**Класс гэж юу вэ**

Объект хандалтат програмчлалын хэлэнд класс болон объектийг ашиглан програм бичдэг. Жава хэлэнд өгөгдлийн үндсэн төрөл болох byte, short, int, long, float, double, char, boolean төрлүүд байдаг. Эдгээрийг ашиглан объект хандалтаар програм бичхэд эдгээр өгөгдлийн төрлүүд хангалтгүй юм. Жишээ нь бид “машин” гэсэн өгөгдлийн төрөл үүсгэн үүнийгээ ашиглан T**oyota, Hyundai, BMW, Benz, GMC, Ford, Cadillac** гэх мэт хувьсагчдыг зарлах шаардлага гарч болно. Ийм өгөгдлийн төрлийг өгөгдлийн хийсвэр төрөл буюу “хэв” гэдэг. Тэгвэл класс нь хийсвэр тодорхойлолт буюу “хэв” бол объект нь уг тодорхойлолтыг ашиглан үүсгэсэн бодит зүйл юм. Энэ хэвээр бий болгосон объектууд нь бүгд хоорондоо ижил төстэй шинжүүдийг агуулна. Классгүй бол объект үүсэхгүй, объектгүй бол классын хэрэг хэрэггүй юм. Тиймээс класс нь объектуудыг үүсгэхэд зориулсан өгөгдлийн хийсвэр төрөл юм. Өгөгдлийн хийсвэр төрлийг програмистууд өөрийн програмын онцлогт зориулан үүсгэж ашигладаг. Өгөгдлийн хийсвэр төрлийг жава хэлэнд класс, интерфейс ашиглан бүтээнэ. Бид энэ удаад классын тухай ярьж байгаа ба интерфейсийг хожим нь ярих болно. Жава –д классыг зарлахдаа классын нэрний өмнө class гэсэн түлхүүр үгийг хэрэглэн зарлах ёстой мөн Жава –н өөрийнх нь түлхүүр үгийг ашиглаж болохгүй. Классын нэр нь нэр үг байх, томоор эхлэх мөн олон үгнээс тогтсон бол үг бүрийн эхний үсгийг томоор бичвэл тохиромжтой. Класс дотор тодорхойлогдсон хувьсагч болон функцүүдийг классын гишүүн гэж хэлдэг. Класс дотор тодорхойлогдсон хувьсагч болон өгөгдлийг **загвар хувьсагчид** (***instance variable***) гэдэг. Яагаад гэвэл энэ классаас үүссэн объект болгон уг хувьсагчийн хуулбарыг өөртөө агуулдаг юм. Ихэнх функцүүдийг **static** болон **public** –аар тодорхойлж болохгүй. Анхааруулахад классын ерөнхий хэлбэрт **main( )** функцийг тодорхойлдоггүй ба жава классуудад main( ) функц хэрэггүй байдаг. Харин ямар үед **main( )** функцийг тодорхойлох вэ гэвэл зөвхөн өөрийн тань програмыг эхлүүлж байгаа тэр класс дотор тодорхойлно. Классын хувьсагчид нь уг классын бүх функцүүдэд ашиглагдаж болно. Классын хувьсагчдыг класс дотор классын функцүүдийн өмнө зарладаг.

*class classname {  
 type instance-variable1;  
 type instance-variable2;  
 //……..  
 type instance-variableN;*

*type methodname1(parameter-list){  
 //body of method  
 }  
 type methodname2(parameter-list){  
 //body of method  
 }  
 //………  
 type methodnameN(parameter-list){  
 //body of method  
 }*

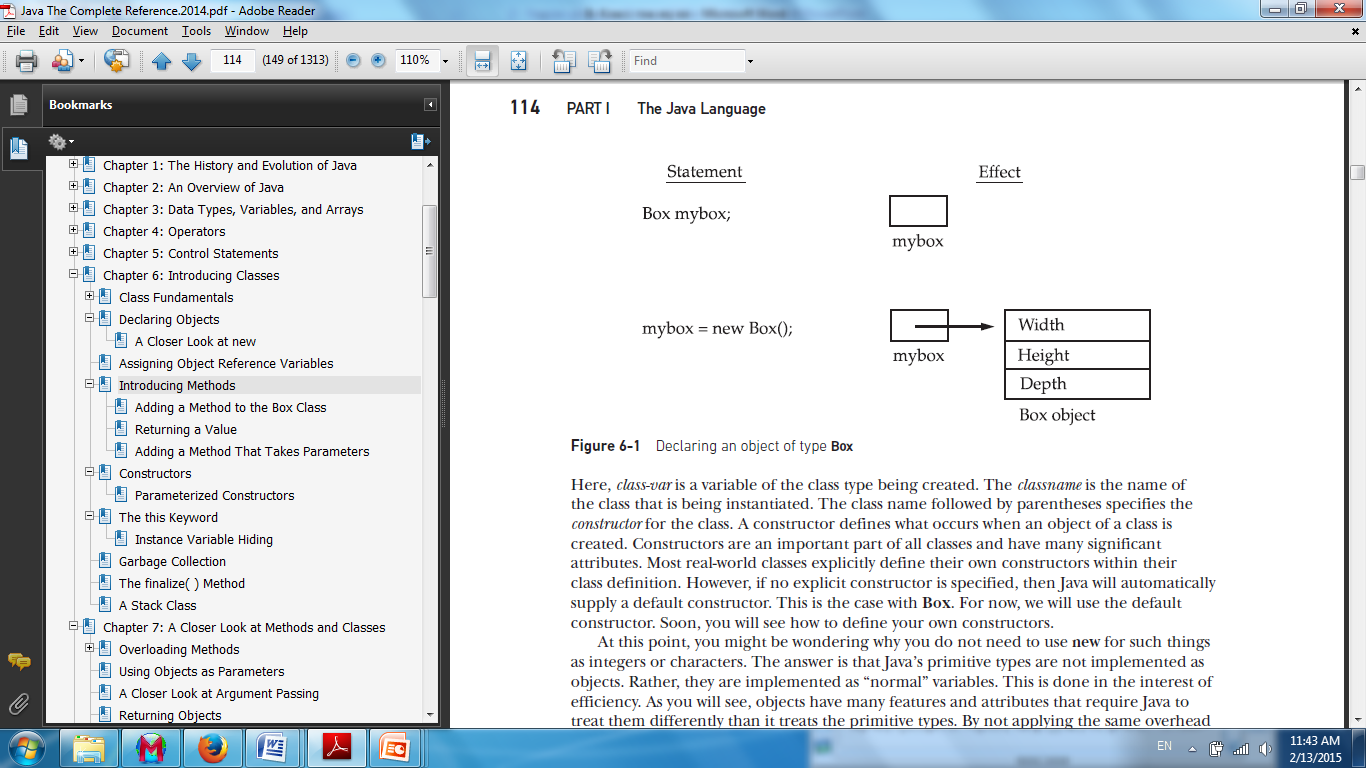
*}*

**Класс –д объект зарлах**

Класс нь өгөгдлийн хийсвэр төрөл учираас шууд ашиглах боломжгүй. Классын объект бол уг классын бодит дүр юм. Класст объект зарлахдаа **new** түлхүүр үгийг ашигладаг.

Жишээ нь: **Box mybox = new Box ( );** Энэ нь Person гэсэн классд odkhuu гэсэн объект зарлаж байна гэсэн үг. Мөн өөрөөр бас дараах бичиглэлээр зарлаж болно. **Box mybox; // объектийг заалтаар заан зарлаж байна.  
 mybox = new Box ( ); // mybox гэсэн объектийг санах ойд байрлуулж байна.**

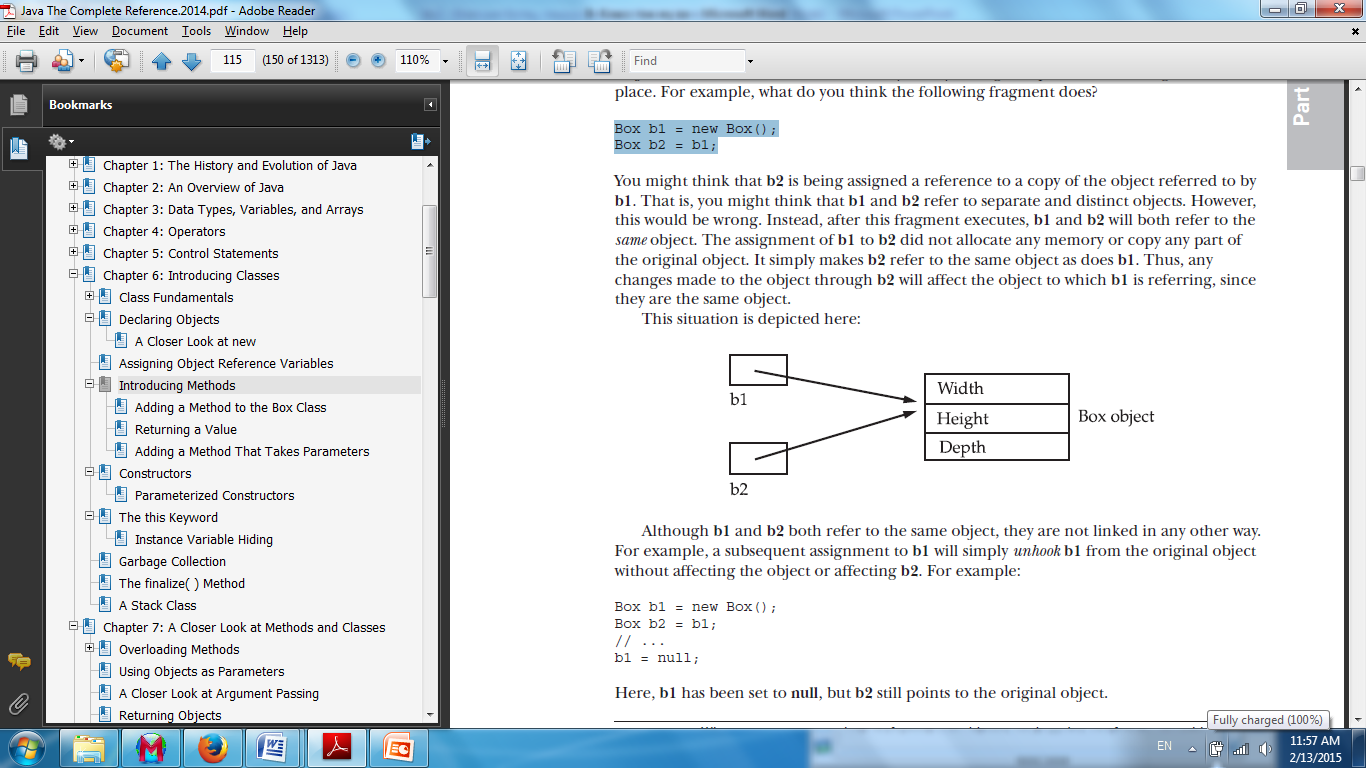
**new түлхүүр үг нь объектод зориулж динамикаар санах ойд байрлуулдаг.**

****

**Объектон заалтан хувьсагчид хандах**

*Box b1 = new Box();*

*Box b2 = b1;*



Та дээрх зүйлийг юу гэж бодож байна. b1, b2 тусдаа объект гэж бодож байна уу үгүйээ. Энэ 2 бол нэг объектийг зааж байгаа болно. Танд зөвөлгөө өгхөд та **нэг объектон заалтан хувьсагчаар** өөр нэг объектон заалтан хувьсагч руу хандах гэж байгаа бол та **тухайн заалтан хувьсагчынхаа хуулбарыг үүсгэх хэрэггүйгээр** зөвхөн заагчийг нь хуулбарлахад болно. Тиймээс энэ 2 нэг ижил объектийг заах болно.

**Функц**

Програм бичих процессийг илүү хялбар болгодо ба хэрэгцээтэй газар нь дуудан хэрэглэх боломжтой. Кодыг дахин давтан бичих үзэгдлээс сэргийлнэ. Функц доторх хувьсагчийг локал хувьсагч гэх ба энэ хувьсагч нь тухайн функц ажиллах үед л хэрэглэгдэх хувьсагч юм. Мөн функц нь дотроо return түлхүүр үгтэй байдаг ба энэ нь тухайн функцээс утга буцаахад хэрэглэгдэх тул тухайн функцийн нэрний өмнө буцаж байгаа өгөгдлийн төрлөөс хамаарсан төрлийг заавал тодорхойлж өгдөг. **return** түлхүүр үггүй функц байж болох ба энэ нь функцийн нэрний өмнө нь **void** түлхүүр үгтэй байна. Тиймээс утга буцаадаг функцийн нэрний өмнө буцаах төрлийг нь тавина. Утга буцаадаггүй функцийн нэрний өмнө **void** төрлийг заавал тавих юм. Энэ утга буцаахгүй гэсэн утга илэрхийлдэг.

Функцийг дараах байдлаар зарладаг.

*Буцаах төрөл*  **Функцын нэр** (Параметр төрөл Параметр) {  
 // Кодын хэсэг  
}

*type* ***name***( *parameter-list* ) {

// body of method

}

Функцийн параметр нь хэд ч байж болох ба олон параметр бичихээр бол параметр тус бүрийн ард талд нь таслалаар тусгаарлан бичдэг.

Параметртэй утга буцаадаггүй функц байж болдог. Энэ нь юунд хэрэг болдог вэ гэвэл тухайн классын гишүүн өгөгдөлд анхны утга олгох зорилгоор хэрэг болдог. Өөрөөр private хандалттай гишүүн өгөгдлүүдэд өөр классаас хандан ажиллах боломжгүй. Тэгэхээр програмын явцад ийм хандалттай гишүүн өгөгдлүүдэд хандаж ажиллах шаардлага гардаг. Тэр үед үйлчилгээ буюу утга буцаадаггүй параметртэй функц ашигладаг.

**Функц дуудалт**

***Нэг классын функцүүд бие биеээ дуудах дүрэм:***

Нэг классын статик функц дотроос тухайн классын статик биш функцийг дуудвал заавал уг классын обьектоор дамжуулж дуудна.Нэг классын статик функцээс тухайн классын статик функцийг дуудвал шууд нэрээр нь эсвэл обьектоор нь дуудаж болно.Нэг классын статик биш функцээс тухайн классын статик болон статик биш функцийг дуудвал шууд нэрээр нь эсвэл обьектоор нь дуудаж болно.

***Нэг классаас өөр классын функц дуудах тохиолдолд:***

Өөр классын статик функцийг дуудвал заавал классын нэрээр нь дуудна.Жишээ нь B класс дотроос А классын static add функцийг дуудвал: A.add( ); Өөр классын статик биш функцийг дуудвал заавал обьектоор нь дамжуулж дуудна.Жишээ нь B класс дотроос А классын add функцийг дуудвал: A myObject = new A( ); myObject.add( );

**Байгуулагч**

Классын объектийг үүсгэх бэлтгэл ажил буюу классын гишүүн өгөгдлүүдэд анхны утга олгоход хэрэглэдэг. Объектийг үүсгэх үед дуудагддаг, класс дотор зарлагдах ба классын нэртэй ижилхэн нэртэй байна. Учир нь байгуулагч функцгүй бол объектийг нь байгуулах буюу үүсгэж болохгүй юм. Нэг класст олон байгуулагч байж болох ба үүнийг хэт ачаалалттай буюу **overloading** гэдэг. Байгуулагч функц нь утга буцаадаггүй тул void эсвэл өөр ямар нэг түлхүүр үг тавьдаггүй. Өмнөө хандалтын тодорхойлогчдыг тавьхдаа програмын бодлогын даалгавраас хамаарч тавьдаг. Зарим тохиолдолд тухайн классын объектийг байгуулах шаардлаггүй гэж үзэхэд урд талд нь **private** гэсэн тодорхойлогч тавьж өгдөг. Байгуулагч функц нь аргументтэй болон аргументгүй байж болдог. Байгуулагч функцийн аргументуудаар нь параметрүүдийг дамжуулж төрөл бүрийн тохиргоотой объектуудыг үүсгэж болдог.

Жишээ нь: **Box** гэсэн классын бүтэц нь уг классын бүх объектуудын хувьд ижил боловч байгуулагч функцийг нь ашиглан өөр өөр **Box** объектууд үүсгэж болно.

*Box b1 = new Box (“apple”);  
Box b2 = new Box (“orange”);  
Box b3 = new Box (“pine apple”);  
Box b4 = new Box (“banana”);*

**Санах зүйл:** Хэрэв програмист нь классын дотор байгуулагч функц бичиж өгөөгүй бол Жава нь автоматаар түүнд аргументгүй байгуулагч функц хийж өгдөг. Өөрөөр хэлбэл байгуулагч функцгүй мэт харагдах класс байж болно. Доорхи жишээнүүдэд объектийн анхны утгуудыг энэ мэтгээр олгох нь сайн зүйл биш юм. Өөрөөр хэлбэл объектийн дотоод мэдээллийг гаднаас нь өөрчилж байна гэсэн үг.

**Аргументүй байгуулагч функц:**

**class** Box {

**double** width;

**double** height;

**double** depth;

}

**class** BoxDemo2 {

**public** **static** **void** main(String[] args) {

Box mybox1 = **new** Box();

Box mybox2 = **new** Box();

**double** vol;

mybox1.width = 10;

mybox1.height = 20;

mybox1.depth = 15;

mybox2.width = 3;

mybox2.height = 6;

mybox2.depth = 9;

vol = mybox1.width \* mybox1.height \* mybox1.depth;

System.***out***.println("Volume is " + vol);

vol = mybox2.width \* mybox2.height \* mybox2.depth;

System.***out***.println("Volume is " + vol);

}

}

----------------------------------------------------------------------------------

**return** түлхүүр үггүй **volume** функц

**class** Box {

**double** width;

**double** height;

**double** depth;

**void** volume() {

System.***out***.print("Volume is ");

System.***out***.println(width \* height \* depth);

}

}

**class** BoxDemo3 {

**public** **static** **void** main(String[] args) {

Box mybox1 = **new** Box();

Box mybox2 = **new** Box();

mybox1.width = 10;

mybox1.height = 20;

mybox1.depth = 15;

mybox2.width = 3;

mybox2.height = 6;

mybox2.depth = 9;

mybox1.volume();

mybox2.volume();

}

}

----------------------------------------------------------------------------------

**return** түлхүүр үгтэй **double** төрлийнбуцаах төрөлтэй **volume** функц

**class** Box {

**double** width;

**double** height;

**double** depth;

**double** volume() {

**return** width \* height \* depth;

}

}

**class** BoxDemo4 {

**public** **static** **void** main(String[] args) {

Box mybox1 = **new** Box();

Box mybox2 = **new** Box();

**double** vol;

mybox1.width = 10;

mybox1.height = 20;

mybox1.depth = 15;

mybox2.width = 3;

mybox2.height = 6;

mybox2.depth = 9;

vol = mybox1.volume();

System.***out***.println("Volume is " + vol);

vol = mybox2.volume();

System.***out***.println("Volume is " + vol);

}

}

----------------------------------------------------------------------------------

**class** Box {

**double** width;

**double** height;

**double** depth;

**double** volume() {

**return** width \* height \* depth;

}

**void** setDim(**double** w, **double** h, **double** d) {

width = w;

height = h;

depth = d;

}

}

**class** BoxDemo5 {

**public** **static** **void** main(String[] args) {

Box mybox1 = **new** Box();

Box mybox2 = **new** Box();

**double** vol;

mybox1.setDim(10, 20, 15);

mybox2.setDim(3, 6, 9);

vol = mybox1.volume();

System.***out***.println("Volume is " + vol);

vol = mybox2.volume();

System.***out***.println("Volume is " + vol);

}

}

----------------------------------------------------------------------------------

**class** Box {

**double** width;

**double** height;

**double** depth;

Box() {

System.***out***.println("Constructing Box");

width = 10;

height = 10;

depth = 10;

}

**double** volume() {

**return** width \* height \* depth;

}

}

**class** BoxDemo6 {

**public** **static** **void** main(String args[]) {

Box mybox1 = **new** Box();

Box mybox2 = **new** Box();

**double** vol;

vol = mybox1.volume();

System.***out***.println("Volume is " + vol);

vol = mybox2.volume();

System.***out***.println("Volume is " + vol);

}

}

----------------------------------------------------------------------------------

**Аргументтай байгуулагч функц:** Хамгийн зөв хэлбэр нь классын байгуулагч функцээр дамжуулан классын гишүүн өгөгдлүүдэд анхны утга олгох нь програмчлалын зөв хэлбэр юм.

**class** Box {

**double** width;

**double** height;

**double** depth;

Box(**double** w, **double** h, **double** d) {

width = w;

height = h;

depth = d;

}

**double** volume() {

**return** width \* height \* depth;

}

}

**class** BoxDemo7 {

**public** **static** **void** main(String[] args) {

Box mybox1 = **new** Box(10, 20, 15);

Box mybox2 = **new** Box(3, 6, 9);

**double** vol;

vol = mybox1.volume();

System.***out***.println("Volume is " + vol);

vol = mybox2.volume();

System.***out***.println("Volume is " + vol);

}

}

**This түлхүүр үг:**

Хэрвээ байгуулагчийн аргументийн нэрийг классын гишүүн өгөгдлийн нэртэй ижилхэн хийвэл яах вэ ? Дараах жишээг ажилглая.

**class** Box {

**double** width;

**double** height;

**double** depth;

Box(**double** width, **double** height, **double** depth) {

width = width;

height = height;

depth = depth;

}

}

**class** BoxThis {

**public** **static** **void** main(String[] args) {

Box mybox1 = **new** Box(10, 20, 15);

Box mybox2 = **new** Box(20, 30, 25);

System.***out***.println("Mybox1");

mybox1.volume();

System.***out***.println("Mybox2");

mybox2.volume();

}

}

Үр дүн: Mybox1 Width = 0.0 Height = 0.0 Depth = 0.0 Mybox2 Width = 0.0 Height = 0.0 Depth = 0.0 **яагаад ингэж гарсан бэ ?**

Үүний учир нь байгуулагчийн аргументад байгаа width, height,depth болон классын гишүүн width;height;depth; өгөгдлүүд нь ижил нэртэй тул алинд нь алинийг нь олгож байгааг жавамэдэхгүйд гол зүйл оршиж байгаа юм. Үнэндээ бол width=width; height=height; depth=depth; ингэж өгсөн команд нь байгуулагчаар орж ирсэн width, height, depth хувьсагчийн утгыг өөрт нь олгосон үйлдэл хийгдсэн байдаг тул классын width; height; depth; хувьсачуудад утга олгогдоогүй юм. Энэ байдлыг засахын тулд мэдээж байгуулагчаар орж ирж байгаа хувьсагчийн нэр болон түүний утгыг авах классын гишүүн өгөгдөл хоёрын нэрийг ондоогоор хйиж өгөх хэрэгтэй болно. Гэвч бодит прогамчлалд энэ арга нь дутагдалтай бөгөөд үүний оронд классын **this** заагч буюу түлхүүр үгийг хэрэглэдэг. **This** гэдэг нь “энэ классын объект, гишүүн өгөгдөл, гишүүн функц” гэсэн утгатай юм. this түлхүүр үг нь өөрийн оршин байгаа класс доторх өгөгдөл болон функцийг заахад хэрэглэддэг.