Лабораторийн ажил 7: Полиморфизм

Лабораторийн ажлын зорилго:

Классын олон хэлбэржилтийг хэрэгжүүлэх.

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	Интерфейс класс ашиглан полиморфизм хэрэгжүүлэх	Ашиглах (Use)	Хэрэглээ	4, 10
2	Хийсвэр класс ашиглан полиморфизм хэрэгжүүлэх	Ашиглах (Use)	Хэрэглээ	4, 10
3	Бодит болон удамшсан класс ашиглан полиморфизм хэрэгжүүлэх	Ашиглах (Use)	Хэрэглээ	4, 10
4	ЮМЛ диаграм зурах	Зурах (Draw)	Синтез	11
5	Тайлан бичих	Зохион бичих (Compose)	Синтез	12
6	Тайлан хамгаалах	Хамгаалах (defend)	Синтез	12
7	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:

Обьект хандлагат програмчлал ба Полиморфизм

Полиморфизм бол обьект хандлагат програмчлалын үндсэн ойлголт, боломжуудын нэг юм.

Полиморфизм гэж юу вэ?

Обьект хандлагат програмчлалын хэлнүүдийн тулгуур 4 ойлголтын нэг нь Polymorphism буюу олон хэлбэршил юм. Polymorphism гэсэн грек үгийг утгачилан орчуулвал “poly” = “олон”, “morph” = “өөр болох” гэсэн утгыг тус тус агуулна. Полиморфизмыг програмын обьект дээр тулгуурлан өөр өөр зүйлийг хийх арга гэж ойлгож болно. Бүр утгачилвал обьект хандлагат прорамчлал дахь полиморфизм хэмээх ойлголт нь “ижил үйлдлээр ялгаатай үр дүнд хүрэх чадвар” юм. Өөрөөр хэлбэл ялгаатай обьектуудад ижил мэдээ дамжуулахад өөр өөр үйлдэл хийнэ гэсэн үг. Полиморфизм хэмээх ойлголт нь Хийсвэрлэл (Abstraction), Битүүмжлэл (Encapsulation), Удамшил (Inheritance) гэсэн 3 ойлголтын нэгдэл юм. Энэ утгаараа тэдгээр 3 ойлголтын програмд үзүүлэх давуу боломж, ач холбогдлуудыг өөртөө шингээсэн обьект хандлагат технологийн гол ухагдахуун юм. Полиморфизмыг зөв хэрэгжүүлсэнээр програмын

- Уян хатан чанар (flexibility)
- Засвар үйлчилгээ авах чанар (maintainability) нэмэгддэг.

Полиморфизмыг хэрэгжүүлэх арга замууд

Обьект хандлагат програмчлалын хэл бүр полиморфизмыг хэрэгжүүлэх өөр өөрийн арга замуудаар хангагдсан байдаг. Тухайлбал, C++ хэл полиморфизмыг дараах байдлаар хэрэгжүүлдэг. Үүнд:

- Дахин тодорхойлогдсон үйлдэл
- Даран тодорхойлогдсон арга
- Виртуал функц гэх мэт

Тэгвэл Java хэлэнд полиморфизмыг хэрхэн хэрэгжүүлдэг вэ? гэсэн асуулт урган гарна. Java хэл полиморфизмыг дараах байдлаар хэрэгжүүлдэг. Үүнд:

- **Дахин тодорхойлогдсон арга (Method Overloading)**
- **Даран тодорхойлогдсон арга (Method Overriding)**
- Хийсвэр класс
- Интерфэйс

Полиморфизмыг Runtime polymorhism (Dynamic polymorphism) болон Compile time polymorphism (static polymorphism) гэж 2 ангилдаг. Полиморфизмын эдгээр нэршлийг Монголчилбол runtime-ийг динамик буюу хожуу холболт (Dynamic Binding), compile time-ийг статик буюу эрт холболт (Static Binding) гэж нэрлэж болно.

Java програм нь гишүүн функцуудыг биелэх үед нь динамик зохион байгуулалттайгаар дууддаг. Үүнийг динамик холболт буюу хожуу холболт (late binding) гэнэ. Хожуу холболт нь полиморфизмыг хэрэгжүүлэх үндсэн механизмуудын нэг юм. Хожуу холболт нь тухайн үед хэрэг болсон функцыг л санах ойд ачааллан ажиллах учраас санах ойг зөв зохистой ашигладаг. Харин ажиллах хурд харьцангуй удаан байх хандлагатай. Хожуу холболтын сонгодог жишээ бол даран тодорхойлогдсон арга юм.

Эрт холболтын тухай гэвэл үнэн хэрэгтэй compile time polymorphism гэсэн ойлголт байхгүй. Java-д эрт холболт гэж дахин тодорхойлдогдсон аргыг хэлдэг. Эрт холболтыг хэрэгжүүлсэнээр хэрэгтэй функцууд бүгд санах ойд ачааллагдаж ажилдаг учраас санах ой их зарцуулах хандлагатай. Харин хурдан ажиллах давуу талтай.

Полиморфизм нь:

- Нэг интерфэйсийг үйл ажиллагааны ерөнхий классд ашиглах боломжийг бий болгодог.
- Полиморфизм нь нэг интерфэйс тодорхойлоод олон газар хэрэгжүүлэх боломжийг ашигладаг.
- Үйлдэл ялгаатай тохиолдолд ялгаатай үр дүн үзүүлдэг.
- Үр дүн нь үйлдэлд ашиглах өгөгдлийн төрлөөс хамаардаг.
- Полиморфизмын гүйцэтгэдэг чухал үүрэг нь янз бүрийн дотоод бүтэц бүхий обьектууд ижил, төстэй нэг гадаад интерфэйсийг хувааж ашиглах боломжийг олгох юм.
- Полиморфизм нь удамшлыг хэрэгжүүлэхэд өргөн ашиглагддаг.

Java хэлэнд полиморфизмын хэрэгжүүлэхдээ дараах 2 төрлийн ойлголтыг (аргыг) ихэвчлэн ашигладаг (Жич: Интерфэйс, хийсвэр класс гэх мэт ойлголтуудыг өөр лаборатори дээр үзэх учраас түр орхив). Үүнд:

1. Дахин тодорхойлогдсон арга (Method Overloading)
2. Даран тодорхойлогдсон арга (Method Overriding)

Арга гэж юу вэ?

Арга (Method) гэж ямар нэгэн нэрээр нэрлэгдсэн, нэрийг нь ашиглан програмын аль ч хэсэгт дуудаж болох кодын багцыг хэлнэ. Ерөнхийдөө функц гэж ойлгож болно.

Дахин тодорхойлогдсон арга

Java хэлэнд нэг класс дотор ижил нэртэй хоёр болон түүнээс дээш арга тодорхойлох боломжтой байдаг. Эдгээр аргууд нь аргументын (параметрийн) тоо эсвэл буцаах төрлөөрөө ялгаатай байх ёстой. Энэхүү ойлголтыг дахин тодорхойлогдсон арга буюу Method Overloading гэж нэрлэдэг.

1. Java хэлэнд дахин тодорхойлогдсон арга гэж нэрлэдэг. Дахин тодорхойлогдсон аргыг дуудахдаа тухайн аргыг бусад аргуудаас ялгаж өгөх гол зүйл нь тодорхойлохдоо өгсөн аргументын тоо болон (эсвэл) төрөл нь байдаг.
2. Дахин тодорхойлогдсон аргуудыг ялгаж өгөх бас нэг зүйл нь түүний буцаах утгын төрөл юм. Гэхдээ ганцхан буцаах утгын төрлөөр аргуудыг ялгах нь хангалтгүй байдаг.
3. Java хэлэнд дахин тодорхойлогдсон аргуудаас параметрийн тоо болон төрөл нь таарч байгаа аргыг дуудаж ажиллуулдаг.
4. Дахин тодорхойлогдсон арга нь хэрэглэгчид compile time polymorphism-ийг хэрэгжүүлэх боломжийг олгодог.
5. Дахин тодорхойлогдсон аргууд ялгаатай онцгой тохиолдол бий болгдог (үүсгэдэг).
6. Дахин тодорхойлогдсон аргууд нь ялгаатай хандалтын төрөлтэй байж болно.

Жишээ програм:

```
class Overload
{
    void demo (int a)
    {
        System.out.println ("a: " + a);
    }
    void demo (int a, int b)
    {
        System.out.println ("a and b: " + a + "," + b);
    }
    double demo(double a) {
        System.out.println("double a: " + a);
        return a*a;
    }
}
class MethodOverloading
```

```
{
    public static void main (String args [])
    {
        Overload Obj = new Overload();
        double result;
        Obj.demo(10);
        Obj.demo(10, 20);
        result = Obj.demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```

Дээрх demo() арга 3 удаа дахин тодорхойлогдсон. Эхний удаад 1 int төрлийн параметртэй, хоёрдахь удаад 2 int төрлийн параметртэй, гуравдах удаа 1 double төрлийн параметртэйгээр тодорхойлогдсон байна. Дуудаж ажиллуулахдаа буцаах утгын төрөл эсвэл параметрийн тоог ашиглана.

Гаралт:

```
a: 10
a and b: 10,20
double a: 5.5
O/P : 30.25
```

Дахин тодорхойлогдсон аргад ашиглагдах дүрэм

1. Нэг эсвэл түүний дэд класст дахин тодорхойлох боломжтой.
2. Java хэлэнд байгуулагч аргыг дахин тодорхойлох боломжтой.
3. Дахин тодорхойлогдсон аргууд нь ялгаатай аргументын тоотой байх ёстой.
4. Дахин тодорхойлогдсон арга үргэлж нэг классын хэсэг байх ёстой бөгөөд ижил нэртэй, ялгаатай параметрүүдтэй байна.
5. Паратетрийн төрөл эсвэл тоо нь өөр өөр байна. Эсвэл аль аль нь (хоёулаа).
6. Дахин тодорхойлогдсон аргууд нь нэг эсвэл өөр өөр утга буцаах төрөлтэй байж болно.
7. Түүнчлэн дахин тодорхойлогдсон арыг compile time polymorphism (static polymorphism) буюу эрт холболт гэж нэрлэдэг.

Даран тодорхойлогдсон арга

Хүү класст эцэг классын аргатай ижил нэртэй арга байж болно. Ийм тохиолдолд эцэг классын эх кодыг өөрчлөлгүйгээр хүү класст эцэг классын аргыг даран тодорхойлдог. Энэ боломжийг даран тодорхойлох арга гэж нэрлэдэг.

Жишээ програм:

```
public class BaseClass
{
    public void methodToOverride() // BaseClass классын арга
    {
         System.out.println ("Энэ бол BaseClass классын арга.");
    }
}
public class DerivedClass extends BaseClass
```

```
{
    public void methodToOverride() // DerivedClass классын арга
    {
         System.out.println ("Энэ бол DerivedClass классын арга.");
    }
}

public class TestMethod
{
     public static void main (String args []) {
        // BaseClass классын зарлагаа болон обьект
        BaseClass obj1 = new BaseClass();
        // BaseClass классын зарлагаа. Гэхдээ обьект нь DerivedClass
классынх
        BaseClass obj2 = new DerivedClass();
        // BaseClass классын аргыг дуудаж байна
        obj1.methodToOverride();
        // DerivedClass классын аргыг дуудаж байна
        obj2.methodToOverride();
     }
}
```

Гаралт:

```
Энэ бол BaseClass классын арга.
Энэ бол DerivedClass классын арга.
```

Даран тодорхойлогдсон аргад ашиглагдах дүрэм

1. Даран тодорхойлогдсон арга нь зөвхөн удамшлын үед яригдах ойлголт.
2. Даран тодорхойлогдсон аргыг ашиглахын тулд обьект үүсгэх хэрэгтэй бөгөөд даран тодорхойлогдсон аргыг ажиллуулах үед үүсгэсэн обьектоо ашигладаг.
3. Даран тодорхойлогдсон аргуудын утга буцаах төрөл нь ижил байх ёстой.
4. Даран тодорхойлогдсон аргууд нь өөр өөр хандалтын төрөлтэй байж болохгүй.
5. Хийсвэр аргуудыг даран тодорхойлох ёстой байдаг.
6. Статик болон финал аргуудыг даран тодорхойлж болохгүй.
7. Байгуулагч аргуудыг даран тодорхойлж болохгүй.
8. Түүнчлэн даран тодорхойлогдсон арыг runtime polymorphism (dynamic polymorphism) буюу хожуу холболт гэж нэрлэдэг.

Даран тодорхойлох арга дахь super түлхүүр үг

Super түлхүүр үг удамшлын харилцаанд яригддаг ойлголт. Өөрөөр хэлбэл удамшлын харилцаанд эцэг классын ямар нэгэн аргад хандах түүнийг дуудах бол super түлхүүр үгийг ашигладаг. Манай жишээнд Car буюу хүү класс дотроос Vehicle буюу эцэг классын move аргыг дуудахад ашиглаж байна.

Жишээ програм:

```
class Vehicle {
    public void move () {
```

```
        System.out.println ("Тээврийн хэрэгслийг нэг цэгээс нөгөөд
шилжихэд ашигладаг.");
    }
}

class Car extends Vehicle {
    public void move () {
      super.move (); // Super буюу эцэг классын аргыг дуудаж,
ашиглах
      System.out.println ("Машин бол сайн тээврийн хэрэгсэл.");
    }
}

public class TestCar {
    public static void main (String args []){
        Vehicle b = new Car (); // Vehicle классын зарлагаа. Гэхдээ
обьект нь Car классынх
        b.move (); // Car классын аргыг дуудаж байна
    }
}
```

Гаралт:

```
Тээврийн хэрэгслийг нэг цэгээс нөгөөд шилжихэд ашигладаг.
Машин бол сайн тээврийн хэрэгсэл.
```

Ажил гүйцэтгэх дараалал:

1. Интерфейс класс ашиглан полиморфизм хэрэгжүүлнэ.
2. Хийсвэр класс ашиглан полиморфизм хэрэгжүүлнэ.
3. Бодит болон удамшсан класс ашиглан полиморфизм хэрэгжүүлнэ.

Суралцахуйн үр дүнг үнэлэх даалгаврууд:

Дараах классуудыг тодорхойлж квадрат, тэгш өнцөгт, тойрог классуудаас тус тус нэг обьект үүсгэх ба обьект бүрийн талбайг олно. Тэгш өнцөгтийн хувьд периметрийг нэмж олно. Дэлгэцэнд обьект бүрийн өнгө, талбай, тэгш өнцөгт обьектын хувьд периметрийн утгыг мөн хэвлэн харна уу.

```
abstract class Shape
{      private String color;
       int s;
       Shape(String color)
       {      ..................  }
       //талбай олох функц
       abstract void square();
       //талбай хэвлэх функц
       void show(String t)
```

```
        {       ……..        }
        void show()
        {       ……..        }
        //өнгө хэвлэх функц
        //Өнгийг буцаах функц
        String getColor() {……}
}
 class Rectangle extends Quadrate
 {      private int b; int perimeter;
        Rectangle(String color, int x, int y)
        {       ..........................    }
        void calculatePer()
        {       //периметрийг олох функц
        }
        void show()
        {       //периметрийг хэвлэх функц
        }
        ……//тодорхойлох шаардлагатай бусад функц
}
 class Circle extends Shape implements A
 {      private int r; Circle(String color, int r)
        {       ..............}

        ……//тодорхойлох шаардлагатай
                        бусад функцууд
 }
interface A
{
        double pi = 3.14;
        //Тойргийн урт олох функц void calculate ();
}
class Test
{       public static void main(String[] args)
        {       ……..        }
};
```

Ашиглах материал:

• Д.Энхжаргал, Java 2 Объект хандлагат програмчлал, 2009 он.