ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ

F.CS202
ОБЪЕКТ ХАНДЛАГАТ ПРОГРАМЧЛАЛ

Лекц №4

Битүүмжлэл ба классын бүрэлдэхүүн

док., дэд проф. Б.Батзолбоо
маг. Б.Мөнхбуян

2021 он

Агуулга

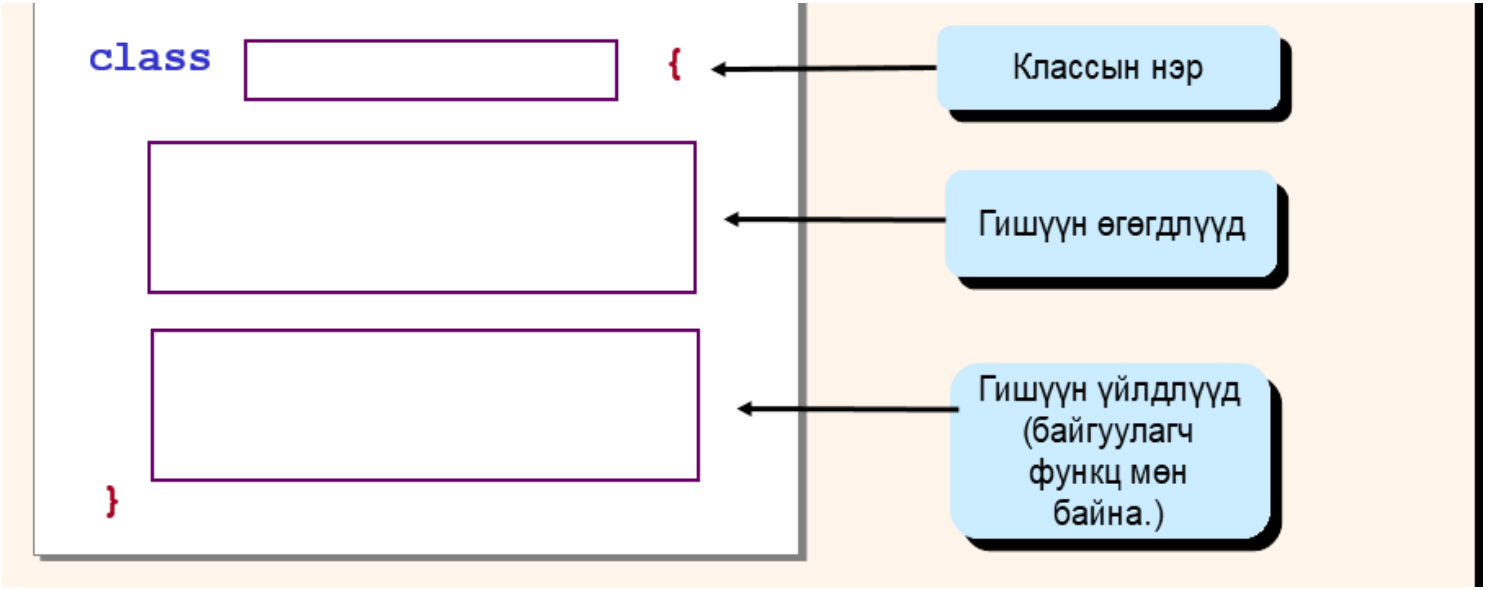
- Классын бүрэлдэхүүн
  - Байгуулагч функц, байгуулагчийн олон хэлбэржилт
  - Устгагч функц
  - Функц, түүний дуудалт
  - Функцийн параметр рүү аргумент дамжуулах
- Классыг пакет болгон зохион байгуулах
- BlueJ дээр хэрэгжүүлэх

Өмнөх лекцээр

- Классын гишүүн өгөгдөл, гишүүн үйлдэл(метод)
- Асуух, өөрчлөх үйлдэл
- Харагдалт
  - public, private
- Үйлдлийн олон хэлбэржилт

Классын бүрэлдэхүүн хэсгүүд

- Өгөгдөл/атрибут – гишүүн, атрибут
- Үйлдэл/метод - харьцаа
- Байгуулагч функц
- Устгагч функц



Байгуулагч функц

- *Байгуулагч* гэдэг нь классаас шинэ объект/инстанс үүсгэх үед ажилладаг онцгой функц юм.

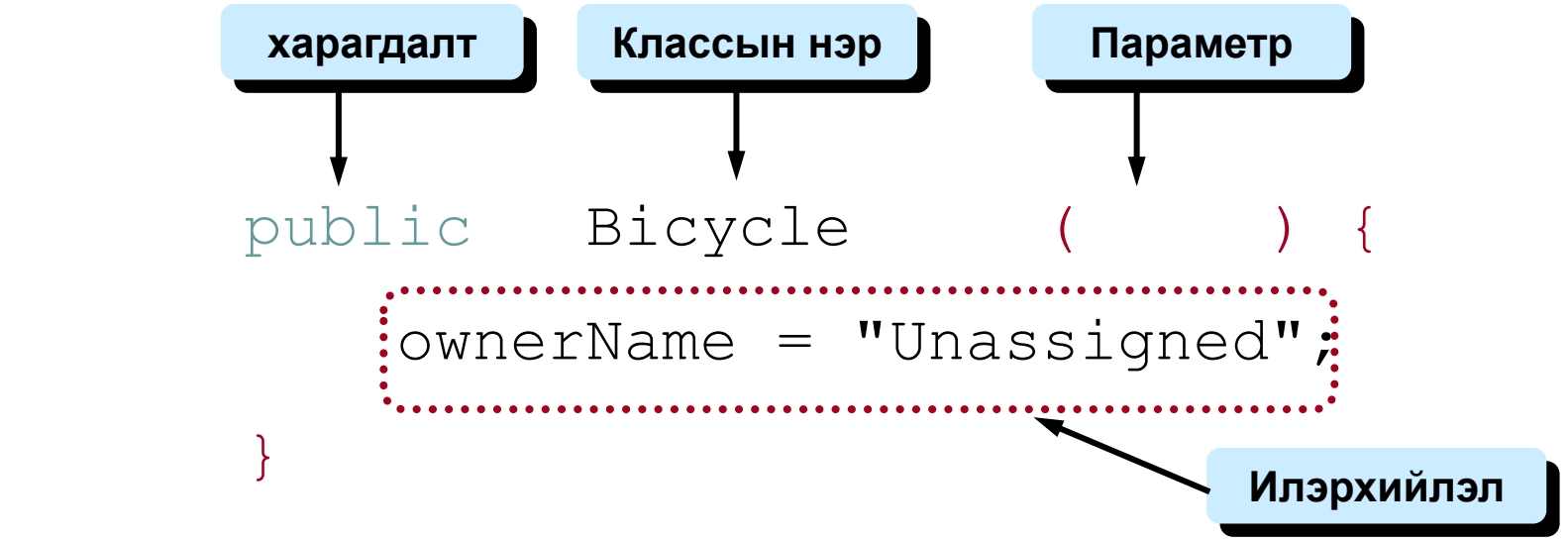
- Байгуулагч функцийн үндсэн зорилго нь объектыг үүсгэх үедээ классын гишүүдэд анхны утга олгох юм. Ингэхдээ байгуулагч функцийн параметрүүдийг далд гишүүдийн анхны утга олгоход ашигладаг.

- Байгуулагч функцийн нэр нь классын нэртэй ижил байна.

- Буцаах утга байхгүй тул буцаах утгын төрөл эсвэл void ашиглахгүй.

Байгуулагч функц

```
public <class name> ( <parameters> ){
    <statements>
}
```



Байгуулагч функц

- Байгуулагчийн төрлүүд:
  - Өгөгдмөл/default байгуулагч
  - Аргументгүй/no-args байгуулагч
  - Параметртэй байгуулагч

Өгөгдмөл байгуулагч

```
public class Student {
public static void main(String[] args) {
  Student s = new Student();
}
}
```


```
public class Student {
Student() {}
public static void main(String[] args)
{
  Student s = new Student();
}
}
```

- Хэрэв классд ямар нэг байгуулагч функц тодорхойлоогүй бол Жава компайляр нь ажиллахдаа кодонд өгөгдмөл байгуулагчийг нэмэх ба энэ нь кодонд харагдахгүй боловч .class файлд нэмэгдсэн байдаг.
- Ямар нэг байгуулагч тодорхойлсон л бол өгөгдмөл байгуулагч ашиглагдахгүй болно.

Аргументгүй байгуулагч

```
class Student
{
      public Student()
      {
           System.out.println("Student классын байгуулагч");
      }
      public static void main(String args[]) {
         new Student();
      }
}
```

Үр дүн:

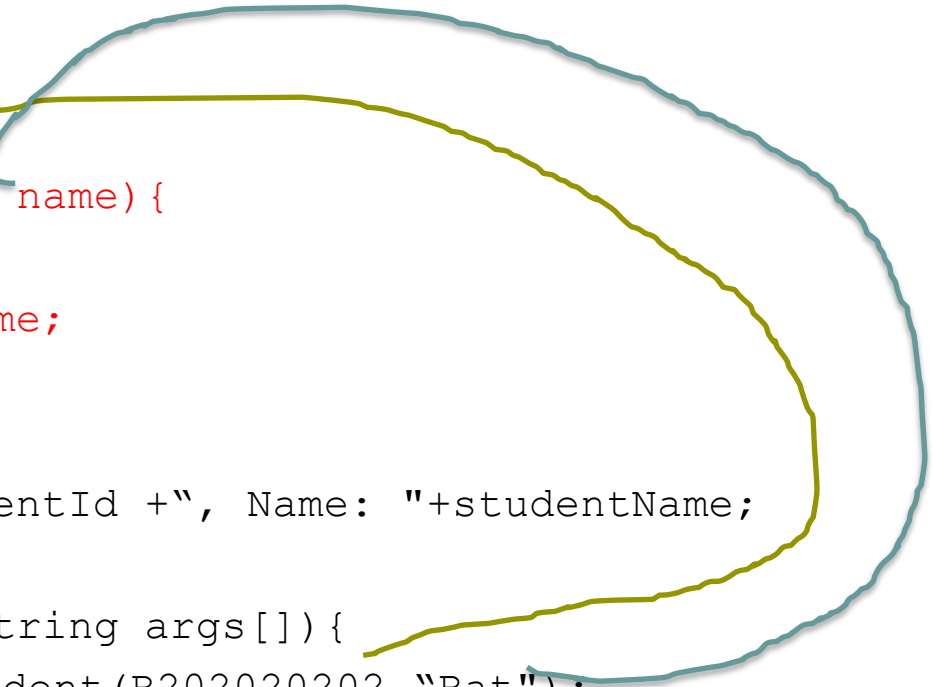
Student классын байгуулагч

- Аргументгүй тул өгөгдмөл байгуулагчтай адил мэт боловч ялгаа нь байгуулагчийн body хэсэгт код бичиж болно.

Параметртэй байгуулагч

```
public class Student {
    String studentId;
    String studentName;

    Student(String id, String name){
        this.studentId = id;
        this.studentName = name;
    }
    String printInfo(){
        return  “Id: “ + studentId +“, Name: "+studentName;
    }
    public static void main(String args[]){
         Student st1 = new Student(B202020202,“Bat");
         Student st2 = new Student(B202020209,“Tergel");
         System.out.println(st1.printInfo());
         System.out.println(st2.printInfo());
    }
}
```



Үр дүн:

```
Id: B202020202, Name: Bat
Id: B202020209, Name: Tergel
```

Методын олон хэлбэржилт

- Дараах тохиолдлуудад методууд ижил нэртэй байж болно
  - Параметрийн жагсаалтаараа ялгаатай (Дүрэм 1)
  - Параметрийн тоо адил боловч өгөгдлийн төрөл нь ялгаатай (Дүрэм 2)

```
public void myMethod(int x, int y) { ... }
public void myMethod(int x) { ... }
```

✓ Дүрэм 1

```
public void myMethod(double x) { ... }
public void myMethod(int x) { ... }
```

✓ Дүрэм 2

Байгуулагчийн олон хэлбэржилт

- Ижил дүрэмтэй
  - Класст нэгээс олон байгуулагч тодорхойлж болно

```
public Person( ) { ... }
public Person(int age) { ... }
```

✓ Дүрэм 1

```
public Pet(int age) { ... }
public Pet(String name) { ... }
```

✓ Дүрэм 2

Байгуулагч ба this

- Классын нэг байгуулагч дотроос нөгөө байгуулагчийг дуудахдаа this үгийг ашиглана.

```
public class Student {
    String studentId;
    String studentName;

    Student(String id) {
        studentId = id;

    }

    Student() {
        studentName = “Unknown”;
    }

    Student(String id, String name){
        this(id);
        this.studentName = name;
    }
}
```

Устгагч функц

- Объектыг санах ойгоос устгана.
- Ямар зорилгоор ашиглах вэ ?
  - Нөөцүүдийг чөлөөлөх,
  - Санах ойд ачаалсан объектын өгөгдлийг устгаж, санах ойг чөлөөлөх,
  - Сүлжээний холболт, өгөгдлийн сангийн холболтыг таслах,
  - Санах ойд байгаа объектын өгөгдлийг өгөгдлийн санд хадгалан, өгөгдлийн сангийн холболтыг таслах,
  - Санах ойд байгаа объектын өгөгдлийг файлд хадгалан, файлыг хаах.
- Нэг классад нэг устгагч функц байж болдог.

Устгагч функц

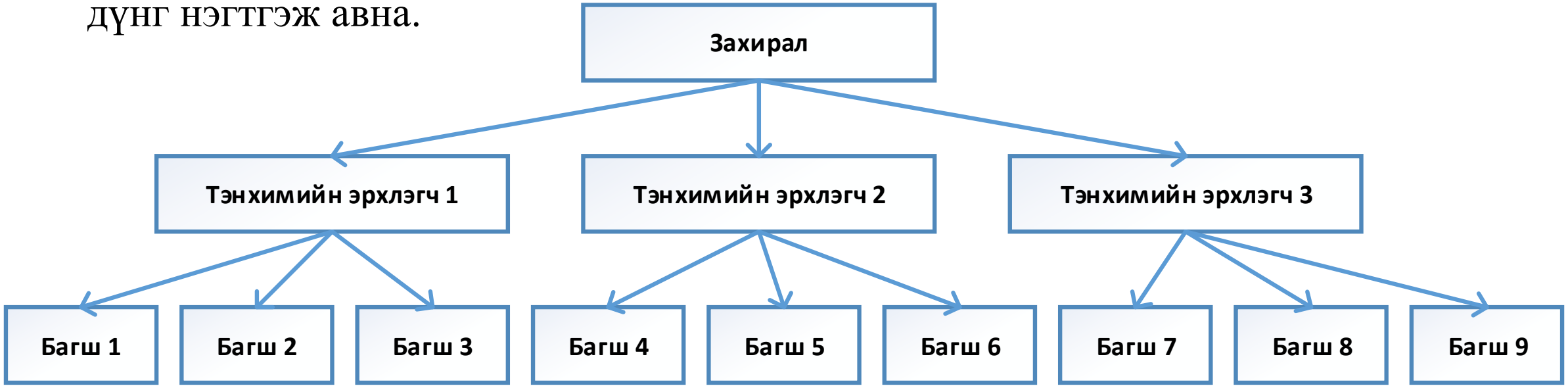
- Объект хандлагат технологид объектыг хоёр замаар устгадаг.
  - Ил - ө.х тусгай үйлдэл эсвэл оператор дуудаж (C++)
  - Далд - объект програмд хэрэггүй болох үед устгагдах (Java,C#)
- Ил үед
  - Ашиглаж байгаа объектыг устгахын тулд заавал устгагч функц дуудна.
  - Объектын устгалт хийгээгүйгээс санах ойн дутагдал үүсэж болно.
- Далд үед
  - Garbage collector (хог цуглуулагч) нь хэрэггүй болсон объектуудыг Java хэлний орчин автоматаар устгадаг.
  - Хог цуглуулагч ажиллаж эхэлсэн үед програмын ажиллагааны хувьд нэмэлт ачаалал өгдөг.

Объект устгалт Java хэлэнд

- Java хэлэнд объект устгалт хог цуглуулагчаар дамжин далд гүйцэтгэгдэнэ.
  - System.gc(); гэж ил байдлаар хог цуглуулагчийг идэвхижүүлж болно.
- Хог цуглуулагч Finalize гэх объектын тусгай үйлдлийг объектын устгахын өмнө дууддаг.
  - Энэ үйлдлийн өөрийн классдаа тодорхойлж болно.
  - Энэ үйлдэл нь аргумент болон буцаах утгагүй байдаг.
- Java хэлэнд автоматаар хийгддэг тул устгагч функц тодорхойлохгүй, хог цуглуулагчийг дуудахгүй програмчилж болно.

Функц гэж юу вэ ?

- Том бодлогыг жижиг бодлогуудад хуваах, том ажлыг жижиг ажлуудад хуваан шийдвэрлэх,
- Жишээ нь: Нэг ажлыг гүйцэтгэхдээ захирал тэнхмийн эрхлэгч нартаа ажлаа хувааж даалгавар өгнө. Тэнхмийн эрхлэгч багш нартаа дахин хувааж даалгавар өгнө. Багш мэдээллийг авч, даалгаврыг биелүүлээд, үр дүнг буцааж өгнө. Тэнхмийн эрхлэгчид үр дүнг авч нэгтгээд захиралд өгнө. Захирал эцсийн үр дүнг нэгтгэж авна.



Функц

- Асуудлыг шийдэхэд хуваагаад нэгтгэх аргыг хэрэгжүүлдэг.
- Функц тодорхойлсоноор кодын давхцалаас зайлсхийнэ.
  - Нэг хэлбэр загварын кодыг дахин дахин бичихгүй.
  - Нэг бичээд олон газар дуудаж ашиглана.
- Програмын дахин ашиглагдах чанарыг нэмэгдүүлдэг.
- Програмын хийсвэрлэлтийг нэмэгдүүлнэ.
- Дэд програм гэж нэрлэж болно.
- Утга буцаадаг бол функц гэнэ. Утга буцаадаггүй бол процедур гэж нэрлэдэг.

Функцийн зарлагаа

Функцийн тодорхойломж

```
харагдалт[Буцах утгын төрөл][Функцын нэр] ( [параметрүүд] ) {
      [Функцын бие]
       return [буцах илэрхийлэл];
   }
```

Функц дуудалт

```
[Хувьсагч]  =  [Функцын нэр] ( [параметрүүд] );
```

Процедурын зарлагаа

- Буцах утга байхгүй бол процедур

Процедурын тодорхойломж

```
public void [Процедурын нэр]( [параметрүүд] ) {
     [Процедурын бие]
}
```

Процедур дуудалт

```
[Процедурын нэр]( [параметрүүд] );
```

Функц

- Функцийн нэр жижиг үсгээр эхэлнэ. Жнь: maxOfList
- Параметрууд нь функцууд хооронд өгөгдөл солилцоход ашиглагдана. Өөрөөр хэлбэл оролтын өгөгдөл, мөн гаралтанд ашиглаж болно. `[төрөл] [параметр] , [төрөл] [параметр]`
- Буцаах утга нь функцийн үр дүнг буцаана.
- Функц дотор зарласан хувьсагчийг локаль хувьсагч гэнэ. Зөвхөн функц дотор ашиглана. Параметртэй ижил нэртэй байж болохгүй.

Локаль хувьсагчууд

- Метод дотор зарлагдах бөгөөд илэрхийллийн үр дүнг түр хадгалах зорилгоор ашиглагдана.

```
public double convert(int num) {
    double result;
    result = Math.sqrt(num * num);
    return result;
}
```

Локаль хувьсагч

Статик функц

- Зарлалтандаа static түлхүүр үгийг ашигладаг ба классын обьектыг үүсгэлгүйгээр классын нэрийг ашиглан дуудаж хэрэглэх боломжтой функц.
- Обьектод харъяалагддаггүй, класст харъяалагддаг.
- Зөвхөн статик хувьсагч болон функцүүд руу хандаж чадна
- main функц нь статик функц тул түүн дотроос статик функцийг л дуудаж болно.
- Статик биш функц дотроос статик биш функц дуудна.
- Статик биш өгөгдөл болон функц руу шууд хандаж чадахгүй боловч обьектын заалтыг ашиглан хандах боломжтой.

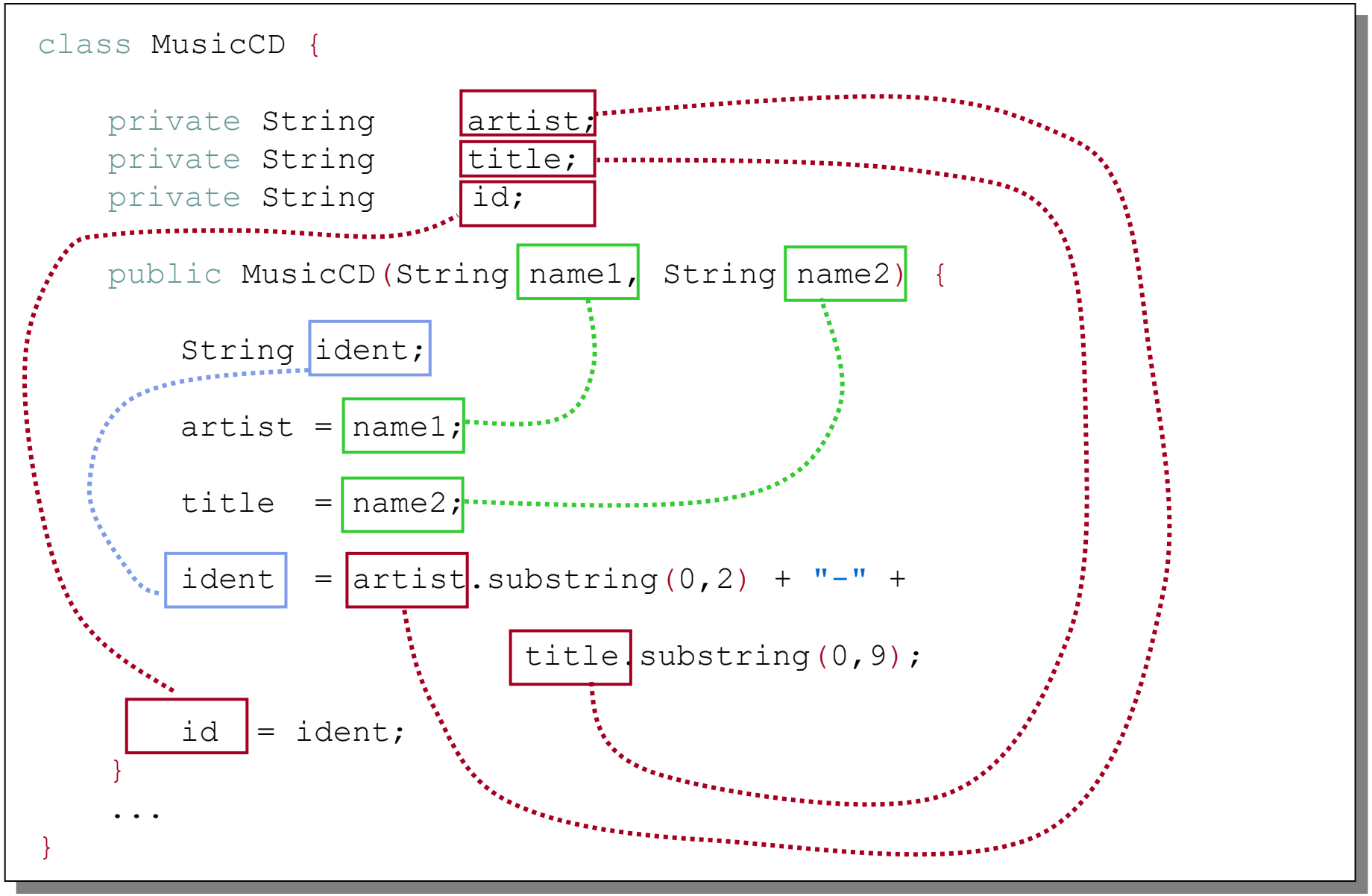
Статик функц

```
public class Program {
    public static int max(int a, int b) {
        if (a > b)
            return a;
        else
            return b;
    }
    public static void printer(int s) {
        System.out.println(s);
    }
    public static void main() {
      printer(max(10,20));
    }
}
```

Анхаарах зүйлс

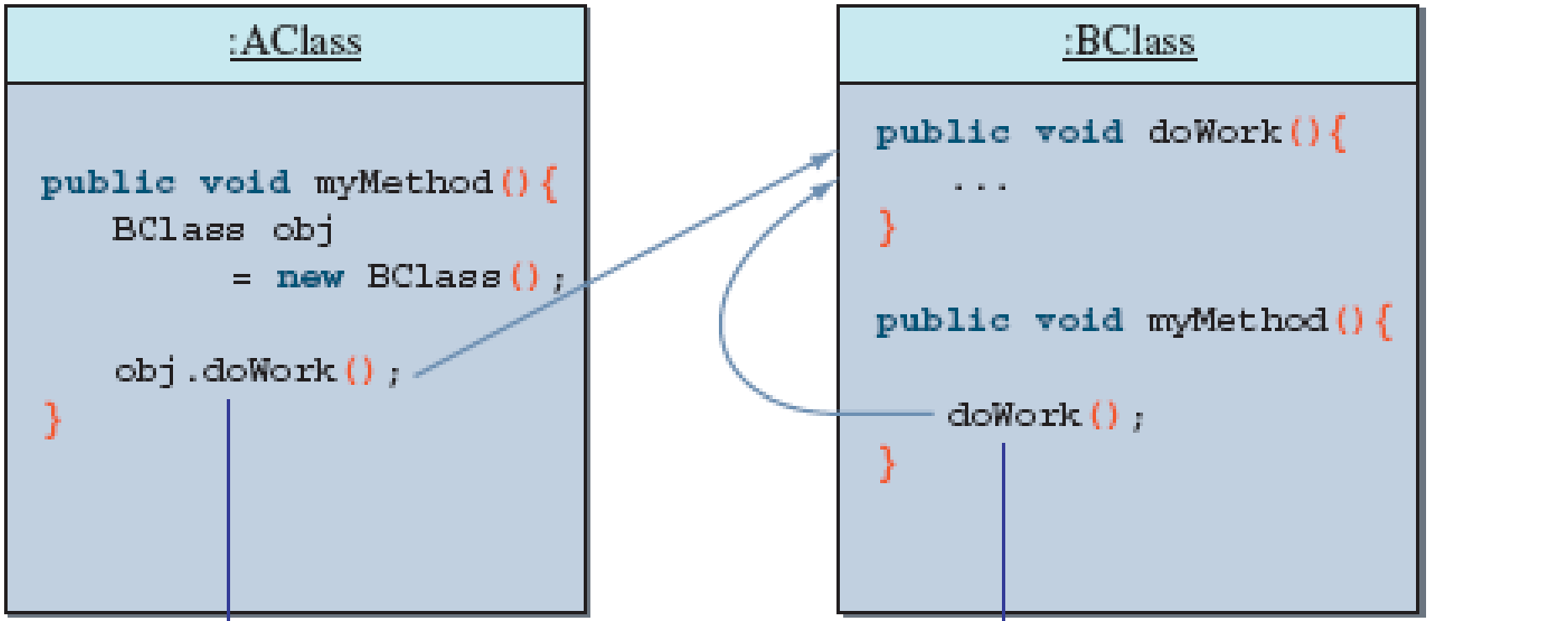
- Функцийн буцааж байгаа утга буцаах өгөгдлийн төрөлтэй таарах ёстой.
- Функцэд дамжуулж байгаа параметрүүдийн тоо, дэс дараалал, тэдгээрийн өгөгдлийн төрлүүд функцийн хүлээж авах параметрүүдийн тоо, дэс дараалал, өгөгдлийн төрлүүдтэй таарах ёстой.

Жишээ


class MusicCD {
private String artist;
private String title;
private String id;
public MusicCD(String name1, String name2) {
String ident;
artist = name1;
title = name2;
ident = artist.substring(0,2) + "-" +
title.substring(0,9);
id = ident;
}
...
}

Ижил классын методуудыг дуудах

- Классын методыг тухайн классын өөр нэг методоос дуудаж ашиглах боломжтой.
  - Энэ тохиолдолд методын нэрийг шууд бичиж дуудна



Call-by-Value параметр дамжуулалт

- Методыг дуудах үед,
  - Параметрт харгалзах аргументын утгыг дамжуулах ба
  - Энэ утгыг хадгалахад зориулж тусдаа санах ой хуваарилагдана

- Методыг ажиллаж байх үед параметр бүрд тусдаа санах ой хуваарилагдсан байдаг тул,
  - Параметр нь методын локаль параметр байна, иймд
  - Параметрт өөрчлөлт хийхэд харгалзах аргументынх нь утгад нөлөөлөхгүй.

Утга дамжуулах - Жишээ

```
class Tester {
    public void myMethod(int one, double two ) {
        one = 25;
        two = 35.4;
    }
}
```

```
Tester tester;
int x, y;
tester = new Tester();
x = 10;
y = 20;
tester.myMethod(x, y);
System.out.println(x + " " + y);
```

Үр дүн

```
10 20
```

Параметрийн санах ойн хуваарилалт

1

```
x = 10;
y = 20;
tester.myMethod( x, y );
```

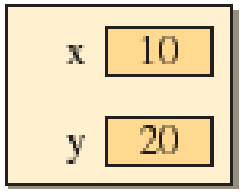
(1)

Дараалал

```
public void myMethod( int one, double two ) {

  one = 25;
  two = 35.4;
}
```

myMethod-г дуудахаас өмнө буюу (1)-т

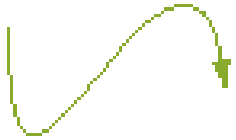


Санах ойн төлөв

Методыг ажиллуулахаас өмнө ямар ч локаль хувьсагч байхгүй

2

```
x = 10;
y = 20;
tester.myMethod( x, y );
```

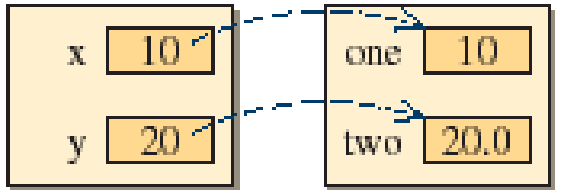


```
public void myMethod( int one, double two ) {
```

(2)

```
  one = 25;
  two = 35.4;
}
```

(2)-т утгууд хуулагдана

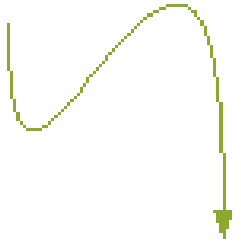


Санах ойд myMethod-д зориулж зай хуваарилах ба аргументын утгуудыг параметр рүү хуулна.

Параметрийн санах ойн хуваарилалт

3

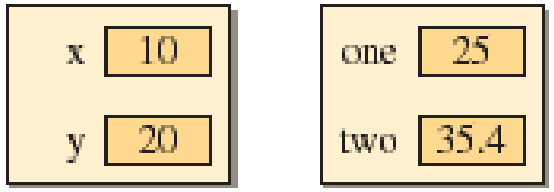
```
x = 10;
y = 20;
tester.myMethod( x, y );
```



```
public void myMethod( int one, double two ) {

  one = 25;
  two = 35.4;
}
```

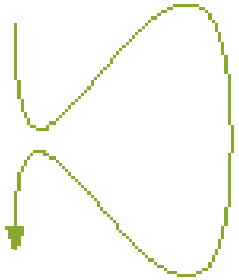
(3)-т буюу утга буцаахын өмнө



Параметрийн утгууд өөрчлөгдөнө

4

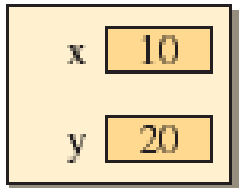
```
x = 10;
y = 20;
tester.myMethod( x, y );
```



```
public void myMethod( int one, double two ) {

  one = 25;
  two = 35.4;
}
```

(4)-т буюу myMethod ажилласны дараа



myMethod-д зориулагдсан санах ойг чөлөөлж, параметрийг устгана. Аргументууд өөрчлөгдөхгүй.

Классыг пакет болгож зохион байгуулах

- Класс А нь класс В-г ашиглах бол эдгээрийн байт код файлууд нь нэг хавтаст байрлаж байх ёстой.
- Програм зохиогчийн тодорхойлсон классыг өөр програмаас дуудаж дахин ашиглах оновчтой арга нь дахин ашиглагдах классуудаа пакетад байрлуулах.
- *Пакет* гэдэг нь Жавагийн классын сан юм.

Пакет үүсгэх

- Дараах алхмуудаар Fraction классыг агуулах myutil пакетыг хэрхэн үүсгэх дарааллыг харуулав.

1. Fraction классын кодын хамгийн эхэнд

```
package myutil;
```

илэрхийллийг бичнэ.

2. **Классын зарлагаа нь** public харагдалттай байна

```
public class Fraction {
   ...
}
```

3. Пакетын нэртэй ижил myutil нэртэй хавтас үүсгэнэ. Жавад пакет бүрд харгалзах(нэг нь нэгтэй) нэг хавтас байх ёстой.

4. Fraction классаа myutil хавтаст хийж компайл хийнэ.

5. CLASSPATH орчны хувьсагчид Myutil хавтсыг агуулж байгаа замыг зааж өгнө.

ДҮГНЭЛТ

- Үйлдэл,
  - Байгуулагч функц
  - Устгагч функц.
- Функц, түүний дуудалт
- BlueJ дээр хэрэгжүүлэх

Эдгээр үйлдлүүд нь классад байх шаардлагатай туслах үйлдлүүд юм.

- Класс байгуулагч болон шаардлагатай бол устгагч функцтэй байна.