilar muayyan vazifa yoki maqsadga erishish uchun zarur bo'lgan operatsiyalar ketma-ketligini aniqlash uchun algoritmlardan foydalanadilar. Dasturiy ta'minotni ishlab chiqishda algoritmni loyihalash ma'lumotlarni saralash, ma'lumotlarni qidirish yoki kirishlarni qayta ishlash kabi dasturiy ilovalarning mantiqiy va funksionalligini amalga oshirish uchun juda muhimdir.

s

Pattern Recognisen Computational thinkingning muhim tarkibiy qismi bo'lib, bizga ma'lumotlar ichidagi qonuniyatlar yoki o'xshashliklarni aniqlash va tahlil qilish imkonini beradi. Informatika, statistika va mashinani o'rganish kabi turli sohalarda naqshlarni aniqlash, tasniflash va samarali talqin qilish uchun naqshni aniqlash usullari qo'llaniladi. Statistik usullar, jumladan, regressiya tahlili va klasterlash algoritmlari matematik modellar va ehtimollar nazariyasiga asoslangan naqshlarni aniqlash uchun asos yaratadi. Neyron tarmoqlar va qarorlar daraxtlari kabi mashinani o'rganish algoritmlari bashorat qilish yoki tasniflash uchun ma'lumotlardan naqshlarni avtonom tarzda o'rganadi. Tasvirni qayta ishlash usullari vizual ma'lumotni tahlil qilish uchun naqshni aniqlashdan foydalanadi, signalni qayta ishlash usullari esa audio yoki biotibbiyot ma'lumotlari kabi signallardagi naqshlarni aniqlaydi. Bundan tashqari, tabiiy tilni qayta ishlash matn ma'lumotlaridagi naqshlarni izohlash va tushunish uchun naqshni aniqlashdan foydalanadi. Umuman olganda, naqshni aniqlash bizga mazmunli tushunchalarni olish, asosli qarorlar qabul qilish va turli sohalardagi murakkab muammolar yechimlarini ishlab chiqish imkonini berib, hisoblash fikrlashni kuchaytiradi.



Computational thinking ko'nikmalari dasturiy ta'minotni ishlab chiqish jarayoni uchun asos bo'lib, dasturchilarga muammolarni tahlil qilish, echimlarni loyihalash va algoritmlarni samarali amalga oshirish imkonini beradi. Dekompozitsiya, naqshni aniqlash, abstraksiya, algoritmni loyihalash va baholash kabi tamoyillarni qo'llash orqali dasturchilar murakkab muammolarni hal qiladigan va foydalanuvchilarga qiymat beradigan dasturiy ilovalar yaratishi mumkin.

A.p2. Turli dasturlash tillarida dasturlash tamoyillari qanday qo'llanilishi va dasturiy ilovalarni ishlab chiqarish haqida tushuntirish.

Kompyuter dasturlashning asosiy printsipalari, turli dasturlash tillarida dasturlarni yaratishda ilg'or qo'llaniladi. Bu printsipallar quyidagi ko'rinishlarda amalga oshiriladi:

Obyektga yo'naltirish: Dasturlash obyektorientlangan dasturlash tillarida qo'llaniladigan eng mashhur printsipalardan biridir. Bu printsip, dastur qurilmalarning obyektlar, sinflar va metodlar yordamida qurilishini ta'minlaydi. Misol uchun, Java va Python dasturlash tillarida obyektlar asosida dasturlar yaratiladi.

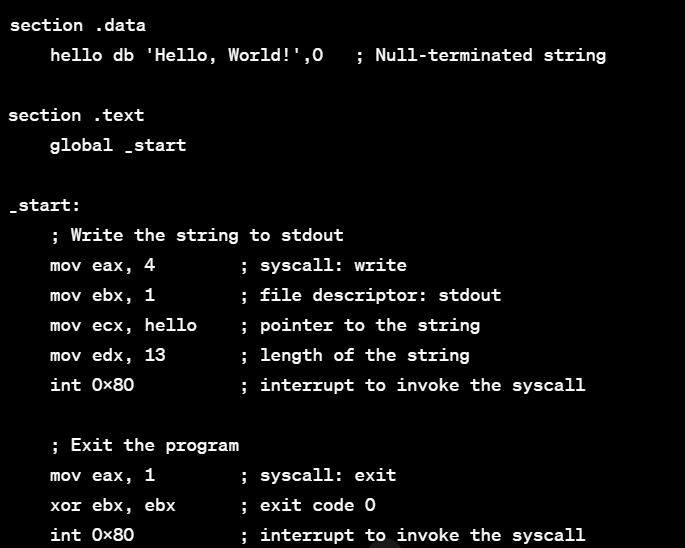
Funksional dasturlash: Funksional dasturlash tillarida amalga oshirilgan yondashuv asosida dasturlar yaratiladi. Bu til turining mohiyati, funksiyalar va lambda ifodalar orqali belgilanadi. Haskell va Scala funksional dasturlash tillariga misollar kiradi.

Protsedural dasturlash: Protsedural dasturlash, dasturlash tillarining eng klassik uslubi hisoblanadi. Bu uslubda, dastur o'sha tartibda bajariladi va ishni bajarish uchun bo'limlar, funksiyalar va operatsiyalar foydalaniladi. C va Pascal dasturlash tillarida protsedural dasturlashdan foydalaniladi.

Funksiyalarni ko'rib chiqish: Bu printsip, dastur kodini ko'rib chiqish va yagona funksiya yoki komponentlarga bo'lib ajratishni ta'minlaydi. Bu uslub, kodni o'zgaruvchilardan va bir qatordan bo'shating, shuningdek, kodni qulay qilib, sinflarga bo'lib ajratadi. JavaScript funksional dasturlashga odatlanadi.

Algoritmlar va ma'lumotlar tahlili: Dasturlash asosiy qismlaridan biri algoritmik yondashuv va ma'lumotlar tahlili. Dasturlash tillari, algoritmik vazifalarni bajarish, ma'lumotlar strukturasini o'rganish va ma'lumotlar ustidan operatsiyalar bajarish uchun ilg'or vositalar taqdim etadi. Python, Java, C++, va C# dasturlash tillari algoritmik yondashuv va ma'lumotlar tahlilini amalga oshirishda keng qo'llaniladi.

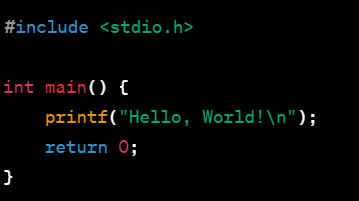
**Past darajadagi tillar (masalan, Assambly):**

****

**Assembly code**

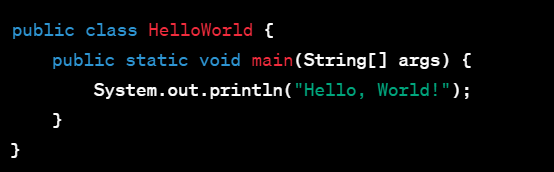
Past darajadagi dasturlash: Assembly kabi past darajadagi dasturlash tillarida mantiqiy fikrlash apparat resurslarini bevosita boshqarish uchun zarurdir. Ishlab chiquvchilar arxitekturani tushunish va apparat vositalarini samarali boshqarish uchun aniq ko'rsatmalar yaratishlari kerak.

**Protsessual tillar (masalan, C):**

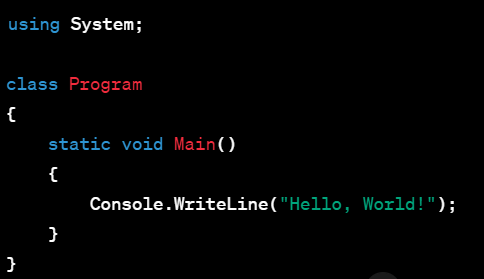
****

**C code**

Protsessual tillardagi protsessual mantiq: C kabi protsessual tillar vazifalarni protseduralar yoki funktsiyalarga taqsimlash orqali dasturlarda mantiqiy fikrlashni tashkil qilishni osonlashtiradi. Ishlab chiquvchilar loyihalarni samarali bajarish uchun algoritmlarni strategiklaydi, boshqaruv tuzilmalaridan foydalanadi va xotirani mantiqiy ravishda boshqaradi. **Ob'ektga yo'naltirilgan tillar (masalan, Java, C#):**

Ob'ektga yo'naltirilgan tillar diqqatni ob'ektlarga va ularning o'zaro ta'siriga yo'naltirish orqali mantiqiy fikrlashni ta'kidlaydi.****

**Java code**

****

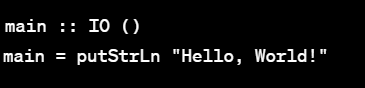
**C# code**

Ob'ektga yo'naltirilgan tillarda mantiqiy fikrlash sinflarni, ob'ektlarni va ularning o'zaro ta'sirini loyihalashni o'z ichiga oladi.

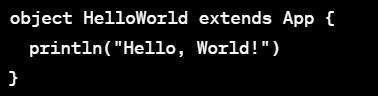
Ishlab chiquvchilar modulli va kengaytiriladigan kod uchun inkapsulyatsiya, meros va polimorfizm kabi tamoyillarni qo'llaydilar.

**Funktsional tillar (masalan, Haskell, Scala):**

Funktsional tillarda mantiqiy fikrlash:

****

**Haskel code**

****

**Scala code**

Funktsional tillar matematik funktsiyalar va o'zgarmas ma'lumotlarga urg'u beradi. Mantiqiy fikrlash hisoblashlarni funksiya sifatida ifodalashni, rekursiyani tushunishni va yuqori darajadagi funktsiyalardan foydalanishni o'z ichiga oladi.

**Skript tillari (masalan, Python, JavaScript):**

Skript tillarida mantiqiy fikrlash:

****

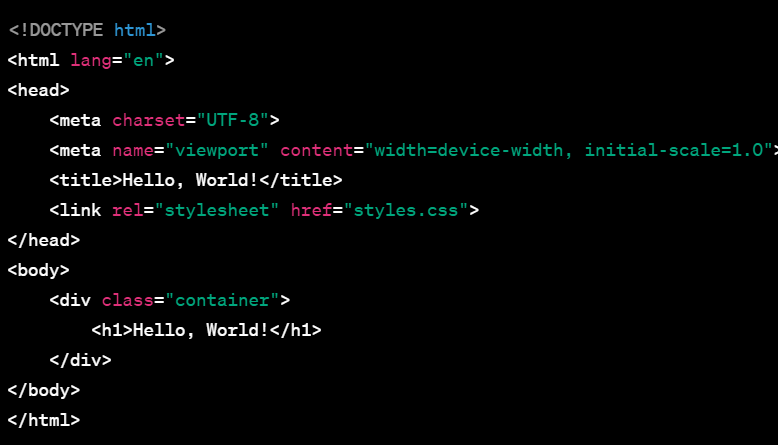
**Python code**

****

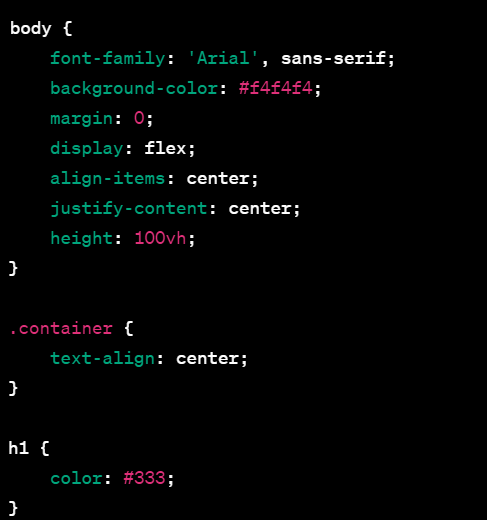
**Java Script code**

Skript tillari ko'pincha o'qilishi va tez rivojlanishini birinchi o'ringa qo'yadi. Mantiqiy fikrlash algoritmlarni loyihalash, ma'lumotlar t uzilmalari bilan ishlash va skriptlarni avtomatlashtirish vazifalarini samarali o'z ichiga oladi.

**Veb-ishlab chiqish tillari (masalan, HTML, CSS, JavaScript):**

****

**Html code**

****

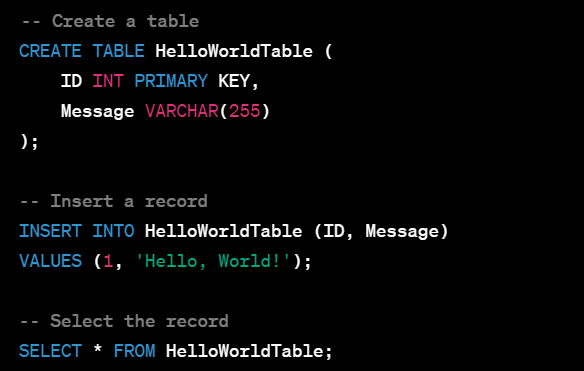
**Css code**

Veb-ishlab chiqishda mantiqiy fikrlash:

Veb-ishlab chiqishda mantiqiy fikrlash foydalanuvchi interfeyslarini loyihalash, asinxron operatsiyalarni boshqarish va mijoz-server o'zaro ta'sirini boshqarishni qamrab oladi.

Ishlab chiquvchilar HTML uchun mantiqiy tuzilmalardan, CSS uchun uslublardan va JavaScript bilan dinamik xatti-harakatlardan foydalanadilar.

**Ma'lumotlar bazasi so'rovlari tillari (masalan, SQL):**

****

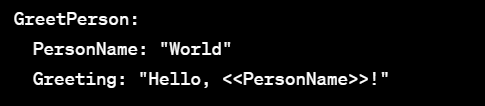
**SQL code**

Ma'lumotlar bazasi so'rovlari tillarida mantiqiy fikrlash:

SQLda mantiqiy fikrlash ma'lumotlarni olish, o'zgartirish va manipulyatsiya qilish uchun so'rovlarni loyihalashni o'z ichiga oladi. Ishlab chiquvchilar ma'lumotlar bazasi tuzilmalarini, munosabatlarini tushunishlari va so'rovlarni mantiqiy optimallashtirishlari kerak.

**Domenga xos tillar (DSL):**

DSL-larda mantiqiy fikrlash:

****

**DSL code**

DSL'lar muayyan muammoli domenlar uchun mo'ljallangan.

Mantiqiy fikrlash til xususiyatlarini domen talablariga aniq moslashtirishni o'z ichiga oladi.

A.p3. Hisoblash fikrlash ko'nikmalari dasturiy dizayn va ishlab chiqarilgan dasturiy ilovalarning sifatiga qanday ta'sir qilishi mumkinligini tahlil qilish.

Dasturiy ta'minotni loyihalash tamoyillari foydalanuvchilarning ehtiyojlarini qondiradigan yuqori sifatli dasturiy ilovalarni ishlab chiqishda hal qiluvchi rol o'ynaydi. Bu tamoyillar qanday qo'llanilishini ko’rib chiqamiz.

**Modullilik**: Dasturiy ta'minotni loyihalash tamoyillari murakkab tizimlarni kichikroq, boshqariladigan modullarga ajratishga urg'u beradi. Har bir modul ma'lum bir funksionallik yoki xususiyat uchun javobgar bo'lib, qayta foydalanish va texnik xizmat ko'rsatishni ta'minlaydi. Modulli dasturiy ta'minotni loyihalash orqali ishlab chiquvchilar butun tizimga ta'sir qilmasdan komponentlarni osongina qo'shishlari, o'zgartirishlari yoki almashtirishlari mumkin, bu esa o'lchov va moslashuvchanlikni osonlashtiradi.

* ***Encapsulation:*** Modullar o'zlarining ichki tafsilotlarini qamrab oladi va boshqa modullar bilan o'zaro ta'sir qilish uchun faqat aniq belgilangan interfeysni ochib beradi. Bu ma'lumotni yashirishga yordam beradi va modullarning ichki holatini himoya qiladi, bog'liqlikni kamaytiradi va kutilmagan nojo'ya ta'sirlarning oldini oladi.
* ***Reusability***: modulli dizayn modullarni turli kontekstlarda yoki loyihalarda qayta ishlatishga imkon berib, qayta foydalanishni osonlashtiradi. Ishlab chiqilgan va sinovdan o'tkazilgandan so'ng, modul kodini qayta yozish yoki o'zgartirishga hojat qoldirmasdan osongina boshqa tizimlarga kiritilishi mumkin.
* ***Maintainability***: Modullik muayyan modullarga o'zgarishlarni izolyatsiya qilish orqali texnik xizmat ko'rsatishni yaxshilaydi. O'zgartirishlar yoki yangilanishlar kerak bo'lganda, ishlab chiquvchilar tizimning qolgan qismiga ta'sir qilmasdan tegishli modullarga e'tibor qaratishlari mumkin. Bu texnik xizmat ko'rsatishni soddalashtiradi va xato yoki xatolarni kiritish xavfini kamaytiradi.
* ***Scalability:*** Modulli dizayn tizimlarni gorizontal yoki vertikal ravishda o'lchash imkonini berish orqali o'lchovni qo'llab-quvvatlaydi. Umumiy tizim arxitekturasini buzmasdan, qo'shimcha modullar yaratish yoki mavjudlarini kengaytirish orqali yangi funksionallikni qo'shish mumkin.
* ***Testability:*** Modulli tizimlarni sinash osonroq, chunki alohida modullar mustaqil ravishda tekshirilishi mumkin. Bu birlik sinovini osonlashtiradi, bunda har bir modul to'g'riligi va funksionalligini ta'minlash uchun alohida sinovdan o'tkaziladi. Bundan tashqari, modulli dizayn sinov paytida bog'liqliklarni taqlid qilish uchun soxta ob'ektlar yoki stublardan foydalanish imkonini beradi.

**Abstraktsiya**: Abstraktsiya keraksiz amalga oshirish tafsilotlarini yashirishni va asosiy xususiyatlar va xatti-harakatlarga e'tiborni qaratishni o'z ichiga oladi. Murakkablikni yo'qotib, dasturiy ta'minotni loyihalash tamoyillari ishlab chiquvchilarga foydalanuvchilarni tizimning asosiy murakkabligidan himoya qiladigan toza, tushunarli interfeyslarni yaratishga imkon beradi. Bu foydalanuvchi tajribasini soddalashtiradi va xato yoki chalkashlik ehtimolini kamaytiradi.

* Identification of Essential Concepts
* Abstraktsiya muammo sohasiga tegishli muhim tushunchalar yoki ob'ektlarni aniqlashdan boshlanadi. Ushbu tushunchalar muammoni tushunish va hal qilish uchun hal qiluvchi ahamiyatga ega bo'lgan ob'ektlar, jarayonlar yoki munosabatlarni o'z ichiga olishi mumkin.
* Removal of Implementation Details: Abstraktsiya muammoni tushunish uchun muhim bo'lmagan amalga oshirish tafsilotlarini olib tashlash yoki yashirishni o'z ichiga oladi. Bu murakkab tizimlarning namoyishini soddalashtiradi va ishlab chiquvchilarga past darajadagi amalga oshirishga emas, balki yuqori darajadagi dizaynga e'tibor qaratish imkonini beradi.
* Creation of Models: Abstraktsiya real dunyo ob'ektlari yoki tizimlarining muhim xususiyatlarini qamrab oluvchi modellarni yaratishga imkon beradi. Ushbu modellar muammoli sohaning soddalashtirilgan va tuzilgan ko'rinishini ta'minlaydi, bu esa dasturiy echimlarni tahlil qilish, loyihalash va amalga oshirishni osonlashtiradi.
* Use of Interfaces: Abstraktsiyaga ko'pincha sinf yoki modul rioya qilishi kerak bo'lgan shartnoma yoki xatti-harakatni belgilaydigan interfeyslardan foydalanish orqali erishiladi. Aniq interfeyslarni belgilab, ishlab chiquvchilar modulli va qayta foydalanish mumkin bo'lgan komponentlarni yaratishi mumkin, ular kattaroq tizimlarga osongina integratsiya qilinadi.
* Level of Detail: Abstraktsiya ishlab chiquvchilarga ular ishlashni xohlagan tafsilotlar darajasini aniqlash imkonini beradi. Yuqori darajadagi abstraktsiyalar tizimning keng ko'rinishini ta'minlaydi, past darajadagi abstraktsiyalar esa aniq tafsilotlar yoki amalga oshirishga qaratilgan. Ushbu moslashuvchanlik ishlab chiquvchilarga o'z ehtiyojlari va maqsadlaridan kelib chiqqan holda tegishli mavhumlik darajasida ishlash imkonini beradi.
* Simplification: Abstraktsiya muammo sohasining eng muhim jihatlariga e'tibor qaratish orqali dasturiy ta'minot tizimlarini loyihalash va amalga oshirishni soddalashtiradi. Bu murakkablik va kognitiv yukni kamaytiradi, bu esa ishlab chiquvchilarga vaqt o'tishi bilan dasturiy ta'minotni tushunish va saqlashni osonlashtiradi.

**Inkapsulyatsiya**: Inkapsulyatsiya - bu ma'lumotlar va usullarni bitta birlik yoki sinfga birlashtirish. Ushbu tamoyil ma'lumotlar yaxlitligini ta'minlashga yordam beradi va ob'ektlarning ichki holatini tashqi shovqinlardan himoya qiladi. Tegishli funktsiyalarni aniq belgilangan chegaralar ichida qamrab olgan holda, dasturiy ta'minotni loyihalash tamoyillari modullik, qayta foydalanish va barqarorlikni ta'minlaydi.

* ***Data Hiding***: Inkapsulyatsiya ob'ektning ichki holatini tashqi dunyodan yashirishga imkon beradi. Sinf ichidagi ma'lumotlarni inkapsulyatsiya qilish va unga faqat aniq belgilangan usullar (qabul qiluvchilar va sozlagichlar) orqali kirishni ta'minlash orqali ishlab chiquvchilar ma'lumotlarga qanday kirish va o'zgartirishni nazorat qilishlari mumkin, bu esa ruxsatsiz manipulyatsiyani oldini oladi.
* ***Modullik***: Inkapsulyatsiya tegishli ma'lumotlar va xatti-harakatlarni birlashtirilgan birliklarga qamrab olish orqali modullikni ta'minlaydi. Har bir sinf osongina tushunilishi, sinab ko'rilishi va saqlanishi mumkin bo'lgan mustaqil modulni ifodalaydi. Bu rivojlanish jarayonini soddalashtiradi va tizimning murakkabligini kamaytiradi.
* ***Information Security***: Inkapsulyatsiya maxfiy ma'lumotlarni ruxsatsiz kirish yoki o'zgartirishdan himoya qilish orqali axborot xavfsizligini oshiradi. Ma'lumotlarga kirishni boshqarish mexanizmlari (masalan, shaxsiy, himoyalangan va ommaviy) yordamida cheklash orqali ishlab chiquvchilar ma'lumotlarga kirishni va faqat mo'ljallangan tarzda o'zgartirilishini ta'minlashi mumkin.
* ***Code Reusability***: Inkapsulyatsiya sinflar ichida umumiy funksiyalarni qamrab olgan holda kodni qayta ishlatishni osonlashtiradi. Inkapsulatsiya qilinganidan so'ng, sinflar tizimning turli qismlarida yoki turli loyihalarda o'zgartirishlarsiz qayta ishlatilishi mumkin. Bu kodni qayta ishlatishga yordam beradi, ortiqchalikni kamaytiradi va umumiy samaradorlikni oshiradi.
* ***Encapsulation of Invariants***: Inkapsulyatsiya ishlab chiquvchilarga sinf ichidagi ma'lumotlar va xatti-harakatlarni inkapsulyatsiya qilish orqali invariantlarni (har doim to'g'ri bo'lishi kerak bo'lgan shartlar) qo'llash imkonini beradi. Ikkala ma'lumot va usullarni bir sinf ichida qamrab olgan holda, ishlab chiquvchilar ma'lumotlarga qanday kirish yoki o'zgartirishdan qat'i nazar, har doim ma'lumotlarning yaxlitligini ta'minlashi mumkin.
* ***Abstraction***: Inkapsulyatsiya sinfning amalga oshirish tafsilotlarini aniq belgilangan interfeys orqasida yashirish orqali abstraktsiyani targ'ib qiladi. Bu ishlab chiquvchilarga sinfning yuqori darajadagi funksionalligiga e'tiborni uning ichki amalga oshirilishi haqida qayg'urmasdan, tizimni loyihalash va texnik xizmat ko'rsatishni soddalashtirishga imkon beradi.

**Konsernlarni ajratish**: Konsernlarni ajratish tizimni har biri funksionallikning o'ziga xos jihati uchun mas'ul bo'lgan alohida qatlamlar yoki tarkibiy qismlarga bo'lish tarafdori. Xavotirlarni ajratib, dasturiy ta'minotni loyihalash tamoyillari ishlab chiquvchilarga tizimning turli jihatlarini mustaqil ravishda ajratish va hal qilish imkonini beradi, bu esa vaqt o'tishi bilan tushunish, saqlash va rivojlanishni osonlashtiradi. Bu, shuningdek, parallel ishlab chiqish va sinovlarni osonlashtiradi, umumiy samaradorlik va sifatni oshiradi.

* ***Mas'uliyatni aniq taqsimlash***: SoC tizimdagi foydalanuvchi interfeysi, biznes mantig'i va ma'lumotlarni saqlash kabi turli tashvishlar yoki mas'uliyatlarni aniqlashni o'z ichiga oladi. Har bir tashvish alohida modul, sinf yoki komponent tomonidan ko'rib chiqilishi kerak, bu esa ishlab chiquvchilarga butun tizimning murakkabligi tufayli e'tiborni jalb qilmasdan muayyan sohalarga e'tibor qaratish imkonini beradi.
* ***Modulli dizayn***: SoC tashvishlarni mustaqil, qayta foydalanish mumkin bo'lgan komponentlarga ajratish orqali modulli dizaynni targ'ib qiladi. Har bir modul bitta tashvishni qamrab olishi va boshqa modullar bilan o'zaro ta'sir qilish uchun aniq belgilangan interfeyslarni taqdim etishi kerak. Ushbu modulli tuzilma kodni qayta ishlatishni osonlashtiradi, testlarni soddalashtiradi va kengaytirilishini ta'minlaydi.
* ***Yaxshilangan texnik xizmat ko'rsatish***: Xavotirlarni ajratish orqali SoC tizimni tushunish, o'zgartirish va saqlashni osonlashtiradi. Ishlab chiquvchilar butun tizimga ta'sir qilmasdan alohida komponentlarga o'zgartirishlar kiritishlari mumkin, kutilmagan nojo'ya ta'sirlar xavfini kamaytiradi va muammolarni tuzatish va muammolarni bartaraf etishni osonlashtiradi.
* ***Kengaytirilgan moslashuvchanlik***: SoC ishlab chiquvchilarga tizimning boshqa qismlariga ta'sir qilmasdan alohida komponentlarni o'zgartirish yoki almashtirish imkonini berish orqali tizimning moslashuvchanligini oshiradi. Bu evolyutsion dizaynni osonlashtiradi, bu erda tizim vaqt o'tishi bilan o'zgaruvchan talablar yoki texnologiyalarga mos kelishi mumkin.
* ***Parallel rivojlanish***: SoC turli guruhlar yoki ishlab chiquvchilarga bir vaqtning o'zida alohida tashvishlar ustida ishlashga imkon berib, parallel rivojlanish imkonini beradi. Har bir jamoa boshqalarga aralashmasdan, rivojlanish jarayonini tezlashtirmasdan va hamkorlikni rag'batlantirmasdan, o'zining maxsus sohasiga e'tibor qaratishi mumkin.
* ***O'zaro bog'liq tashvishlar***: SoC xavotirlarni ajratish tarafdori bo'lsa-da, ba'zi tashvishlar bir nechta modullarni kesishi mumkinligini ham tan oladi. Ro'yxatga olish, xavfsizlik va xatolarni qayta ishlash kabi o'zaro bog'liq muammolar takrorlanishning oldini olish va izchillikni ta'minlash uchun modulli va qayta ishlatilishi mumkin bo'lgan tarzda hal qilinishi kerak.

**Oddiylik:** Oddiylik dasturiy ta'minot dizaynining asosiy printsipi bo'lib, u murakkablikni minimallashtirish va aniqlikni maksimal darajada oshirishga urg'u beradi. Oddiy dizaynlarni tushunish, disk raskadrovka qilish va saqlash osonroq bo'lib, xatolar ehtimolini kamaytiradi va umumiy ishonchlilikni oshiradi. Oddiylikka ustunlik berib, dasturiy ta'minotni loyihalash tamoyillari dasturiy ilovalarning intuitiv va foydalanuvchilarga qulay bo'lishini ta'minlashga yordam beradi va foydalanuvchilarning ehtiyojlarini samarali qondiradi.

* ***Aniq va ixcham kod***: Oddiylik uchun tushunarli, qisqa va tushunarli kod yozish juda muhimdir. Ishlab chiquvchilar tavsiflovchi o'zgaruvchilar nomlaridan foydalanishga harakat qilishlari, izchil kodlash konventsiyalarini saqlashlari va kodlarida keraksiz murakkablik yoki ortiqchalikdan qochishlari kerak.
* ***Minimalistik dizayn:*** Dizayndagi soddalik tizim arxitekturasi va dizaynlarini iloji boricha minimalistik saqlashni o'z ichiga oladi. Bu ortiqcha muhandislik va keraksiz xususiyatlardan qochish, faqat tizim talablarini qondirish uchun muhim bo'lgan narsalarga e'tibor berishni anglatadi.
* ***Yagona javobgarlik printsipi (SRP):*** Sinf yoki modulni o'zgartirish uchun faqat bitta sababga ega bo'lishi kerakligini ko'rsatadigan Yagona javobgarlik printsipiga rioya qilish soddalikni ta'minlaydi. Tizimning har bir komponenti aniq va o'ziga xos mas'uliyatga ega bo'lishini ta'minlash orqali ishlab chiquvchilar kodlar bazasini sodda va boshqarish oson bo'lishi mumkin.
* ***Erta optimallashtirishdan qochish***: Oddiylik ko'pincha aniqlik va barqarorlikni muddatidan oldin optimallashtirishdan ustun qo'yishni anglatadi. Ishlash yoki kengayishni muddatidan oldin optimallashtirish o'rniga, ishlab chiquvchilar osonlik bilan saqlanishi va kengaytirilishi mumkin bo'lgan toza, tushunarli kod yozishga e'tibor qaratishlari kerak.
* ***Hujjatlar va sharhlar***: Aniq hujjatlar va sharhlar turli komponentlarning maqsadi va funksionalligi haqida tushuncha berish orqali dasturiy tizimning soddaligini sezilarli darajada oshirishi mumkin. Yaxshi yozilgan hujjatlar yangi ishlab chiquvchilarga tizimni tezda tushunishga yordam beradi va doimiy texnik xizmat ko'rsatish va muammolarni bartaraf etishni osonlashtiradi.
* ***Foydalanuvchiga yo'naltirilgan dizayn***: Oddiylik kodlar bazasidan tashqari foydalanuvchi interfeysi va foydalanuvchi tajribasiga ham kiradi. Kognitiv yukni minimallashtirish va intuitiv ish oqimlarini ta'minlash kabi foydalanuvchiga yo'naltirilgan dizayn tamoyillari dasturiy ta'minotdan foydalanish va oxirgi foydalanuvchilar uchun tushunarli bo'lishini ta'minlaydi.
* ***Refaktoring***: Kodlar bazasini muntazam ravishda qayta ishlash vaqt o'tishi bilan soddalikni saqlash uchun zarurdir. Refaktoring kodni tashqi xatti-harakatini o'zgartirmasdan o'qilishi, barqarorligi va soddaligini yaxshilash uchun qayta qurishni o'z ichiga oladi. Kod bazasini doimiy ravishda takomillashtirish orqali ishlab chiquvchilar texnik qarzlar to'planishining oldini olishlari va tizimning sodda va boshqarilishi mumkin bo'lishini ta'minlashlari mumkin.

**Moslashuvchanlik va kengaytirilish**: Dasturiy ta'minotni loyihalash tamoyillari o'zgaruvchan talablar va muhitlarga oson moslashish imkonini beruvchi moslashuvchan va kengaytiriladigan dizaynlarni targ'ib qiladi. Moslashuvchanlikni hisobga olgan holda dasturiy ta'minotni loyihalash orqali ishlab chiquvchilar keng qamrovli qayta ishlash yoki qayta loyihalashni talab qilmasdan kelajakdagi yaxshilanishlar va o'zgartirishlarni qabul qilishlari mumkin. Bu dasturiy ilovalarning vaqt o'tishi bilan foydalanuvchilar uchun dolzarb va qimmatli bo'lib qolishini, rivojlanayotgan ehtiyojlar va texnologiyalarga moslashishini ta'minlaydi.

* ***Modulli arxitektura***: Modulli arxitektura tizimni mustaqil, almashtiriladigan modullarga bo'lish orqali moslashuvchanlikni ta'minlaydi. Har bir modul o'ziga xos funksionallik yoki xususiyatni qamrab oladi, bu butun tizimga ta'sir qilmasdan alohida komponentlarni o'zgartirish yoki almashtirishni osonlashtiradi. Ushbu modullilik moslashuvchan tizim konfiguratsiyasiga imkon beradi va yangi funksiyalarni qo'shishni osonlashtiradi.
* ***Open-Closed Principle (OCP):*** Open-Closed Principle dasturiy ta'minot ob'ektlari kengaytirish uchun ochiq, lekin o'zgartirish uchun yopiq bo'lishi kerakligini bildiradi. Mavjud kodlarini o'zgartirmasdan kengaytirilishi mumkin bo'lgan komponentlarni loyihalash orqali ishlab chiquvchilar tizim moslashuvchan va o'zgarishlarga chidamli bo'lishini ta'minlashi mumkin. Ushbu tamoyil o'zgartirish emas, balki kengaytma orqali kelajakdagi o'zgarishlarga moslashish uchun abstraktsiya, interfeyslar va polimorfizmdan foydalanishni rag'batlantiradi.
* ***Dizayn naqshlari***: Dizayn naqshlari umumiy dizayn muammolariga qayta foydalanish mumkin bo'lgan echimlarni taqdim etadi va moslashuvchanlik va kengayishni rag'batlantiradi. Strategiya namunasi, kuzatuvchi namunasi va zavod namunasi kabi naqshlar ishlab chiquvchilarga komponentlarni ajratish, harakatni dinamik ravishda o'zgartirish va mos ravishda ularning aniq turlarini ko'rsatmasdan ob'ektlar yaratish imkonini beradi. Ushbu naqshlarni qo'llash orqali ishlab chiquvchilar texnik xizmat ko'rsatish, kengaytirish va o'zgaruvchan talablarga moslashish osonroq bo'lgan tizimlarni loyihalashlari mumkin.
* ***Dependency Injection (DI***): Dependency Injection - bu qaramliklarni tashqi ko'rinishga keltirish orqali komponentlar o'rtasida bo'shashmasdan bog'lanishga erishish usuli. Komponentlarga ularni qattiq kodlash o'rniga bog'liqliklarni kiritish orqali ishlab chiquvchilar tizimni yanada moslashuvchan va kengaytirilishi mumkin. DI komponentlarni ish vaqtida osongina almashtirish yoki sozlash imkonini beradi, bu modulli dizayn va sinovdan o'tishni osonlashtiradi.
* ***Plug-in arxitekturasi***: Plagin arxitekturasi ishlab chiquvchilarga plaginlarni dinamik ravishda qo'shish yoki olib tashlash orqali tizimning funksionalligini kengaytirish imkonini beradi. Plaginlar - bu tizimning asosiy kod bazasini o'zgartirmasdan, uning imkoniyatlarini oshirish uchun moslashuvchan mexanizmni ta'minlovchi, ish vaqtida yuklanishi va tushirilishi mumkin bo'lgan mustaqil modullar. Ushbu yondashuv odatda veb-brauzerlar, kontentni boshqarish tizimlari va integratsiyalashgan rivojlanish muhitlari (IDE) kabi ilovalarda qo'llaniladi.
* ***Versiyalash va moslik***: Versiyalash va moslikni hisobga olgan holda tizimlarni loyihalash ularning vaqt o'tishi bilan moslashuvchan va kengaytirilishini ta'minlaydi. Aniq versiya siyosatlarini belgilash va oldingi versiyalar bilan orqaga qarab muvofiqlikni saqlab qolish orqali ishlab chiquvchilar mavjud foydalanuvchilarni yoki integratsiyani buzmasdan o'zgartirishlar va yaxshilanishlarni kiritishlari mumkin. Bu barqarorlik va uzluksizlikni saqlab qolgan holda tizimni bosqichma-bosqich rivojlantirish imkonini beradi.

Dasturiy ta'minotni loyihalash tamoyillari foydalanuvchilarning ehtiyojlarini qondiradigan yuqori sifatli dasturiy ta'minotni ishlab chiqarish uchun zarurdir. Ushbu tamoyillarga rioya qilgan holda, ishlab chiquvchilar modulli, mavhum, inkapsullangan, yaxshi tuzilgan, sodda, moslashuvchan va kengaytiriladigan dasturiy ta'minotni yaratishi mumkin, natijada foydalanuvchining yuqori tajribasini taqdim etadi va foydalanuvchi qoniqishini oshiradi.

1. Topshiriq 2

B.P4. Mijoz talablariga mos keladigan kompyuter dasturi uchun dizayn yaratish.

Mijozlarning talablariga javob beradigan kompyuter dasturi dizaynini ishlab chiqarish uchun biz dasturiy yechimning tuzilishi, funksionalligi va tarkibiy qismlarini ko'rsatishimiz kerak. Mijoz talablari asosida omborni boshqarish tizimi uchun yuqori darajadagi dizayn:

**Foydalanuvchi interfeysi (UI):**

UI foydalanuvchilarga ombor tizimi bilan o'zaro aloqada bo'lishlari uchun intuitiv interfeysni ta'minlashi kerak.

U mahsulotlarni qo'shish, mahsulot miqdorini yangilash, mahsulot miqdorini olish, qolmagan mahsulotlar ro'yxatini yaratish va barcha mahsulotlarni ko'rsatish imkoniyatlarini o'z ichiga olishi kerak.

Oson navigatsiya uchun kiritish shakllari va tugmalari aniq belgilanishi kerak.

**Ma'lumotlar modeli:**

Ombordagi mahsulotlarni ko'rsatish uchun ma'lumotlar modelini aniqlang.

Har bir mahsulot nomi va miqdori kabi atributlarga ega bo'lishi kerak.

**Funktsionallik:**

***Mahsulot qo'shish:***

Foydalanuvchiga mahsulot nomi va miqdorini kiritishni taklif qiling.

Mahsulotni tizim ma'lumotlar bazasiga qo'shing.

***Mahsulotni yangilash:***

Foydalanuvchidan yangilanadigan mahsulot nomini kiritishni taklif qiling.

Foydalanuvchiga tanlangan mahsulot miqdorini yangilashiga ruxsat bering.

***Mahsulot miqdorini oling:***

Foydalanuvchiga mahsulot nomini kiritishni taklif qiling.

Tanlangan mahsulot miqdorini ko'rsating.

***Qolgan mahsulotlar roʻyxatini yarating:***

Miqdori 0 bo'lgan mahsulotlarni aniqlang va qolmagan mahsulotlar ro'yxatini tuzing.

***Barcha mahsulotlarni ko'rsatish:***

Ma'lumotlar bazasidan barcha mahsulotlarni oling va ularning nomlari va miqdorini ko'rsating.

***Ma'lumotlarni saqlash:***

Mahsulot ma'lumotlarini doimiy ravishda saqlash mexanizmini joriy qiling.

Mahsulot ma'lumotlarini saqlash uchun ma'lumotlar bazasi yoki fayl xotirasidan foydalanishni o'ylab ko'ring.

***Foydalanuvchi kiritishiga ishlov berish:***

Kutilgan format va diapazonga mos kelishiga ishonch hosil qilish uchun foydalanuvchi kiritishini tasdiqlang.

Istisnolarni ehtiyotkorlik bilan bajaring va foydalanuvchiga mazmunli xato xabarlarini taqdim eting.

***Sinov:***

Dasturiy ta'minotning funksionalligini tekshirish uchun keng qamrovli sinov rejasini ishlab chiqamiz. Har bir komponentni, shu jumladan chekka holatlar va chegara shartlarini yaxshilab sinab ko'ring.

***Hujjatlar:***

Dasturiy ta'minotni loyihalash va amalga oshirish tafsilotlarini hujjatlashtiring.

Omborni boshqarish tizimini ishlatish uchun foydalanuvchi qo'llanmalari va qo'llanmalarini taqdim eting.

**Foydalanuvchining fikr-mulohazalari:**

Rivojlanish jarayonida foydalanuvchilarning fikr-mulohazalarini to'plash va yaxshilash kerak bo'lgan sohalarni aniqlash.

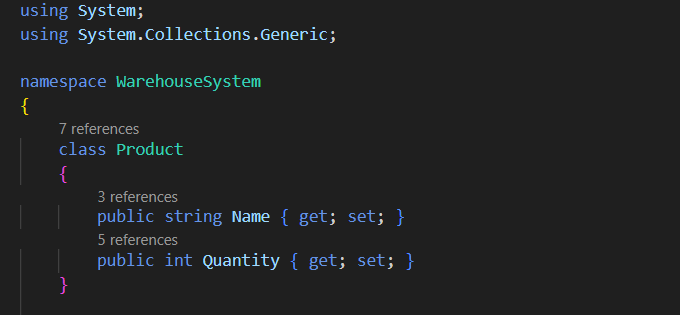
Foydalanuvchi takliflarini kiriting va foydalanishga oid har qanday muammolarni hal qiling.

Ushbu dizaynga rioya qilish orqali biz mijozning talablariga samarali javob beradigan mustahkam va foydalanuvchilarga qulay omborlarni boshqarish tizimini yaratishimiz mumkin.

B.P5. Taklif qilingan yechimlarga o'zgartirishlar kiritish uchun boshqalar bilan dizaynni ko'rib chiqish

Dizaynni ko'rib chiqish dasturiy ta'minotni ishlab chiqish jarayonida hal qiluvchi bosqich bo'lib, taklif qilingan echimlar talablar va sanoatning ilg'or amaliyotlariga mos kelishini ta'minlaydi. Ushbu bosqichda tengdoshlar va manfaatdor tomonlar bilan hamkorlik shaffoflikni ta'minlaydi, yuzaga kelishi mumkin bo'lgan muammolarni erta aniqlaydi va dasturiy ta'minotning umumiy sifatini oshiradi. Dizaynni ko'rib chiqish jarayonida jamoa a'zolari va tengdoshlari bilan shug'ullanish turli xil istiqbollar va tushunchalarni beradi. Har bir ishtirokchi o'ziga xos mahorat va tajribaga ega bo'lib, tuzoqlarni aniqlaydigan, yaxshilanishlarni taklif qiladigan va mustahkam dizaynni kafolatlaydigan keng qamrovli sharhni qo'llab-quvvatlaydi. Dasturiy ta'minotga bevosita ta'siridan tashqari, dizayn sharhlari bilim almashish uchun platforma yaratadi. Guruh a'zolari tajriba almashadilar, qasddan dizayn qarorlarini qabul qilishadi va domen bilimlarini uzatishadi. Ushbu hamkorlikdagi almashinuv nafaqat dizaynni yaxshilaydi, balki jamoa a'zolarining kasbiy rivojlanishiga hissa qo'shib, uzluksiz o'rganish madaniyatini rivojlantiradi. Aslini olganda, dizayn ko'rib chiqishlari dasturiy ta'minotni ishlab chiqish hayotiy tsiklida asosiy omil bo'lib xizmat qiladi, bu nafaqat texnik mukammallikni ta'minlaydi, balki hamkorlik va bilimga asoslangan jamoa madaniyatini targ'ib qiladi.

C.P6. Mijoz talablariga javob beradigan kompyuter dasturini yaratish



**Mahsulot klassi:**

Bu sinf ombordagi mahsulotni ifodalaydi.

U ikkita xususiyatga ega: mahsulot nomini saqlash uchun nom (string) va mahsulot miqdorini saqlash uchun Quantity (int).

**Ishlatilgan kutubxonalar:**

**Tizim:** Ko'p ishlatiladigan qiymat va mos yozuvlar ma'lumotlari turlarini, hodisalar va hodisalarni qayta ishlash vositalarini, interfeyslarni, atributlarni va ishlov berish istisnolarini belgilaydigan asosiy turlar va asosiy sinflarni taqdim etadi.

**System.Collections.Generic:** Ro'yxatlar, lug'atlar, navbatlar va steklar kabi umumiy to'plam sinflarini taqdim etadi.

**Amaldagi ma'lumotlar turlari:**

**string:** Belgilar ketma-ketligini ifodalaydi.

int: 32 bitli imzolangan butun sonni ifodalaydi.

**Sinf ta'rifi:**

Mahsulot: Ombordagi mahsulotni ifodalash uchun sinfni belgilaydi.

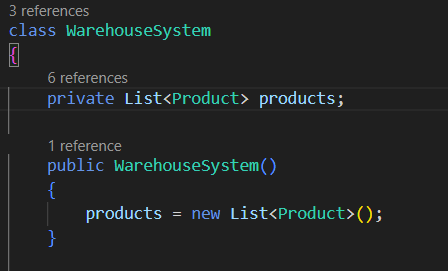
**Xususiyatlari:**

Ism: mahsulot nomini saqlash uchun string tipidagi xususiyat.

Miqdor: mahsulot miqdorini saqlash uchun int turidagi xususiyat.

csharp

Kodni nusxalash



**Ombor tizimi klassi:**

Bu sinf ombor operatsiyalarini boshqaradi.

Unda mahsulotlar ro'yxatini saqlash uchun List<Mahsulot> turidagi shaxsiy maydon mahsulotlari mavjud.

Konstruktor mahsulotlar ro'yxatini ishga tushiradi.

Amaldagi ma'lumotlar turlari:

**List<T>:** Belgilangan turdagi ob'ektlarning qattiq terilgan ro'yxatini ifodalaydi.

Mahsulot: Ro'yxatda saqlangan ob'ektlar turi.

Sinf ta'rifi:

**WarehouseSystem:** Ombor operatsiyalarini boshqarish uchun sinfni belgilaydi.

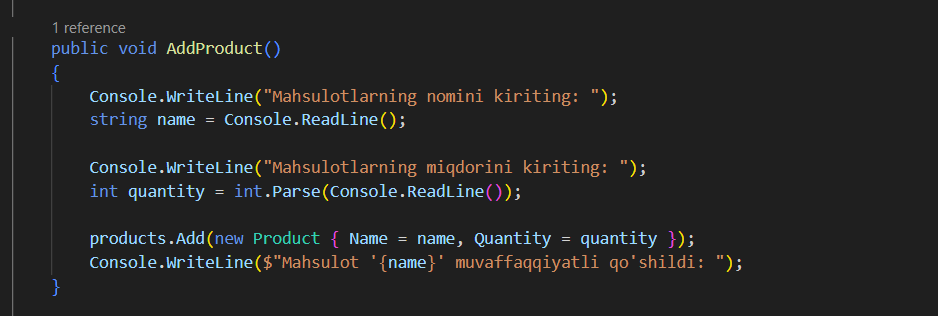
Maydon:

**private List<Mahsulot> mahsulotlar;:** Mahsulotlar roʻyxatini saqlash uchun shaxsiy maydonni eʼlon qiladi.

Konstruktor:

**public WarehouseSystem():** WarehouseSystem sinfining yangi nusxasini ishga tushiradi.

**mahsulotlar = new List<Mahsulot>();:** Mahsulotlar roʻyxatini ishga tushiradi.



**AddProduct usuli:**

Foydalanuvchidan yangi mahsulot nomi va miqdorini kiritishni taklif qiladi.

Kiritilgan nom va miqdor bilan yangi Mahsulot ob'ektini yaratadi va uni mahsulotlar ro'yxatiga qo'shadi.

Mahsulot qo'shilgandan so'ng muvaffaqiyatli xabarni ko'rsatadi.

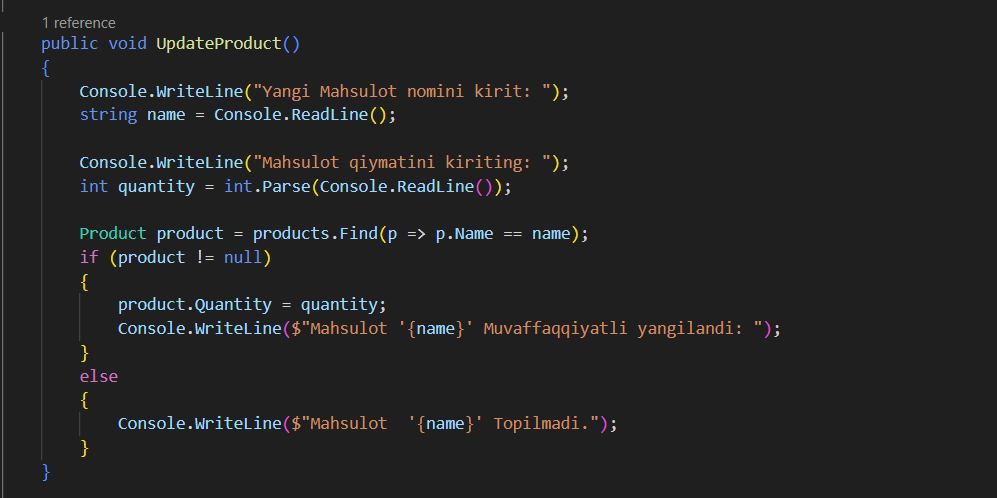
**Qo'llaniladigan usullar:**

**Console.WriteLine():** Belgilangan satr qiymatini standart chiqish oqimiga yozadi.

**Console.ReadLine():** standart kirish oqimidan keyingi belgilar qatorini o'qiydi.

**int.Parse():** Raqamning satr tasvirini uning 32-bitli imzolangan tamsayı ekvivalentiga aylantiradi.

**List<T>.Add():** Ob'ektni ro'yxat oxiriga qo'shadi.



**Mahsulotni yangilash usuli:**

Foydalanuvchidan mavjud mahsulot nomini va yangi miqdorini kiritishni taklif qiladi.

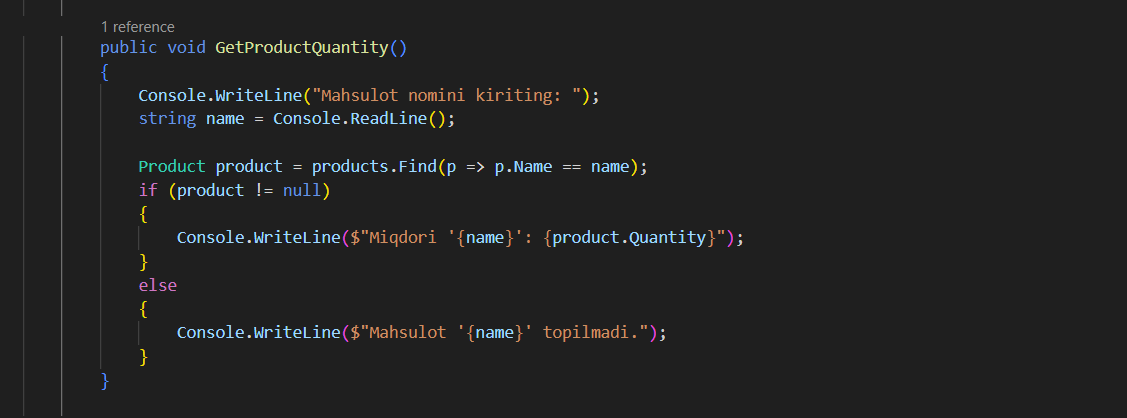
Mahsulotlar ro'yxatida kiritilgan nomi bilan mahsulotni topadi.

Agar mahsulot topilsa, uning miqdori yangi qiymat bilan yangilanadi.

Agar mahsulot topilmasa, mahsulot topilmaganligini bildiruvchi xabar paydo bo'ladi.

**Qo'llaniladigan usullar:**

***List<T>.Find():*** Belgilangan predikat bilan belgilangan shartlarga mos keladigan elementni qidiradi.



**GetProductQuantity usuli:**

Foydalanuvchiga mahsulot nomini kiritishni taklif qiladi.

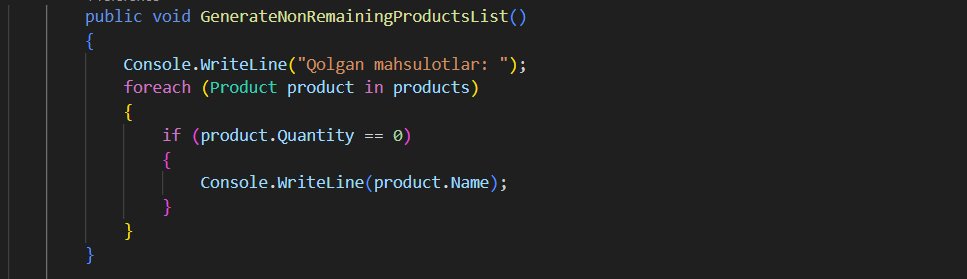
Mahsulotlar ro'yxatida kiritilgan nomi bilan mahsulotni topadi.

Agar mahsulot topilsa, uning miqdori ko'rsatiladi.

Agar mahsulot topilmasa, mahsulot topilmaganligini bildiruvchi xabar paydo bo'ladi.

Qo'llaniladigan usullar:

**List<T>.Find():** Belgilangan predikat bilan belgilangan shartlarga mos keladigan elementni qidiradi.



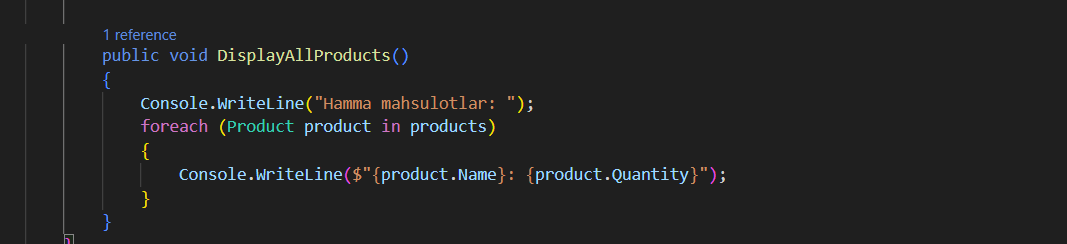
GenerateNonRemainingProductsList usuli:

Mahsulotlar ro'yxatidagi barcha mahsulotlarni takrorlaydi.

0 (qolgan mahsulotlar) miqdori bo'lgan mahsulotlarning nomlarini chop etadi.

**Takrorlash:**

***foreach tsikli:*** Mahsulotlar ro'yxatidagi har bir mahsulot ustidan takrorlanadi.

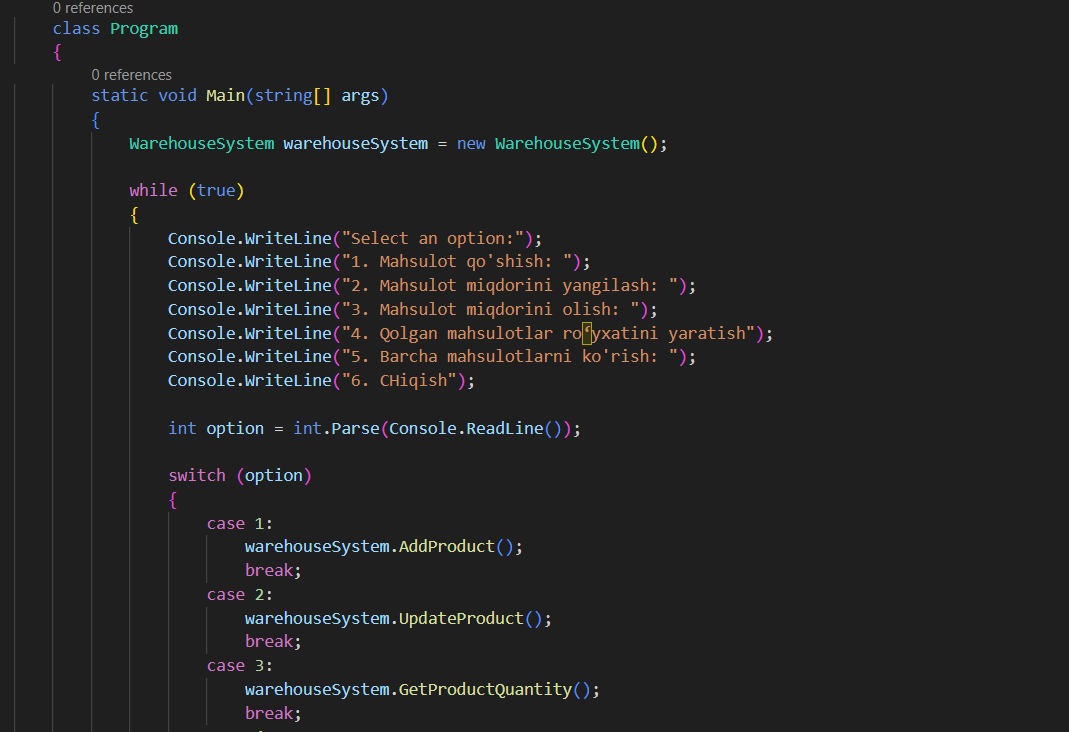


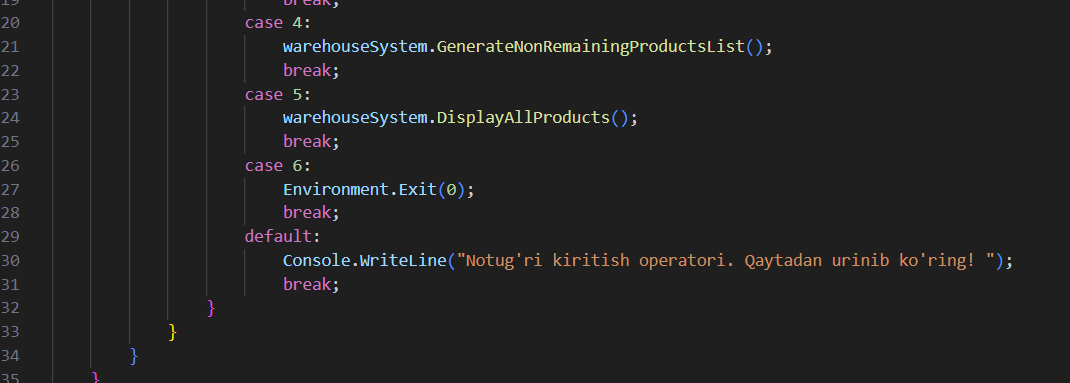
**DisplayAllProducts usuli:**

Barcha mahsulotlarni ularning miqdori bilan birga ko'rsatadi.

**Takrorlash:**

***foreach tsikli:*** Mahsulotlar ro'yxatidagi har bir mahsulot ustidan takrorlanadi.





**Dastur sinfi (Asosiy usul):**

Main usuli WarehouseSystem sinfining namunasini yaratadi.

U foydalanuvchiga variantlar menyusini ko'rsatish va foydalanuvchi kiritgan ma'lumotlar asosida tegishli operatsiyalarni bajarish uchun cheksiz tsiklni ishga tushiradi.

Variantlar:

1: Mahsulot qo'shing.

2: Mahsulot miqdorini yangilang.

3: Mahsulot miqdorini oling.

4: Qolgan mahsulotlar ro'yxatini yarating.

5: Barcha mahsulotlarni ko'rsatish.

6: Dasturdan chiqing.

**Qo'llaniladigan usullar:**

***Console.WriteLine():*** Belgilangan satr qiymatini standart chiqish oqimiga yozadi.

***Console.ReadLine():*** standart kirish oqimidan keyingi belgilar qatorini o'qiydi.

***int.Parse():*** Raqamning satr tasvirini uning 32-bitli imzolangan tamsayı ekvivalentiga aylantiradi.

Environment.Exit(): Belgilangan chiqish kodi bilan dasturdan chiqadi.

**Loop:**

***while loop:*** Foydalanuvchi dasturdan chiqishni tanlamaguncha menyu parametrlarini qayta-qayta bajaradi.

**Switch bayonoti:**

***switch bayonoti:*** Variant qiymatiga qarab bajariladigan ko'plab kod bloklaridan birini tanlaydi.

C.P7. Yakuniy kompyuter dasturi mijoz talablariga qanchalik mos kelishini ko'rib chiqish

Yakuniy kompyuter dasturiy ta'minoti mijozning talablariga qanchalik mos kelishini baholash dasturiy ta'minotni ishlab chiqish jarayonida hal qiluvchi qadam bo'lib, etkazib beriladigan yechim mijozning ehtiyojlari va maqsadlariga samarali javob berishini ta'minlaydi. Ushbu baholash dasturiy ta'minotning turli jihatlarini dastlab belgilangan talablar va spetsifikatsiyalar bilan taqqoslaganda tizimli baholashni o'z ichiga oladi.

Birinchidan, mijozning asl talablari bo'yicha hujjatlarni, shu jumladan har qanday foydalanuvchi hikoyalari, foydalanish holatlari yoki funktsional spetsifikatsiyalarni batafsil ko'rib chiqish juda muhimdir. Bu dasturiy ta'minot uchun kutilgan xususiyatlar, funksiyalar va ishlash mezonlarini aniq tushunish imkonini beradi.

Keyinchalik, dasturiy ta'minotning har bir komponenti muvofiqligini aniqlash uchun tegishli talab bo'yicha baholanadi. Bu foydalanuvchi interfeysi, funksionalligi, ma'lumotlarni qayta ishlash imkoniyatlari, xavfsizlik choralari va dasturiy ta'minotning boshqa tegishli jihatlarini o'rganishni o'z ichiga oladi. Haqiqiy amalga oshirish va belgilangan talablar o'rtasidagi har qanday og'ishlar yoki nomuvofiqliklar aniqlanadi va hujjatlashtiriladi.

Bundan tashqari, dasturiy ta'minotdan foydalanish qulayligi va foydalanuvchi tajribasi uning foydalanish qulayligi, qulaylik va intuitivlik nuqtai nazaridan mijozning talablariga javob berishiga ishonch hosil qilish uchun baholanadi. Bu oxirgi foydalanuvchilarning dasturiy ta'minot bilan o'zaro aloqalari bo'yicha fikr-mulohazalarini yig'ish uchun vakillik bilan yaroqlilik sinovlari sessiyalarini o'tkazishni o'z ichiga olishi mumkin.

Funktsional talablardan tashqari, unumdorlik, ishonchlilik, masshtablilik va xavfsizlik kabi funktsional bo'lmagan talablar ham baholanadi. Dasturiy ta'minotning javob vaqtini, resurslardan foydalanishni va turli xil yuklash sharoitlarida o'tkazish qobiliyatini baholash uchun unumdorlik testi o'tkazilishi mumkin. Xavfsizlikni baholash zaifliklarni aniqlashga yordam beradi va dasturiy ta'minot va uning ma'lumotlarini himoya qilish uchun tegishli choralar ko'rilishini ta'minlaydi.

Baholash jarayonida har qanday noaniqliklarga oydinlik kiritish, tashvishlarni bartaraf etish va dasturiy ta'minotning ularning kutganlariga mos kelishini tasdiqlash uchun mijoz bilan muloqot qilish zarur. Mijoz va oxirgi foydalanuvchilarning fikr-mulohazalari yig'iladi va baholash natijalariga kiritiladi.

Nihoyat, har qanday aniqlangan muammolar, muvofiqlik sohalari va keyingi takomillashtirish bo'yicha tavsiyalarni o'z ichiga olgan baholash natijalarini hujjatlashtiradigan keng qamrovli hisobot tayyorlanadi. Ushbu hisobot manfaatdor tomonlar uchun qimmatli ma'lumotnoma bo'lib xizmat qiladi va yetkazib berilgan dasturiy ta'minot mijozning talablari va maqsadlariga javob berishiga ishonch hosil qiladi.

Yakuniy kompyuter dasturiy ta'minoti mijozning talablariga qanday mos kelishini baholash dasturiy ta'minotni ishlab chiqishning muhim jihati bo'lib, etkazib beriladigan yechim mijozning ehtiyojlari, umidlari va biznes maqsadlariga mos kelishini ta'minlaydi. Bu tafsilotlarga jiddiy e'tibor berishni, puxta baholash jarayonlarini va loyihaning butun hayoti davomida manfaatdor tomonlar bilan samarali muloqotni talab qiladi.

. XULOSA

Xulosa qilib aytganda, dasturlashni rivojlantirish bo'yicha amalga oshiriladigan puxta rejalashtirish, tirishqoqlik bilan bajarish va doimiy takomillashtirish bilan ajralib turadi. Ushbu jarayon davomida biz mijozlar talablari, dizayn tamoyillari, ishlab chiqish metodologiyalari va loyihalarni boshqarish strategiyalari o'rtasidagi murakkab o'zaro bog'liqlikni o'rganib chiqdik. Kontseptuallashtirish va loyihalashning dastlabki bosqichlaridan boshlab amalga oshirish va baholashning yakuniy bosqichlarigacha bo'lgan har bir bosqich muhim ahamiyatga ega va yuqori sifatli dasturiy echimlarni taqdim etishning asosiy maqsadiga hissa qo'shadi. Bizning tadqiqotlarimiz dasturiy ta'minotni ishlab chiqish bo'yicha sa'y-harakatlarni mijozlar ehtiyojlari va kutishlariga moslashtirish muhimligini ta'kidladi. Samarali muloqot, hamkorlik va fikr-mulohazalarga javob berishni birinchi o'ringa qo'yish orqali ishlab chiquvchilar o'zlari taqdim etayotgan dasturiy ta'minot nafaqat manfaatdor tomonlar talablariga javob berishini, balki undan ham oshib ketishini ta'minlashi mumkin. Bundan tashqari, bizning tahlilimiz dasturlashni rivojlantirish bo'yicha sa'y-harakatlar muvaffaqiyatiga erishishda individual mas'uliyat, ijodkorlik va samarali o'zini o'zi boshqarishning muhim rolini ta'kidladi. Hisobdorlik, innovatsiyalar va doimiy takomillashtirish madaniyatini rivojlantirish orqali ishlab chiqish guruhlari dasturiy ta'minotni ishlab chiqishning murakkabliklarida ishonch va chaqqonlik bilan harakat qilishlari mumkin. Oldinga qarab, keyingi o'rganish va harakat qilish uchun bir nechta yo'llar mavjud. Birinchidan, rivojlanayotgan texnologiyalar, metodologiyalar va ilg'or tajribalar bo'yicha doimiy izlanishlar sohadagi yutuqlardan xabardor bo'lish uchun zarur bo'ladi. Bundan tashqari, rivojlanish guruhlari ichida hamkorlik, aloqa va loyihalarni boshqarishni yaxshilash bo'yicha davom etayotgan sa'y-harakatlar samaradorlik va mahsuldorlikni optimallashtirish uchun hal qiluvchi ahamiyatga ega bo'ladi. Bundan tashqari, texnologiyaning doimiy rivojlanib borayotgan tabiati va tarmoqlar bo'ylab dasturiy echimlarga bo'lgan ishonch ortib borayotganini hisobga olgan holda, fanlararo hamkorlik va o'zaro funktsional tajribaga ehtiyoj ortib bormoqda. Kelajakdagi tadqiqotlar innovatsiyalarni rivojlantirish va dasturiy ta'minotni ishlab chiqishdagi murakkab muammolarni hal qilish uchun turli ko'nikmalar to'plamlari, istiqbollari va metodologiyalarining integratsiyasini o'rganishi mumkin. Dasturlashni ishlab chiqish ham texnik tajriba, ham strategik bashoratni talab qiladigan dinamik va ko'p qirrali ishdir. Mijozlarga yo'naltirilganlik, hamkorlik va doimiy takomillashtirish tamoyillarini o'zlashtirgan holda, ishlab chiquvchilar dasturiy ta'minotni ishlab chiqishning murakkabliklarida ishonch bilan harakat qilishlari va foydalanuvchilarning hayotini kengaytiradigan va boyitadigan echimlarni taqdim etishlari mumkin.

Manbalar ro’yXati

Manbalar ro’yxati harvardda bo’lishi kerak

**So’zlar soni:**