Лабораторийн ажил 7: Массив, давталт, жагсаалт, буулгалт

Лабораторийн ажлын зорилго:

Массив ашиглах, давталт ашиглан массив боловсруулах, жагсаалт, буулгалт ашиглах

Лабораторийн ажлын суралцахуйн үр дүнгүүд:

Энэ лабораторийн ажлыг гүйцэтгэснээр оюутан дараах чадваруудтай болсон байна.

д/д	Суралцахуйн үр дүнгүүд	Суралцахуйн үр дүнг илэрхийлэх үйл үг	Суралцахуйн үр дүнгийн түвшин (Блумын)	CLOs хамаарал
1	Массив ашиглах	Ашиглах (Use)	Хэрэглээ	6
2	Давталт ашиглах	Ашиглах (Use)	Хэрэглээ	6
3	Тайлан бичих	Зохион бичих (Compose)	Синтез	12
4	жагсаалт, буулгалт ашиглах	Ашиглах (Use)	Хэрэглээ	6
5	Тайлан хамгаалах	Хамгаалах (defend)	Синтез	12
6	Англи хэл дээр холбогдох материал бусад эх үүсвэрээс унших	Унших (Read)	Ойлголт	13

Ашиглах програм хангамж/техник хангамж, бусад хэрэглэгдэхүүнүүд:

Лабораторийн компьютер эсвэл өөрийн зөөврийн компьютерийг ашиглана.

Онолын ойлголтууд:

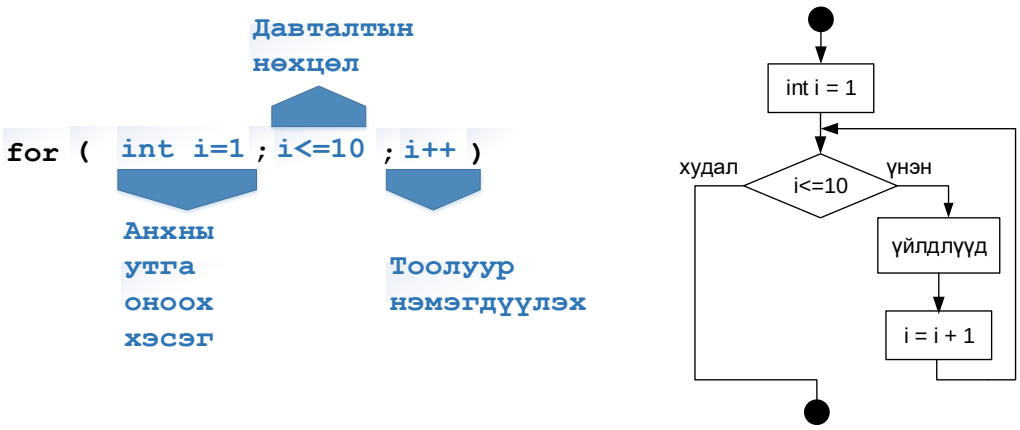
Давталт

Дараах 3 төрлийн давталт програмчлалын хэлэнд байдаг. Үүнд:

1. Параметрт буюу тоолуурт давталт (for)
2. Эхэлж нөхцөлөө шалгадаг давталт (while)
3. Дараа нь нөхцөлөө шалгадаг давталт (do . . . while)

Параметрт буюу тоолуурт давталт

Бичигдэх хэлбэр: Блок схемээр дүрсэлбэл:



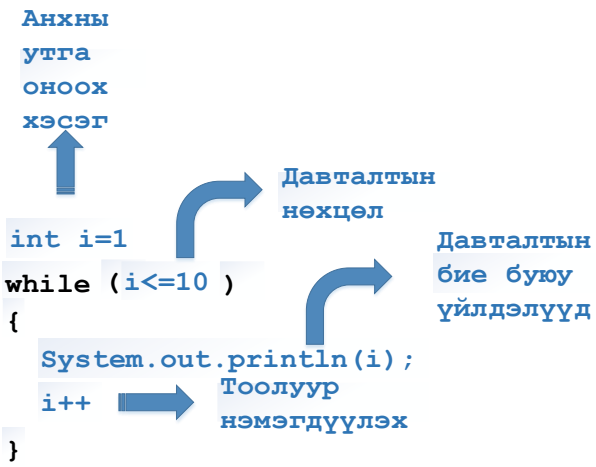
Зураг 7.1 for давталтын блок схем

Жишээ нь:

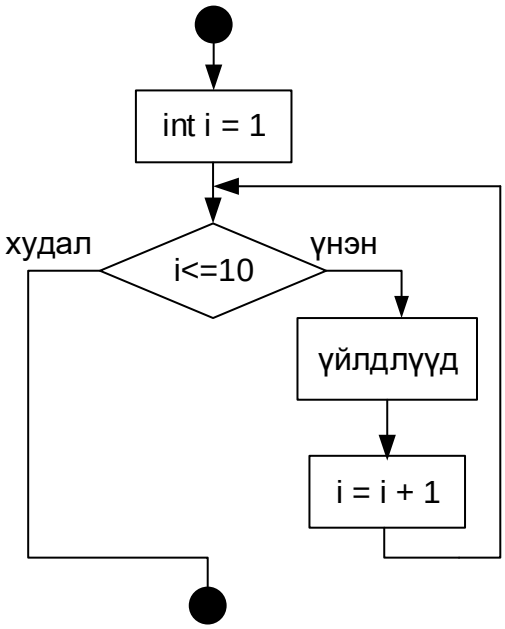
```
for (int i = 1; i<=7; i++)
    System.out.println(i);
for (int i = 1,j = 7; i<=7 && j >=1; i++, j--)
     System.out.print(i + " - " + j + " ");
```

Эхэлж нөхцөлөө шалгадаг давталт

Бичигдэх хэлбэр:



Блок схемээр дүрсэлбэл:



Зураг 7.2. while давталтын блок схем

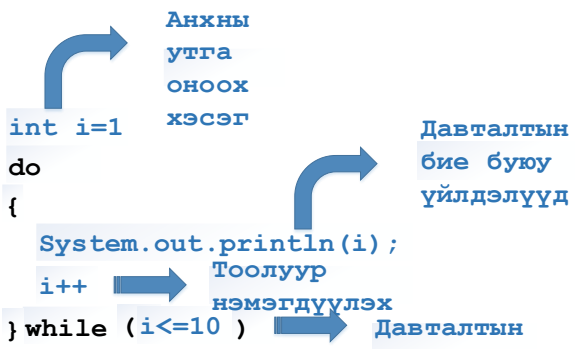
Жишээ нь:

```
int i = 1;
while (i<=7)
{
     System.out.println(i);
```

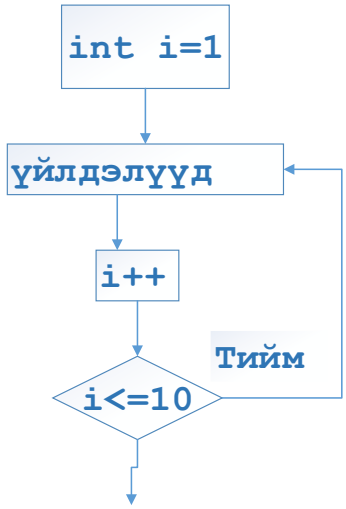
```
        i++;
}
int i = 1,j = 7;
while (i<=7 && j >=1)
{
        System.out.print(i + " - " + j + " ");
        i++;
        j--;
}
```

Дараа нь нөхцөлөө шалгадаг давталт

Бичигдэх хэлбэр:



Блок схемээр дүрсэлбэл:



Зураг 7.3 Дараах нөхцөлт давталтын блок схем

Жишээ нь:

```
int i = 1;
do
{
        System.out.println(i);
        i++;
} while (i<=7);
int i = 1,j = 7;
do
{
        System.out.print(i + " - " + j + " ");
        i++;
        j--;
} while (i<=7 && j >=1);
```

Үргэлжлүүлэх команд

Бичигдэх хэлбэр : **continue**;

Энэ нь do, for, while давталтын командуудтай хамтарч хэрэглэгдэнэ. Өөрөөр хэлбэл энэ команд нь давталтын командуудын блок дотор хэрэглэгдэнэ.

Програм биелж байгаад тухайн давталтын командын блок доторх continue –д удирдлага шилжвэл түүний дараагийн үйлдлүүд биелэхгүй бөгөөд дараагийн цикл шууд эхэлнэ.

Массивын тухай үндсэн ойлголт

Массив бол компьютерын ухаан болон програм хангамжийн хөгжилд чухал байр суурь эзэлдэг ойлголт юм. Үүнийг ашиглан маш олон програмчлалын бодлого бодож, асуудлыг шийдвэрлэсэн олон програмууд зохиогдож байна. Цаашид ч зохиогдсоор байх болно. Програм хангамжийн хөгжлийг дагаад массив гэдэг ойлголт ч улам боловсронгуй болж, хөгжсөөр байна.

Массив нь нэг ижил төрлийн олон утгыг хадгалах зориулалттай хувьсагч юм. Ингэхдээ олон утгуудыг дугаарлаж, дарааллан байрлуулна. Энэ байрлалын дугаарыг *индекс* гэж нэрлэнэ. Массивын бүх элементүүдийн тоог массивын *хэмжээ* гэнэ. Массивын утга нэг бүрт хандахдаа массивын нэр болон массивын дугаарлалт буюу индексээр хандана. ХүснэгтийнНэр[индекс]

Жава хэлэнд массивын эхний элементийн индекс 0-ээс эхэлнэ. Хэрэв массив n элементтэй бол массивын сүүлийн элементийн индекс n-1 байна. Жишээ нь: c[0], c[1], ... c[n-1]. Хэрэв массив 5 элементтэй бол эхний элементийн индекс 0-ээс эхэлж, сүүлийн 5 дахь элементийн индекс 4 байна. Жишээ нь: c[0],c[1],c[2],c[3],c[4]. Энэ тохиолдолд массивын элементийн хязгаараас хэтэрсэн индекст хандахад (c[5] эсвэл c[8] гэхэд) алдаа гарна.

Массивын бүх элементүүд ижил төрөлтэй байхаас гадна массивын элемент бүр тусдаа нэг хувьсагчтай адил бие даасан шинж чанартай байна. Энэ нь массивын нэг элементийн утгыг авах, массивын нэг элементэд утга оноох үйлдлүүд бусдаасаа үл хамаарна. Элементийн утгыг дараах байдлаар авна. Жишээлбэл:

```
int j = c[1];
int I = c[0] + c[1] + 5;
```

Элементийн утга оноох, жишээлбэл:

```
c[0] = 0;
c[1] = 5;
c[2] = c[1] + 5;
c[3] = i;
```

Массивын элемент нь арифметик, логик, харьцуулах зэрэг үйлдлүүдэд нэг хувьсагчийг төлөөлөн оролцож болно. Мөн массивын элементийн индекс нь бүхэл тоо тул бүхэл тоо төрлийн (byte, short, int, long зэрэг) хувьсагч ашиглан массивын элемент руу хандаж болно. Өөрөөр хэлбэл массивын индекс зааж хандахдаа бүх тоо төрлийн хувьсагч ашиглаж болно. Жишээ:

```
int i = 0;
c[i] = 5;
```

Дараах жишээнд массивын эхний 5 элементийг дэлгэцэнд хэвлэж байна.

```
for (int j=0; j< 5; j++)
            System.out.println(c[j]);
```

Массивт хандахдаа бүхэл тоон хувьсагч ашиглаж байхдаа массивын элементийн тоог тооцож, хязгаараас хэтрүүлэхгүй байх ёстой. Дээрх жишээнд хэрэв массив 5-аас цөөн элементтэй бол алдаа гарна.

Массивыг үүсгэх

Си хэлэнд массивыг зарлахдаа хэмжээ зааж өгч үүсгэдэг. Өөрөөр хэлбэл массив зарлах, массив үүсгэх үйлдэл нэг үйлдлээр гүйцэтгэгддэг. Гэтэл объект хандлагат програмчлалын хэлүүдэд

тухайлбал Жава хэлэнд массив үүсгэхээсээ өмнө зарлах шаардлагатай байдаг. Иймээс Жава хэлэнд массив үүсгэх 2 алхам байдаг. Үүнд:

1. Массив зарлах,
2. Массив үүсгэх.

Жава хэлэнд массив зарлахдаа өгөгдлийн төрөл болон массивын нэрийг өгнө. Жава хэлэнд массивыг 2 хэлбэрээр зарлаж болно. Энэ хувьсагч массив гэдгийг илэрхийлсэн [] хаалтыг массивын төрлийн ард бичиж болно. Эсвэл хаалтыг массивын нэрийн ард бичиж болно.

```
төрөл[] массив_нэр;
төрөл  массив_нэр[];
```

Үүнд: төрөл нь массивын төрөл, массив_нэр нь массивын нэр юм.

Жишээ нь:

```
int[] a;
double[] ar1;
double  ar2[];
```

Дээрх жишээнд массивуудын (гурван өөр массив) нэрийг зарласан байна. а нэртэй массивын элэментүүд int төрөлтэй, харин ar1 ба ar2 массивуудын элементүүд double төрөлтэй байна.

Жава хэлэнд массив үүсгэхдээ эхлээд тухайн массиваа зарласан байх шаардлагатай. Мөн дараах хоёр зүйлс байх ёстой. Нэгдүгээрт хэмжээг зааж өгөх ёстой. Хоёрдугаарт new оператор ашиглах ёстой. Жишээ 1:

```
a = new int[10]; // int төрлийн 10 элементтэй массив
```

Жишээ 2:

```
int n = 5;
ar1 =  new double[n]; // double төрлийн 5 элементтэй массив
```

Массивыг үүсгэх үед түүний хэмжээг тодорхойлдог. Ингэснээр массив үүссэний дараа түүний хэмжээ тогтмол байна. Массив үүссэний дараа түүний хэмжээг өөрчлөх, элемент устгаж, нэмэх боломжгүй байдаг. Мөн массив үүссэний дараа массивын элементүүдийн утга өгөгдмөл утгатай байна. Бид өгөгдмөл утгын тухай өмнөх лабораторийн удирдамж дээр хувьсагч зарлах хэсэгт үзсэн. Жишээлбэл int төрлийн хувьсагчийн өгөгдмөл утга 0 байдаг.

Үүсгэсэн массив бүрийн хэмжээг цээжлэх шаардлагагүй. Үүний оронд массивын нэрний ард .length бичихэд тухайн массивын хэмжээ буцна. Жишээ нь:

```
int razmer = mas1.length;
a.length
```

Массивын .length шинж чанарын утгыг өөрчлөх боломжгүй, зөвхөн унших боломжтой байдаг.

Жава хэлэнд массивын зарлалт, үүсгэлтийг тусдаа хийх нь тухайн массивыг дахин ашиглах боломжийг бүрдүүлдэг уян хатан чанартай. Жишээлбэл нэг удаа массив зарлачихаад түүнээс хоёр удаа массив үүсгэх боломжтой. Дараах жишээнд а массивыг нэг удаа зарлаж, хэмжээг хоёр өөрөөр өгч үүсгэсэн бөгөөд үүсгэх бүрт массивын хэмжээг хэвлэсэн байна.

```
int a[];

a = new int[10];
System.out.println(a.length);

a = new int[5];
System.out.println(a.length);
```

Массив зарлалт, үүсгэлт хоёрыг нэг мөрөнд бичиж болно. Дараах хэлбэрээр бичнэ. Үүнд:

```
төрөл[] массив_нэр  = new төрөл[хэмжээ];
```

Жишээ нь:

```
int[] mas2  = new int[3];
```

Элементүүдийн анхны утгатай массив үүсгэх

Массивын элементүүдийн утгыг массив зарлахдаа өгч үүсгэж болно. Үүнийг дараах хэлбэрээр бичнэ. Үүнд: `төрөл[] нэр = {эл0, эл1, …, элN};`

Жишээлбэл

```
double[] ar2 = {3.14, 2.71, 0, -2.5, 99.123}; // double төрлийн 5 элементтэй
```

Дээрх байдлаар массивыг үүсгэхэд массив олгох утгууд тодорхой, тогтсон байх үед ашиглана. Энэ үед массивын элементийн тоог зааж өгөх шаардлагагүй. Мөн new оператор ашиглах шаардлагагүй.

Олон хэмжээст массив

Бид массивыг ганц индекстэй байдлаар үзлээ. Массив нь олон индекстэй буюу олон хэмжээстэй байж болно. Өмнө үзсэн нэг хэмжээст массивыг шугаман дараалал, тоон цуваатай зүйрлэж болох бол хоёр хэмжээст массивыг матриц, хүснэгт, шатрийн хөлөг, хавтгайтай зүйрлэж болно. Харин гурван хэмжээст мастрицийг 3 хэмжээст огторгуйтай зүйрлэж болно. Бид програмчлалд нэг хэмжээст массиваас гадна, хоёр, гурван хэмжээст массивыг өргөн ашигладаг. Харин 4 ба түүнээс дээш хэмжээст массивыг төдийлөн өргөн ашигладаггүй. Магадгүй энэ нь бидний 4 хэмжээст орон зайг төсөөлөн бодох чадвар муутай байдагтай холбоотой юм. Гэвч 4 ба түүнээс олон хэмжээсийг програмчлалд ашиглаж болно. Бид жишээ болгон хоёр хэмжээст массив авав.

Хоёр хэмжээст массивыг зарлахдаа дараах байрлаар зарлана. Жишээ нь:

```
int[][] multi;
```

Хоёр хэмжээст массивыг үүсгэхэд олон үйлдэл бичих ба new операторыг олон ашигладаг. Дараах жишээн дэх хоёр хэмжээст массив үүсгэлтийг тайлбарлавал 5 мөр, 10 баганатай массив байна. Эхний үйлдэл нь 5 мөрийг үүсгэж байна. Дараагийн 5 үйлдэл нь 5 мөр тус бүрийн хувьд 10 баганыг үүсгэж байна. Үүнд:

```
multi = new int[5][];
multi[0] = new int[10];
multi[1] = new int[10];
multi[2] = new int[10];
multi[3] = new int[10];
multi[4] = new int[10];
```

Хоёр хэмжээст массивын элементүүдийн утгыг массив зарлахдаа өгч үүсгэж болно. Жишээ нь:

```
int[][] multi = new int[][]
{ { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};
```

Хоёр хэмжээст массивт хандахдаа дараах байдлаар хандана. Жишээ нь:

```
multi[0][0] = 1;
multi[0][1] = multi[0][0] + 1;
```

Массивтай ажиллахдаа давталт ашиглах нь

Олон элементтэй массивын бүх элементүүдэд дараалан хандахад нэг үйлдлийг олон дахин бичих шаардлага гардаг. Иймээс олон тооны элементтэй массивд for давталт ашиглавал тохиромжтой байдаг. Энэ for давталтад байдаг параметр буюу тоолуурыг массивын индекс заах байдлаар ашигладаг. Жишээ нь:

```
for(int i=0; i<=2; i++)
    mas2[i] = (i+1) * 10;
```

Массивын хэмжээ, индекс бүхэл тоо байдаг тул тоолуур нь бүхэл тоон тоолуур байна. Массивын индекс тэгээс эхэлдэг тул давталтын тоолуурын анхны утга 0 байна. Массивын хэмжээс n байвал давталтын нөхцөл тоолуурыг n – 1 хүртэл давтах ёстой. Массивын бүх элемент рүү хандахын тулд тоолуурыг нэгээр нэмэгдүүлнэ. Дээрх жишээний хувьд mas2 массив нь 3 элементтэй байна. Иймээс давталтын нөхцөл i<=2 буюу тоолуурыг 2 хүртэл давтана. Үүнээс гадна давталтын нөхцөл нь i < 3 эсвэл i < mas2.length байж болно. Дараах жишээнд .length ашиглан тухайн массивын хэмжээг мэдэхгүйгээр дэлгэцэнд ar2 нэртэй массивын элементүүд хэвлэх байдлыг үзүүлэв. Үүнд:

```
for(int i = 0; i < ar2.length ; i++)
   System.out.print(ar2[i] + "  ");
```

Дээрх жишээнүүдэд массивын эхний элементээс эхлэн төгсгөлийн элемент хүртэл дарааллаар хэвлэсэн байна. Үүний эсрэгээр массивын төгсгөлийн элементээс эхний элемент хүртэл хэвлэж болно. Жишээ нь:

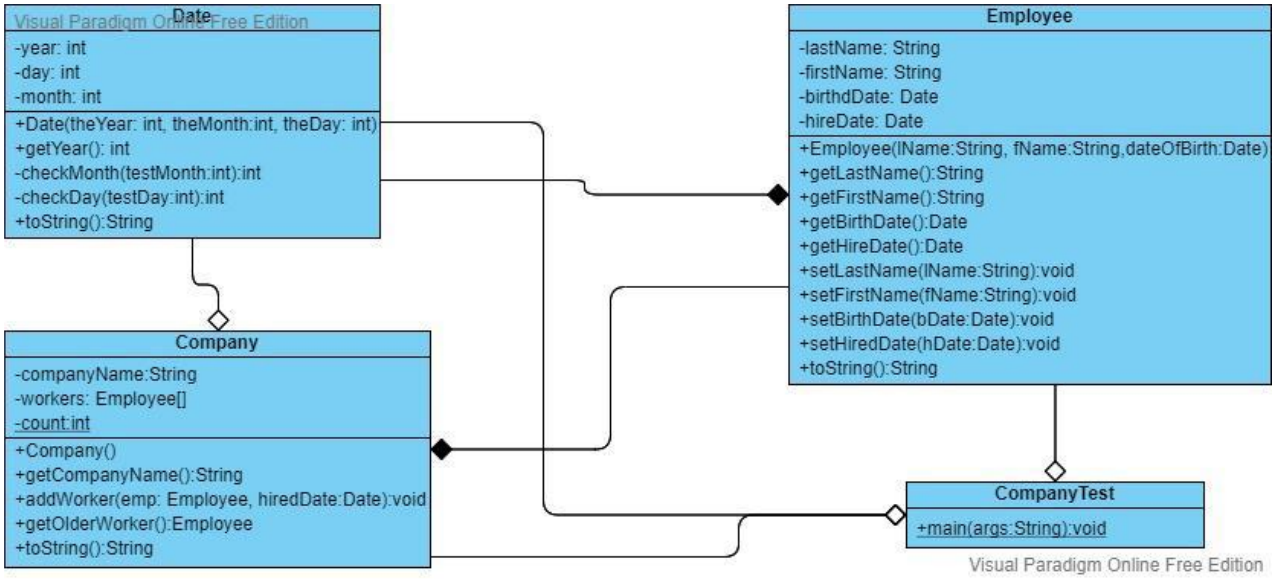
```
for(int i = ar2.length - 1; i >=0; i--)
   System.out.print(ar2[i] + "  ");
```

Энэ үед давталтын тоолуурын анхны утга массивын хэмжээнээс нэгийг хассантай тэнцүү байна. Мөн давталтын нөхцөл тоолуур тэгээс их буюу тэнцүү байна. Харин тоолуур алхам бүрт нэгээр багасна.

Массивыг боловсруулахдаа бодлогын даалгавар, алгоритмаас хамааран for давталт ашиглахаас гадна while, do while давталтуудыг ашиглаж болно.

Массивт объект хадгалах

Жишээ 3: Огноо (Date), Ажилтан *(Employee),* компани *(Company)* классыг тодорхойлох



Зураг 7.4 Жишээ програмын класс диаграм

Date класс

```
public class Date
{
    private int year; // ямар нэгэн он
    private int month; // 1-12 хүртэлх сар
    private int day; // сараас хамаарч 1-31 хүртэлх утга хадгална
    /*
    * Байгуулагч функц дотор сар ба өдрийг шалгах
    * checkMonth, checkDay аргыг дуудах
    */
    public Date(int theYear, int theMonth, int theDay){
        year = theYear;
        month = checkMonth(theMonth); // Сарыг баталгаажуулах
        day = checkDay(theDay); // Өдрийг баталгаажуулах
    }
    public int getYear(){
        return year;
    }
/*
* Тухайн тоо сар эсэхийг шалгах арга
*/
    private int checkMonth(int testMonth){
        if(testMonth > 0 && testMonth <= 12) {
            return testMonth;
```

```
        } else {
            return -1;
        }
    }
    /*
    * Сар ба жилд үндэслэн зөв өдөр эсэхийг шалгах арга
    */
    private int checkDay(int testDay){
        // 12 сард ногдох өдрүүд
        int daysPerMonth[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31};
        // Өдөр зөв эсэхийг шалгах
        if(testDay > 0 && testDay <= daysPerMonth[month]){
            return testDay;
        }
        // Өндөр жил эсэхийг шалгах
        if(month == 2 && testDay == 29 && (year % 400 == 0 || (year % 4 == 0 &&
            year % 100 != 0))){
            return testDay;
        }
        // Зөв өдөр биш бол
        return -1;
    }
    /*
    * Он/Сар/Өдөр форматаар тэмдэгт мөр буцаах арга
    */
    public String toString(){
        return String.format("%d/%d/%d", year, month, day);
    }
}
```

- **`Date`** класс нь **`year`** (жил), month (сар), day (өдөр) **`гэсэн 3 гишүүн өгөгдөлтэй.`**
- Байгуулагч функц нь **`int`** төрлийн 3 параметрээр хангагдана.
    - **`year`** гишүүн өгөгдөлд **`int`** төрлийн хамгийн эхний параметрыг шууд олгоно.
    - **`month`** гишүүн өгөгдөлд **`Date`** класст **`private`** харагдалтаар тодорхойлсон **`checkMonth()`** нэртэй функцийг дуудаж ажиллуулна. Энэхүү функц нь сар гишүүн өгөгдөл зөв утга авсан эсэхийг шалгаад ямар нэг буруу утга авбал -1 гэсэн утгыг оноох үүрэгтэй.

- **day** гишүүн өгөгдөлд **checkDay()** функцийг дуудаж ажиллуулна. Энэ функц нь жил ба сарын мэдээлэлд үндэслэн өдөр гишүүн өгөгдөл зөв утга авсан эсэхийг шалгах үүрэгтэй.

• **getYear()** асуух функц нь **Date** классын **private** харагдалт бүхий **year** гишүүн өгөгдлийн утгыг буцаана.
• **toString()** функц нь **Date** классын **String** дүрслэлийг заана.

Ажилтан класс

```
public class Employee
{
    private String lastName; // Ажилтны овог
    private String firstName; // Ажилтны нэр
    private Date birthDate; // Ажилтны төрсөн огноо
    private Date hireDate; // Ажилд орсон огноо
    /*
    * Ажилтны нэр, овог, төрсөн огноо, ажилд орсон огноог олгох байгуулагч
функц
    */
    public Employee(String lName, String fName , Date dateOfBirth){
        lastName = lName;
        firstName = fName;
        birthDate = dateOfBirth;
    }
    public String getLastName(){
        return lastName;
    }
    public void setLastName(String lName){
        lastName = lName;
    }
    public String getFirstName(){
        return firstName;
    }
    public void setFirstName(String fName){
        lastName = fName;
    }
    public Date getBirthDate(){
        return birthDate;
    }
    public void setBirthDate(Date bDate){
```

```
        birthDate = bDate;
    }
    public Date getHireDate(){
        return hireDate;
    }
    public void setHireDate(Date hDate){
        hireDate = hDate;
    }
    /*
    * Employee классаас үүсэх объектын мэдээллийг тэмдэгт мөр рүү хөрвүүлэх
    */
    public String toString(){
        return String.format("%s овогтой %s", lastName, firstName);
    }
}
```

- **Employee** класс нь **lastName**, **firstName**, **birthDate**, **hireDate** гэсэн гишүүн өгөгдөлтэй. Үүнээс **birthDate**, **hireDate** нь **Date** класс төрөлтэй объект байх юм. Уг класс нь “Ажилтан бол төрсөн огноотой”, “Ажилтан бол ажилд орсон огноотой” нөхцөлийг хангах тул бүрдмэл харьцааны шинж чанарыг илэрхийлж чадах юм.
    - Байгуулагч функц нь гурван параметрээр хангагдах бөгөөд овог, нэр, төрсөн өдрийг харгалзах гишүүдэд олгоно.
    - **Employee** классын **private** харагдалттай гишүүн өгөгдлүүдэд зориулан асуух болон өөрчлөх үйлдлүүдийг тодорхойлсон.

• Мөн классын **String** дүрслэлийг **toString()** үйлдэлд тодорхойлсон байна

Company класс

```
public class Company {
    private String companyName; // Компаний нэр
    private Employee[] workers; // Компанийн ажилтнуудыг хадгалах массив
    // Массивд Employee классаас үүссэн объектуудыг хадгална.
    static int count = 0; // Ажилтны тоог хадгалах статик гишүүн өгөгдөл
    /*
    * Компаний нэр, ажилтны тоог оролтоор хүлээн авна.
    * Ажилтнуудыг хадгалах массивыг үүсгэнэ.
    */
    public Company(String name, int workersNum){
        companyName = name;
        // Ажилтнуудыг хадгалах массивыг үүсгэх
        workers = new Employee[workersNum];
    }
    public String getCompanyName(){
        return companyName;
    }
    /*
    * Компанид ажилтан нэмэх гишүүн үйлдэл
    * Компанид нэмэх ажилтан ба ажилд орсон огноог параметрээр авна.
```

```
    */
    public void addWorker(Employee emp, Date hiredDate){
        // Ажилтны ажилд орсон огноог тохируулах
        emp.setHireDate(hiredDate);
        // Ажилтныг массивд нэмээд count -н утгыг нэгээр нэмэгдүүлнэ.
        workers[count++] = emp;
    }
    /*
    * Компанид хамгийн олон жил ажилласан ажилтныг олох функц
    */
    public Employee getOlderWorker(){
        // Хамгийн олон жил ажилласан ажилтныг хадгалах локал хувьсагч
        // Ажилтнуудыг хадгалж буй массивын хамгийн эхний ажилтныг хадгалах
        Employee olderWorker = workers[0];
        // Хамгийн олон жил ажилласан ажилтны ажилд орсон огноог хадгалах локал
хувьсагч
        // Ажилтнуудыг хадгалж буй массивын хамгийн эхний ажилтны ажилд орсон
огноог хадгалах
        int xYear = workers[0].getHireDate().getYear();
        // Бүх ажилтныг шалгах
        for(int i =1; i < count; i++){
            // Дараагийн ажилтны ажилд орсон огноог хадгалах локал хувьсагч
            int yYear = workers[i].getHireDate().getYear();
            // Дараагийн ажилтны ажилд орсон огноо бага байх
            if(xYear > yYear){
                xYear = yYear;
                olderWorker = workers[i];
            }
        }
        return olderWorker;
    }
    // Компаний бүх ажилтны мэдээллийг нэгтгэж буцаах
    public String toString(){
        String allInfo = "";
        for(int i =0; i < count; i++){
            allInfo = allInfo + workers[i].toString() + "\t" + "Ажилд орсон
огноо: " + workers[i].getHireDate() + "\n";
        }
        return allInfo;
    }
}
```

- **`Company`** класс нь нэр, ажилтнууд, ажилтны тоо гэсэн гишүүн өгөгдлийг агуулна. Энд **`workers`** нь **`Employee`** класс төрөлтэй объектын олонлог байх тул массиваар зарласан байна. Энэ нь “Компани бол ажилтантай” гэх нөхцөлийг хангана. Харин **`count`** нь компанийн нийт ажилтны тоог илэрхийлэх бөгөөд анхны утга **`0`** байна. Мөн статик тул классын хүрээнд ашиглагдана. Статик гишүүн өгөгдөл нь тухайн классаас үүссэн объектуудын дунд ганц утгыг хуваалцана. Өөрөөр хэлбэл, классын гишүүн өгөгдлүүд нь объект бүрийн хувьд байдаг. Харин статик гишүүн өгөгдөл нь классын бүх объектуудын дунд ганц байдаг. Статик гишүүн өгөгдлийг объект бус класс дээр дууддаг. Жишээлбэл, **`Company.count;`**
- Байгуулагч нь нэр ба байх боломжтой ажилтны тоог авч, их бие дотор ажилтнуудыг хадгалах массивыг үүсгэнэ.

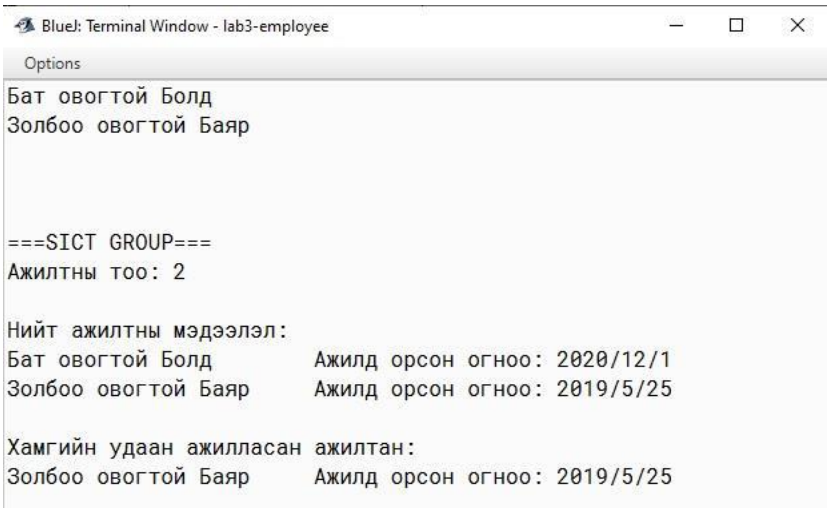
- **addWorker()** функц нь компанид шинэ ажилтан нэмэх үүрэгтэй. Тиймээс оролтын утгаар Employee төрөлтэй emp объект, emp объектын ажилд орох огноог авна.
- **getOlderWorker()** функц нь компанийн ажилтнуудаас хамгийн олон жил ажилласан ажилтныг олох үүрэгтэй.
- **toString()** нь компанийн бүх ажилтны мэдээллийг нэгтгэж буцаах үүрэгтэй дүрслэгджээ.

Ажиллуулагч класс

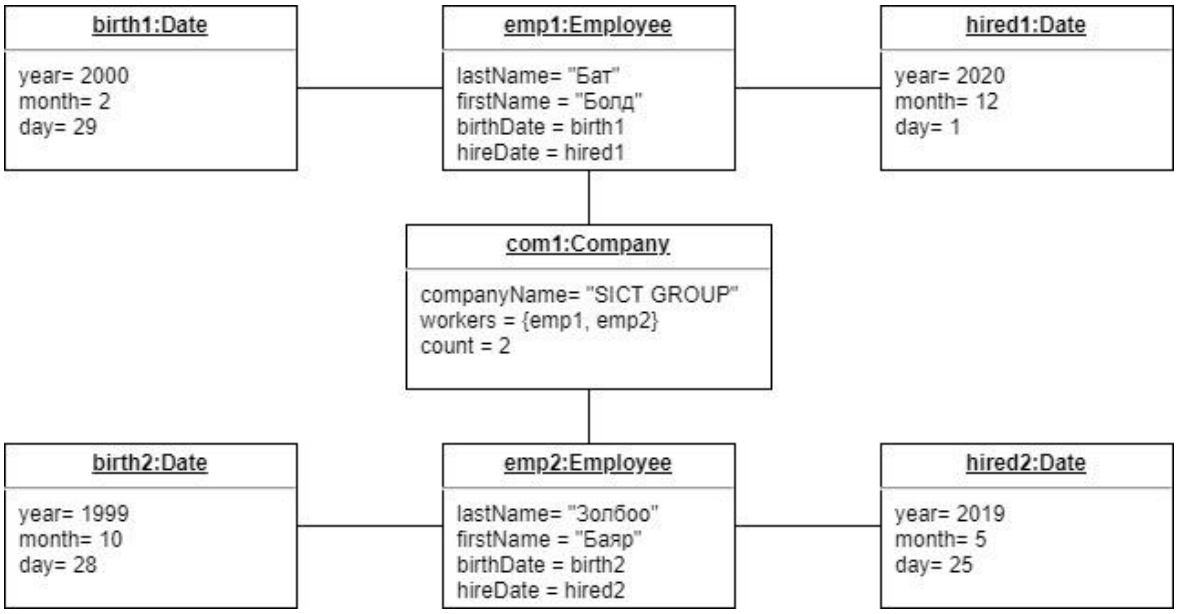
```
public class CompanyTest
{
    public static void main(String args[]){
    // Эхний ажилтны төрсөн огноог үүсгэх
    Date birth1 = new Date(2000, 2, 29);
    // Эхний ажилтанг үүсгэх
    Employee emp1 = new Employee("Бат", "Болд", birth1);
    // Эхний ажилтны мэдээллийг хэвлэх
    System.out.println(emp1);
    // Хоёр дахь ажилтны төрсөн огноог үүсгэх
    Date birth2 = new Date(1999, 10, 28);
    // Хоёр дахь ажилтанг үүсгэх
    Employee emp2 = new Employee("Золбоо", "Баяр", birth2);
    // Хоёр дахь ажилтны мэдээллийг хэвлэх
    System.out.println(emp2);
    // Компани үүсгэх
    Company com1 = new Company("SICT GROUP", 10);
    // Эхний ажилтны ажилд орсон огноог үүсгэх
    Date hired1 = new Date(2020, 12, 1);
    // Хоёр дахь ажилтны ажилд орсон огноог үүсгэх
    Date hired2 = new Date(2019, 5, 25);
    // Эхний ажилтанг компанид нэмэх
    com1.addWorker(emp1, hired1);
    // Хоёр дахь ажилтанг компанид нэмэх
    com1.addWorker(emp2, hired2);
    // Компаний тухай мэдээлэл
    System.out.println("\n\n");
    System.out.println("===" + com1.getCompanyName() + "===");
    System.out.print("Ажилтны тоо: ");
    System.out.println(Company.count);
    System.out.println("\nНийт ажилтны мэдээлэл: ");
```

```
    System.out.println(com1.toString());
    System.out.println("Хамгийн удаан ажилласан ажилтан: ");
    Employee olderWorker = com1.getOlderWorker();
    System.out.println(olderWorker + " \tАжилд орсон огноо: " +
    olderWorker.getHireDate());
    }
}
```

- **`CompanyTest`** класс нь ажиллуулагч үүргийг хүлээх класс бөгөөд **`main()`** функц дотор хоёр ажилтан үүсгэж, компанид нэмэх үйлдлийг биелүүлж байна. Эцэст нь компанийн тухай мэдээллийг терминал руу хэвлэжээ.



Зураг 7.5 Жишээ кодын үр дүн



Зураг 7.6 Жишээ кодын объект диаграм

Жагсаалт/List

Шугаман дараалал хэлбэрээр элементүүдийг хадгалахад зориулагдсан бүтцийг жагсаалт гэнэ. Жагсаалт нь хэдэн ч элементтэй байж болох бөгөөд объект нэмэх, устгах, хандах боломжтой.

Жишээ нь:

```
import java.util.*;
List<Person> friends;   //Person төрөлтэй friends жагсаалт зарлаж байна
Person person;                  //Person классын объект зарлагаа
friends = new ArrayList<Person>( ); // friends жагсаалтаа үүсгэнэ
person = new Person("jane", 10, 'F');     //person объектоо үүсгэнэ
friends.add( person );          //үүсгэсэн объектоо жагсаалтдаа нэмж байна
person = new Person("jack", 6, 'M');      //дахин нэг person объект үүсгэж
                                          байна
friends.add( person );          //үүсгэсэн объектоо жагсаалтдаа нэмж байна
Person p = friends.get( 1 );  //жагсаалтаас 2 дахь объектыг буцааж байна
```

Буулгалт/Map

Түлхүүр(key) ба утга(value) гэсэн хослолоор элементийг хадгалдаг бүтцийг буулгалт гэнэ.

Жишээ нь:

```
import java.util.*;
Map catalog;              //Map зарлагаа
// TreeMap төрөлтэй map үүсгэе
catalog = new TreeMap<String, String>( );
//map руу элементүүд нэмэх
catalog.put("CS101", "Intro Java Programming");
catalog.put("CS301", "Database Design");
catalog.put("CS413", "Software Design for Mobile Devices");
//өгөгдсөн утга бүхий түлхүүр байгаа эсэхийг шалгах
if (catalog.containsKey("CS101")) {
      System.out.println("We teach Java this semester");
} else {
      System.out.println("No Java courses this semester");
}
```

Ажил гүйцэтгэх дараалал:

Өмнөх лабораториуд дээр тодорхойлсон классуудыг өргөтгөж объект массив агуулсан бүрдмэл классууд үүсгэнэ. Ингэхдээ:

1) Эхлээд нэмэх классуудыг дэвтэр дээрээ төлөвлөж, тэмдэглэх;
   a. Үгээр бичих эсвэл ноорог зураг гаргаж болно.
2) Бүрдмэл харьцааг илэрхийлж чадах классуудыг шинээр нэмж тодорхойлох;
   a. Энэ үед “... бол ...-тай4” хэллэгийг анхаарах;
3) Объект массив ашиглах, элементүүдэд хандах;
   a. Жишээ нь: **`Company`** классын **`workers[]`** гишүүн өгөгдөл шиг;
4) Объект массивыг жагсаалт болгон өөрчилж турших
   a. Жишээ нь: **`Company`** классын **`workers[]`** массивыг жагсаалт болгох
5) Буулгалт ашиглаж турших

а. Department(id,name) бүтэц бүхий буулгалт/map үүсгэж workers-н ажилладаг салбарыг оноож өгөх

6) Лабораторийн ажлын хүрээнд тодорхойлсон классуудыг агуулах класс диаграмыг зурна. Мөн ажиллуулагч классын main функц доторх объектуудаар объект диаграм зурна.

а. Диаграмыг дурын хэрэгсэл ашиглан зурж болох ч *draw.io* вебсайтыг ашиглавал илүү хялбар байх болно.

Ашиглах материал:

- Ю.Намсрай, Т.Гантөр, Д.Ганцоож, Програмчлалын Жава хэл, 2015 он.
- Д.Энхжаргал, Java 2 Объект хандлагат програмчлал, 2011 он.