Жамбыл облысы әкімдігі білім басқармасы

«Білім» кәсіби гуманитарлық-техникалық колледжі

(білім беру ұйымының атауы)

**Оқу сабағының жоспары**

(теориялық немесе өндірістік оқыту)

Компоненттік әдіс және CASE операторы

(сабақ тақырыбы)

**Модуль/Пән атауы** Ішкі тапсырмалардың өзара әрекеттесу сызбасын жасап, оларды бір тапсырмаға әзірлеу

**Дайындаған педагог** Нургисаева У.М

**20\_**25**\_ жылғы** «\_30\_\_\_» \_\_\_қаңтар \_\_\_\_

**1. Жалпы мәліметтер**

Курс, оқу жылы, топ 2 курс, 2БҚ-23

Сабақ түрі: Тәжірибелік

**2. Мақсаты, міндеттері:**

**Оқу:** Компоненттік әдістің теориялық негіздерін меңгеру. CASE технологияларының рөлін және олардың бағдарламалық қамтамасыз етуді әзірлеудегі маңызын түсіну.

#### Дамыту: Бағдарламалаудағы компоненттік тәсілдерді тиімді қолдану қабілетін жетілдіру.

#### Тәрбиелік: Заманауи бағдарламалау әдістеріне қызығушылықты арттыру және үздіксіз өзін-өзі жетілдіруге ынталандыру.

**3. Оқу-жаттығу процесінде білім алушылар меңгеретін күтілетін нәтижелер және кәсіби дағдылар тізбесі:** CASE технологияларын қолданып, бағдарламалық жүйелерді жобалай білу. CASE операторының синтаксисін меңгеру және оны әртүрлі бағдарламалау тілдерінде пайдалана алу.

**4. Қажетті ресурстар:** ДК немесе ноутбуктер

***5. Сабақтың барысы: (90 минут)***

**5.1. Ұйымдастыру кезеңі:** *( 3 мин )*

**5.2. Үй жұмысын жан-жақты тексеру:**

***“Менен сұрақ - Сізден жылдам жауап” әдісі*** *(15 минут)*

**Компоненттік әдіс және CASE операторы**

**Компоненттік әдіс және CASE операторы. ( 90 – жылдардың ортасынан бастап қазіргі кезге дейін)**

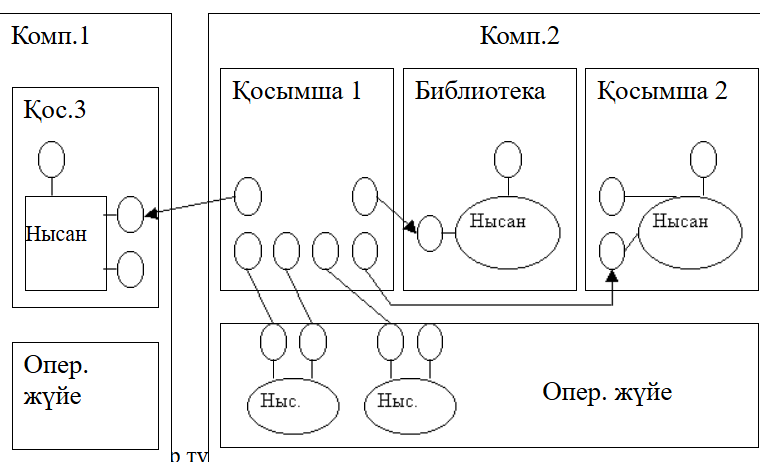
Компоненттік әдіс әр түрлі компоненттерден тұратын бір – біріне қатыссыз программа бөліктерін бір стандартты библиотекаға біріктіруге арналған, осы бөліктер бір-бірімен екілік интерфейстердің стандарттары арқылы байланыс орната алатын программа құруға арналған. Қарапайым нысандардан компоненттердің нысандарының айырмашылығы  осы нысандардың қасиеттері бойынша жинақтап шақыруға арналған динамикалық библиотека немесе орындалуға дайын файлдар күйінде беріледі. Динамикалық библиотекадағы нысандар екілік код түрінде берілгендіктен кез – келген программалық тілде қолдана беруге болады. COM (компонент объект модуль) базасының технологиясының және қосымшаларды таратудың  CORBA технологиясын құруға компоненттік әдісті негізі ретінде алынған. CORBA (сұраныстардың нысандарын өңдеуге арналған архитектураның жалпы түрі)  COM  технологиясы Microsoft фирмаларды Windows жүйесіндегі алғашқы нұсқаларындағы қосымша құжаттарды жетілдіріп қолдануға арналған OLE 1 технологиясын танытты (OLE 1 – нысандарды байланыстыру және енгізу). Әр түрлі типтегі файлдарды әр түрлі  процестегі бір немесе бірнеше компьютерлерде COM технологиясы арқылы ақпараттарды алмастыра беруге болады. Компьютерлер арасындағы ақпараттарды алмастыруды қамтамасыз ететін модификацияланған СОМ түрін DCOM деп атайды. DCOM (COM бойынша ақпараттарды тарату) қосымшалардың қызметін қамтамасыз ететін арнайы COM нысанын қолданады. Осы нысан COM класының данасы болып табылады.  COM нысаны қарапайым нысандар сияқты өрістер мен тәсілдерден тұрады. Қарапайым нысандардан әрбір COM нысанының айырмашылығы өрістер арқылы функцияларға қатынас орнатуға арналған бірнеше интерфейстерді қолдана алады. Осы мүмкіншіліктерді әрбір  интерфейске жеке –  жеке тәсілдердің адрестерін сақтайтын кесте құру арқылы орындалады. COM кластары интерфейсіндегі мұрагерлерін қолдайды.  Ал мұрагерлерді қолдауды сүйемелдейтін немесе сол интерфейстің тәсілдер кодын сүйемелдейді. Әрбір интерфейстің I әрпінен басталатын аттары және де IID деп аталатын уникалды, ауқымды идентификаторы сервердің құрамына кез – келген нысан кіруі мүмкін. Сервердің 3 түрі бар:

а) ішкі сервер динамикалық библиотекада қолданылады. Клиенттердің қосымшаларына қосылу арқылы адрестер кеңістігін бірге қолданады. Ішкі серверлер қосымша құралдарды керек етпегендіктен ең ыңғайлы сервер болып табылады;

б) төңіректік (локальді) сервер. Клиенттермен бір компьютерде жұмыс істей алатындай жеке ұйымдастырылады;

в) қашықтағы сервер басқа компьютерде жұмыс істейді.

Мысалы: Microsoft Word -  жергілікті сервер болып табылады. Жергілікті және қашықтағы сервер арасындағы байланыс COM және DCOM технологиясы бойынша орындалады. Қолданушы COM  нысанының көмекшісі froxy нысанының, ал адрестер кеңістігінде COM  нысанына арналған немесе сол клиентке арналған бітегіші (закрыватель) орналастырылады. Көмекші клиенттен тапсырманы қабылдап операциялық жүйенің қызмет көрсету бөлігі арқылы бітегішке тапсырады. Бітегіш тапсырманы анықтап COM нысанына өткізеді.Алынған шешім осы адрестер арқылы кері жүреді. OLE технологиясы ішкі қызмет көрсету жүйесіне арнайы программамен қамтамасыз етеді. Осы жүйеде нысанның шақыратын интерфейсті n интерфейсі деп атайды. OLE базасында құрылған Activex технологиясы бір компьютерге немесе таратылған желіге қызмет көрсететін программа ұйымдастырады.



Бағдарламалауда және жүйелік жобалауда компоненттік әдіс пен CASE (Computer-Aided Software Engineering) технологиялары маңызды рөл атқарады. Компоненттік әдіс бағдарламалық жасақтама жасаудың модульдік тәсілі болып табылады, ал CASE құралдары бағдарламалық жүйелерді құрастыру, тестілеу және сүйемелдеу процесін автоматтандырады.

**1. Компоненттік әдіс**

Компоненттік әдіс – бұл бағдарламалық жүйелерді жеке, өзара байланысқан компоненттерден құрастыру тәсілі. Компонент – белгілі бір функционалды орындайтын тәуелсіз бағдарламалық модуль. Бұл әдіс үлкен бағдарламалық жүйелерді құру, оңтайландыру және қолдау барысында тиімді шешім болып табылады, өйткені ол кодты қайта пайдалану, жүйені модульдік негізде дамыту және жаңарту мүмкіндігін ұсынады.

**1.2. Компоненттік әдістің негізгі принциптері**

1. **Модульдік құрылым** – бағдарламалар бір-бірінен тәуелсіз модульдерден (компоненттерден) тұрады.
2. **Қайта пайдалану (Reuse)** – бір рет жасалған компонентті басқа жобаларда қолдануға болады.
3. **Инкапсуляция** – әрбір компонент өз деректерін ішкі түрде сақтайды және сыртқы интерфейстер арқылы ғана қол жеткізуге мүмкіндік береді.
4. **Тәуелсіздік** – бір компонентті өзгерткенде, басқа компоненттерге минималды әсер етуі қажет.
5. **Қарапайым интеграция** – компоненттер стандартты интерфейстер арқылы байланысуы тиіс.

**1.3. Компоненттік әдістің артықшылықтары**

✅ **Тиімді қайта пайдалану:** Бір компонентті бірнеше жобада қолдану арқылы әзірлеу уақытын қысқарту.  
✅ **Оңай кеңейту және өзгерту:** Жүйеге жаңа мүмкіндіктер қосу немесе бар компоненттерді өзгерту жеңілдейді.  
✅ **Қателерді азайту:** Тестілеуден өткен және бұрын қолданылған компоненттер сенімдірек болады.  
✅ **Техникалық қолдаудың қарапайымдылығы:** Компоненттік құрылым бағдарламаны оңай жөндеуге және жаңартуға мүмкіндік береді.  
✅ **Өндірістік процестерді оңтайландыру:** Команда мүшелері әртүрлі компоненттермен бір уақытта жұмыс істей алады.

**1.4. Компоненттік әдісті қолдану салалары**

🔹 **Үлкен бағдарламалық жүйелерді әзірлеу** – ERP, CRM, банк жүйелері, өндірістік басқару жүйелері.  
🔹 **Веб-қосымшалар** – REST API, MVC архитектурасы.  
🔹 **Кірістірілген жүйелер** – IoT құрылғылары, микроконтроллерлер.  
🔹 **Жасанды интеллект және машиналық оқыту** – модульдік нейрондық желілер, деректерді талдау жүйелері.

**2. CASE (Computer-Aided Software Engineering) технологиялары**

**2.1. CASE технологиясының мәні**

CASE (Компьютерлік қолдаумен бағдарламалық жасақтама жасау) – бұл бағдарламалық қамтамасыз етуді автоматтандырылған түрде әзірлеу, тестілеу және басқару үшін қолданылатын әдістер мен құралдар жиынтығы.

CASE құралдары бағдарламалық жасақтама өмірлік циклінің әртүрлі кезеңдерінде қолданылады:

* **Жобалау** – диаграммалар, UML модельдері.
* **Код генерациясы** – автоматтандырылған код жазу.
* **Тестілеу** – бағдарламаның функционалдығын тексеру.
* **Құжаттандыру** – техникалық құжаттар мен API сипаттамалары.

**2.2. CASE құралдарының түрлері**

1. **Жоғарғы деңгейлі CASE құралдары** – жүйені жобалау және талдау үшін қолданылады (мысалы, UML, BPMN).
2. **Орта деңгейлі CASE құралдары** – бағдарламалау және код генерациясына бағытталған (мысалы, Rational Rose, Enterprise Architect).
3. **Төмен деңгейлі CASE құралдары** – кодты тестілеу және дебаг жасау үшін (мысалы, Selenium, JUnit).

**2.3. CASE технологиясының артықшылықтары**

✅ Даму жылдамдығын арттырады.  
✅ Қателерді азайтады.  
✅ Командадағы үйлесімділікті жақсартады.  
✅ Автоматтандыру арқылы ресурстарды үнемдейді.

**3.1. CASE операторының мәні**

CASE операторы – бұл бағдарламалау тілдерінде шартқа байланысты бірнеше нұсқаның бірін таңдауға мүмкіндік беретін басқару операторы. Ол **if-else** конструкциясына ұқсас, бірақ кодтың құрылымын жақсартады және оқылымдылығын арттырады.

**3.2. CASE операторының синтаксисі**

Көптеген бағдарламалау тілдерінде CASE операторы әртүрлі түрде жазылады.

**C тілінде (switch-case):**

c

switch(x) {

case 1:

printf("Бірінші нұсқа\n");

break;

case 2:

printf("Екінші нұсқа\n");

break;

default:

printf("Белгісіз мән\n");

}

**Python тілінде (match-case, Python 3.10+):**

python

match x:

case 1:

print("Бірінші нұсқа")

case 2:

print("Екінші нұсқа")

case \_:

print("Белгісіз мән")

**3.3. CASE операторының артықшылықтары**

✅ Кодтың оқылымдылығын арттырады.  
✅ Бірнеше шартты өңдеуді жеңілдетеді.  
✅ Бағдарламаның орындалу жылдамдығын жақсартады.

**3.4. CASE операторының кемшіліктері**

❌ Тек дискретті мәндермен (сан, символ) жұмыс істей алады.  
❌ Барлық бағдарламалау тілдерінде бірдей қолжетімді емес.