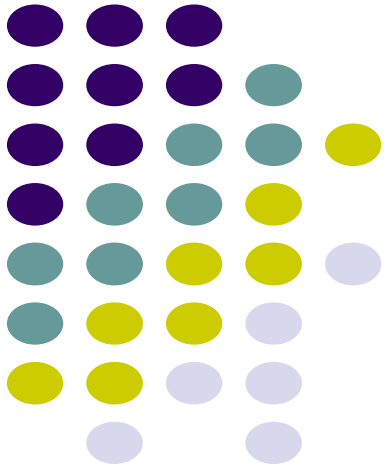
ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ

F.CS202
ОБЪЕКТ ХАНДЛАГАТ ПРОГРАМЧЛАЛ



Лекц №10

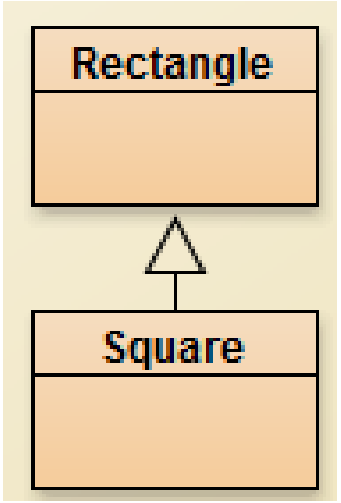
Удамшил

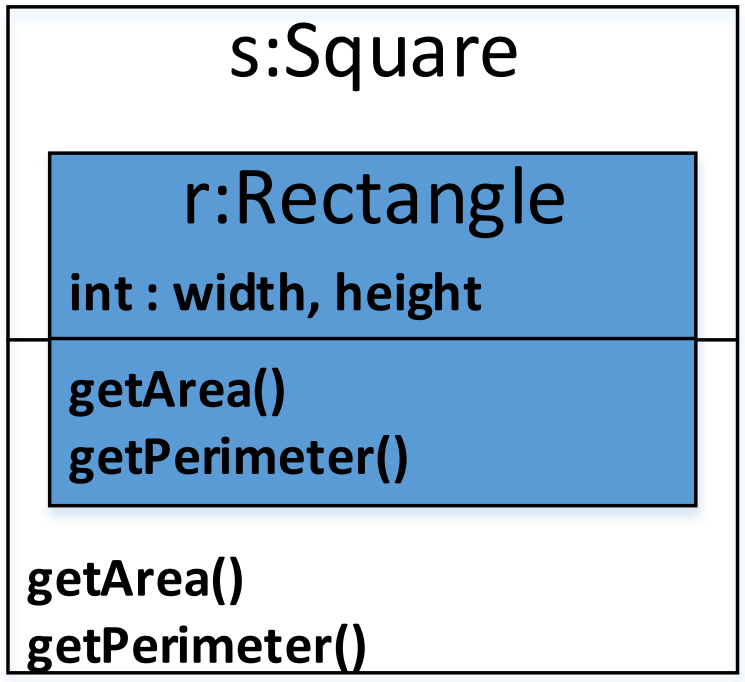
док., дэд проф. Б.Батзолбоо
маг. Б.Мөнхбуян

2021 он

Эцэг, хүү класс

- Rectangle классыг үндсэн класс, эх класс, эцэг класс
- Square классыг үүссэн класс, дэд класс, хүү класс
- Хүү класс нь эцэг классаа өөртөө агуулж байдаг.





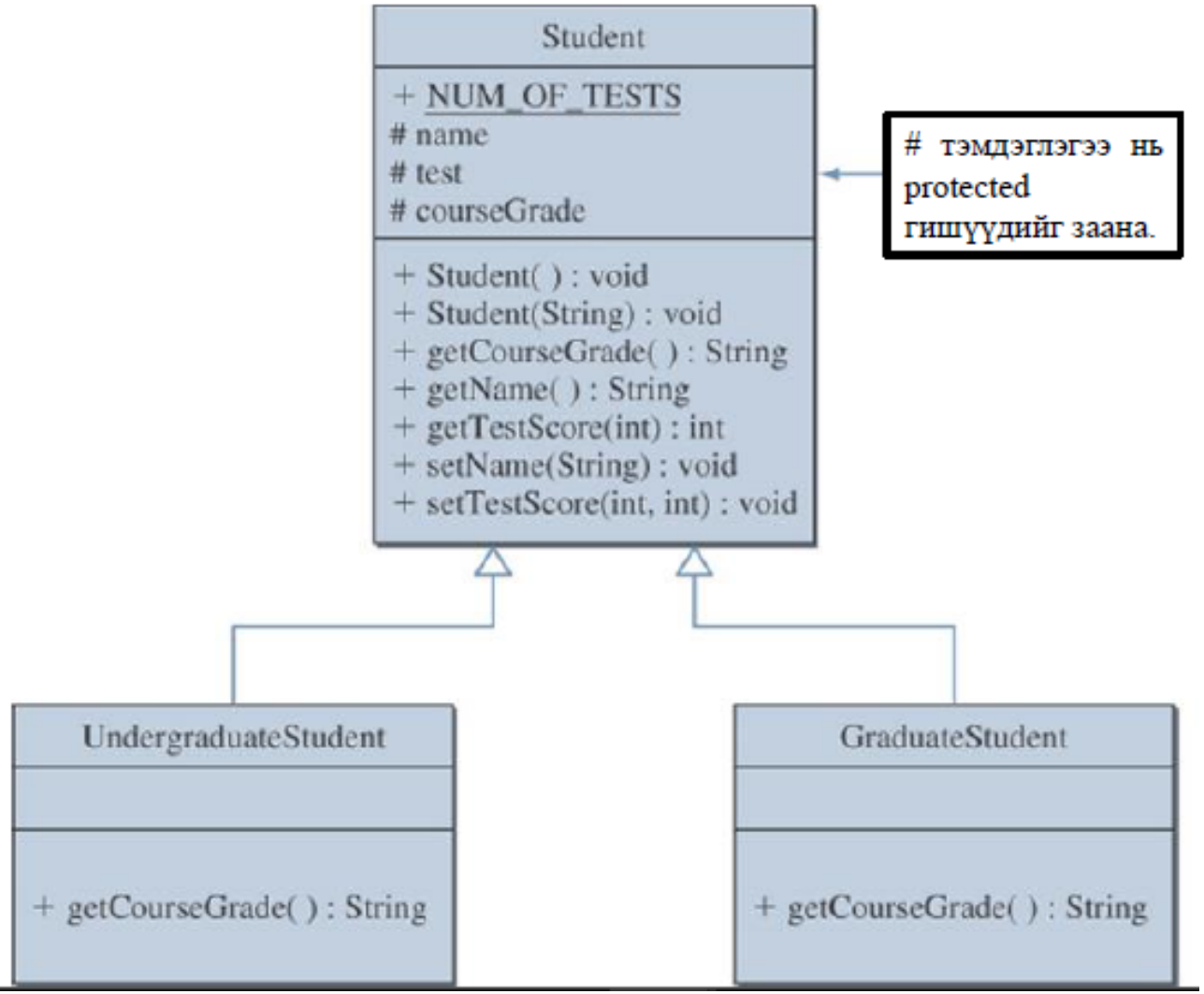

```
public class Program {
    public static void main() {
      Rectangle r = new Rectangle();
      Square s = new Square();
    }
}
```

Агуулга

- Удамшлыг UML диаграм дээр дүрслэх
- Эцэг, хүү класс, Object класс
- Удамшлын харьцааг шалгах instanceof үйлдэл
- Хүү классын байгуулагч функц
- Удамшилд public, private болон protected харагдалтууд
- Дарж тодорхойлох, түүнээс татгалзах, overloading
- Нийлмэл удамшил.

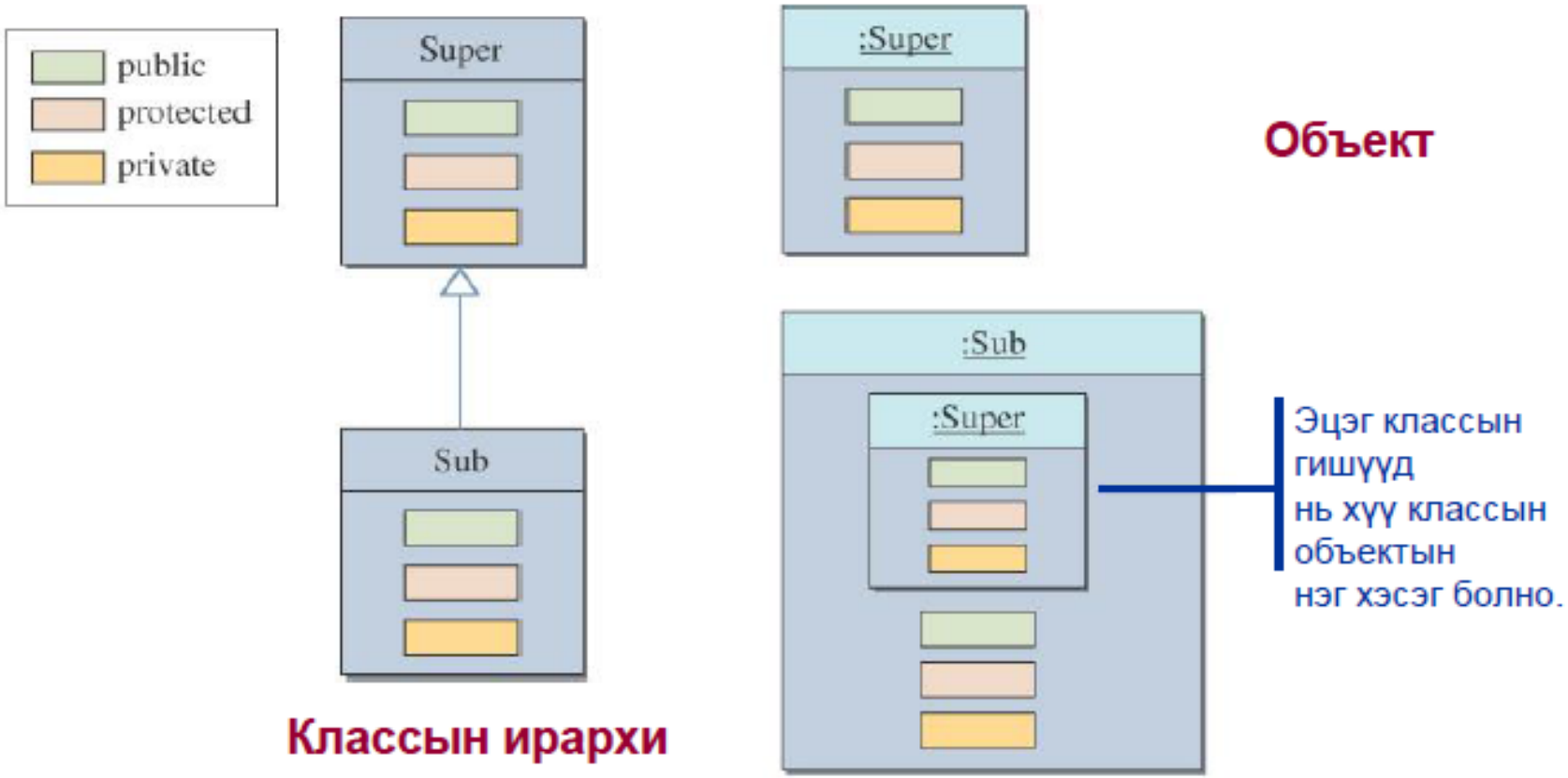
Класс диаграмд удамшлын харилцаа - UML

- Бакалавр, магистр оюутны төгсөх эсэх талаар хэвлэдэг програм бич. Бакалавр оюутан дундаж оноо 70-аас дээш байхад, магистр оюутан 80-аас дээш байхад төгсөнө. Удамшил ашиглана.
- Програм дахь классууд:
  - Student эцэг класс,
  - UndergraduateStudent, GraduateStudent хүү класс.



Эцэг, хүү класс

- Rectangle классыг үндсэн класс, эх класс, эцэг класс
- Square классыг үүссэн класс, дэд класс, хүү класс
- Хүү класс нь эцэг классаа өөртөө агуулж байдаг.



Object класс

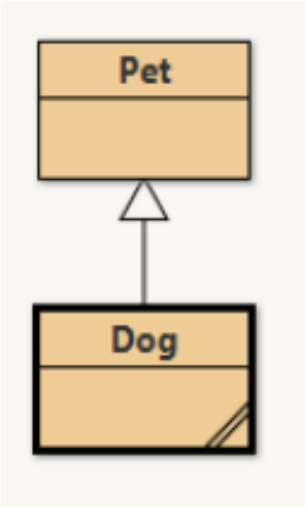
- Java бол объект хандлагат програмчлалын хэл. Иймээс бүх класс эцэг класстай байдаг. Иймээс анх ямар классаас удамшдаг вэ гэсэн асуулт гарч ирнэ. Java хэлэнд бүх классын эцэг класс object класс байдаг.

```
public class Rectangle {
  ...
```

- Дээрх жишээнд Rectangle класс object классаас удамшсан байна. Object нь класс тул үйлдэлтэй байна. Үүнд:
  - clone() - объектийг хуулбарлах,
  - equals(Object obj) - 2 объектыг тэнцүү эсэхийг шалгах,
  - finalize() – устгагч функц удамшиж ирнэ,
  - toString() – объектыг тэмдэгт мөр

Instanceof үйлдэл – Удамшлын харьцааг шалгана

```
public class Pet {
    private String name;
    public String getName() { return name; }
    public void setName(String newName) { name=newName; }
}
public class Dog extends Pet {
    ...
}
```




```
public class Program {
    public static void main() {
        Dog baavgai = new Dog();
        if (baavgai instanceof Pet)
            print(baavgai.getName() + " is Pet");
    }
}
```

Хүү классын байгуулагч функц

- Эцэг классын байгуулагч функцийг тохирох параметруудаар дуудах ёстой. Эцэг класс олон хэлбэржилт бүхий байгуулагч функцтэй байж болно
- Өөрийн нэмж тодорхойлсон хувийн харагдалттай гишүүдэд анхны утга оноох ёстой.
- Хэрэв байгуулагч функц бүхий эцэг классаас хүү удамшсан боловч байгуулагч функц тодорхойлоогүй бол эцэг классын параметргүй (буюу өгөгдмөл) байгуулагчийг дууддаг.

эцэг / хүү	байгуулагчтай	байгуулагчгүй
байгуулагчтай	заавал	байж болно
байгуулагчгүй	байж болохгүй	байж болно

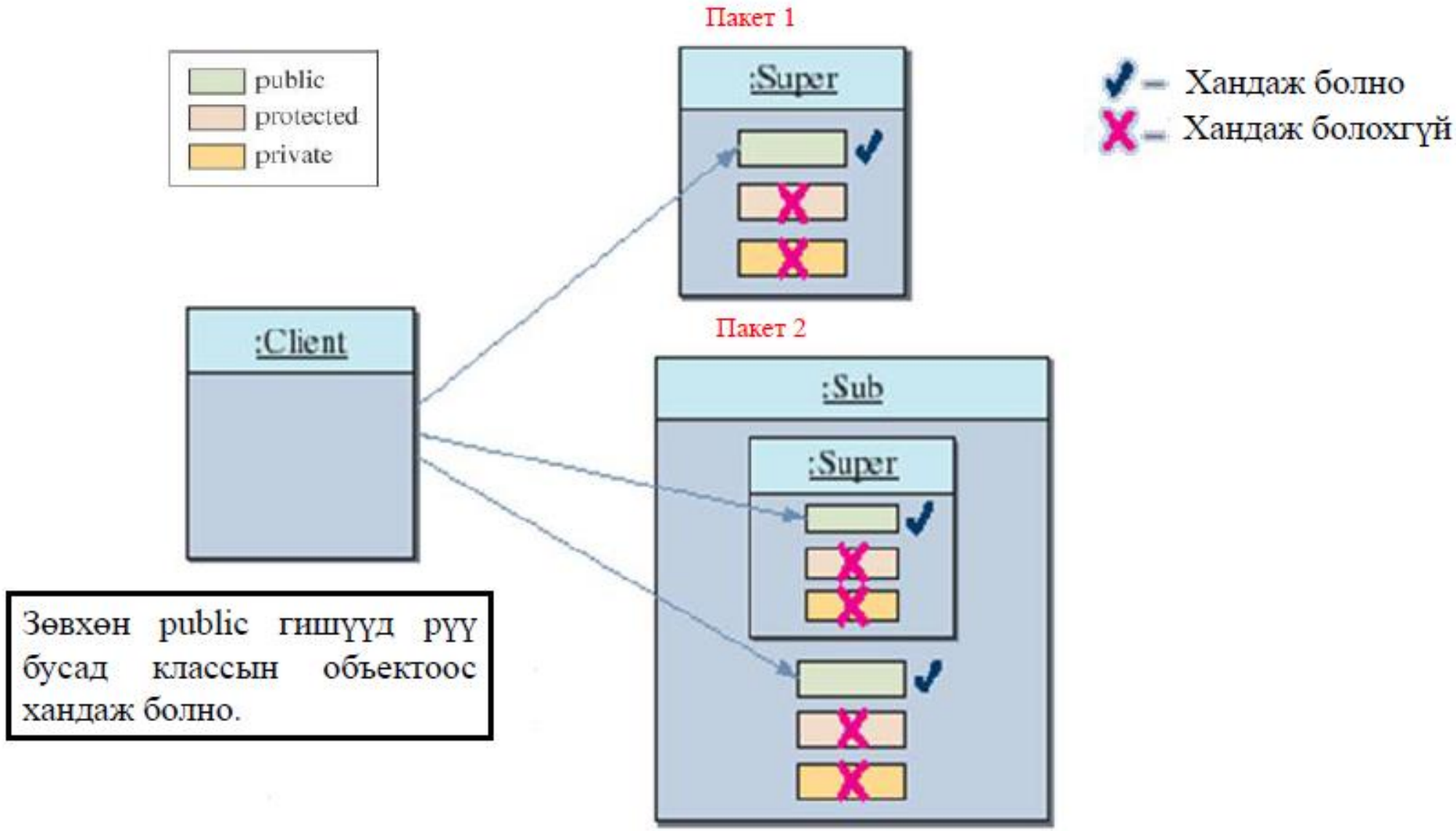
Хүү классын байгуулагч функцийн жишээ

```
public class Computer {
  private String manufacturer; ...
  public Computer (String manufacturer, ...) {
    this.manufacturer = manufacturer; ...
  }
}
public class Laptop extends Computer {
  private double weight; ...
  public Laptop (String manufacturer, ...,
                 double weight, ...) {
    super(manufacturer, ...);
    this.weight = weight; ...
  }
}
```

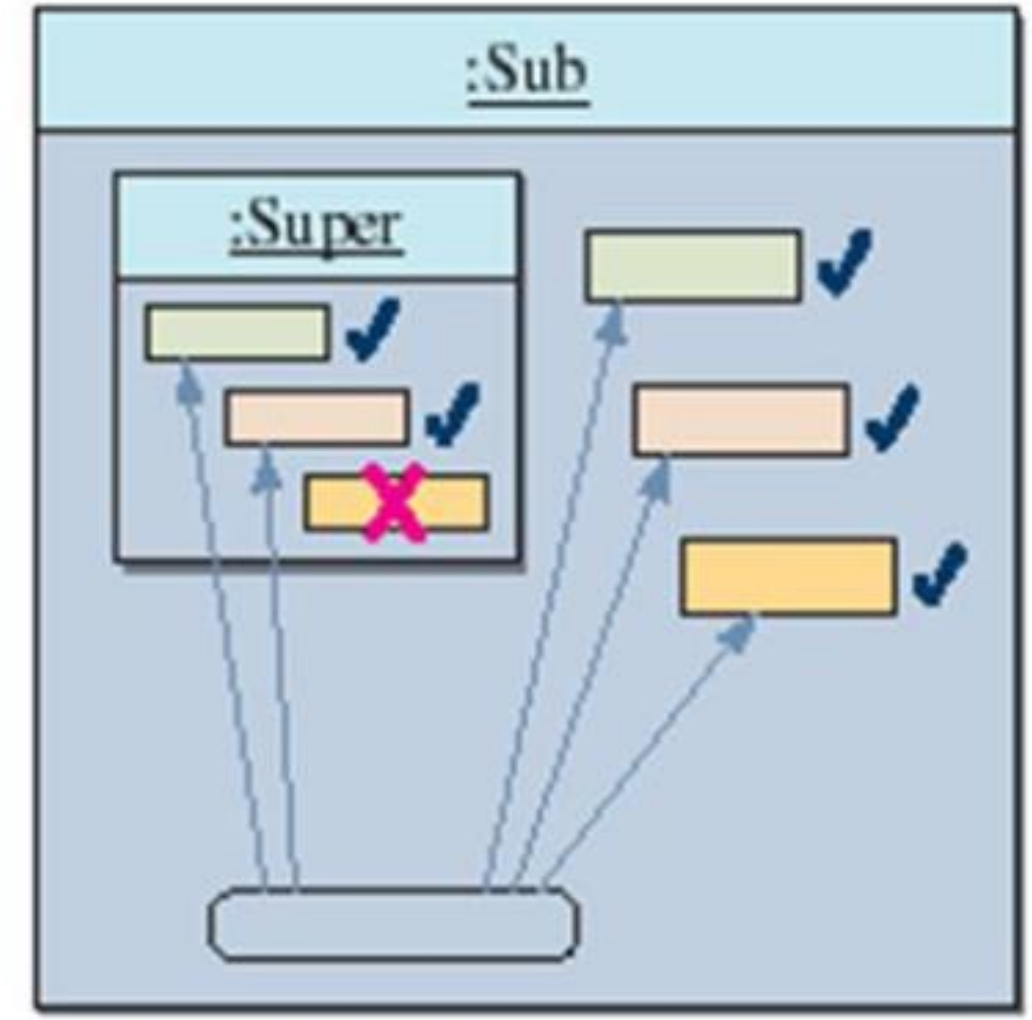
private VS protected VS public

- Public харагдалттай гишүүн болон үйлдэлд бүх классын объект хандаж болдог.
- private харагдалттай гишүүн болон үйлдэлд зөвхөн тухайн классын объект хандаж болно. Харин бусад классын болон түүний хүү классын объект хандаж болохгүй.
- protected харагдалттай гишүүн болон үйлдэлд тухайн класс болон түүний хүү классын объект хандаж болно. Харин бусад классын объект хандаж болдоггүй. Иймээс эцэг класс хүү классуудад хандах боломжийг олгохдоо protected гэж зарладаг.

Бусад классын объектоос хандах



Хүү классын үйлдлээс эцэг классын гишүүд рүү хандах



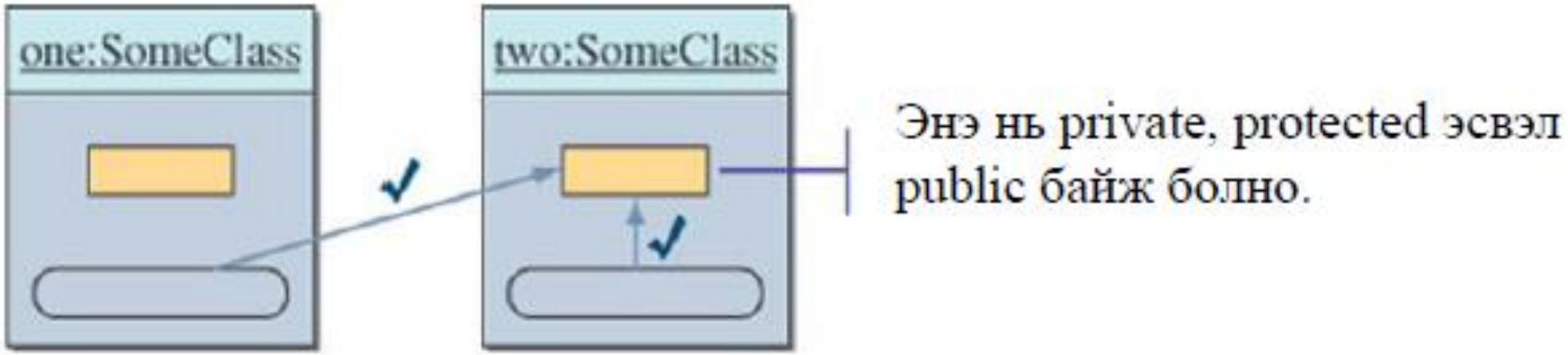
✓ – Хандаж болно

✗ – Хандаж болохгүй

Хүү классын үйлдлээс эцэг классын private гишүүдээс бусад руу хандаж болно.

Нэг классын объект нөгөө объект руу хандах

- Нэг классын 2 объект бие бие рүүгээ хандах



Дарж тодорхойлох (override)

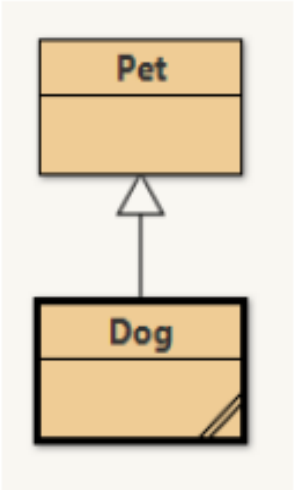
- Хэрэв хүү классын үйлдэл нь эцэг классын үйлдэлтэй ижил тодорхойломжтой байвал үйлдлийг дарж тодорхойлох гэнэ.
- Энэ нь эцгийн үйлдлийг сайжруулж өөрчлөх үед ашиглагдана.

```
public class A { ...
   public int M (float f, String s) { bodyA }
}
public class B extends A { ...
   public int M (float t, String l) { bodyB }
}
```

- Хэрэв B классын (эсвэл B-ийн хүү классын) объектын M үйлдлийг дуудвал bodyB ажиллана.
- B класс дотроос super.M(...) гэж bodyA –г ажиллуулж болно.
- Хүү класс болон эцэг классд M үйлдэл нь ижил буцаах утгын төрөлтэй байна.

Дарж тодорхойлох - Жишээ

```
public class Pet {
     ...
     public String speak() {
          return name + ":I'm pet";
     }
}
public class Dog extends Pet {
    public String speak() {
        return name + ":How how";
    }
}
```




```
public class Program {
    public static void main() {
        Pet pet = new Pet();
        Dog baavgai = new Dog();
        System.out.println(pet.speak());
        System.out.print(baavgai.speak());
```

Дарж тодорхойлох жишээ

```
public class Shape {
      public String getName() { return “Shape”; }
      public double getArea() { return 0;}
}
public class Square extends Shape {
      private int x,y,length;
      public Square(int x, int y, int length) {...}
      public String getName() { return “Square”; }
      public double getArea() { return length*length;}
}
```

Үйлдлийн олон хэлбэржилт - overloading

- Ижил нэртэй,
- ялгаатай параметр болон ялгаатай дараалалтай параметрүүдтэй үйлдлүүд,
- нэг класс дотор байвал үүнийг үйлдлийн олон хэлбэржилт буюу overloading гэнэ.
- Байгуулагч функцүүд ихэвчлэн олон хэлбэртэй байдаг.
- Жишээ нь:
- **`MyClass (int inputA, int inputB)`**
- **`MyClass (float inputA, float inputB)`**

Дарж тодорхойлохоос татгалзах

- Хэрэв эцэг класс тодорхой үйлдлээ хүү классаар дахин тодорхойлуулах хүсэлгүй бол final түлхүүр үг ашиглан хорьж болно.

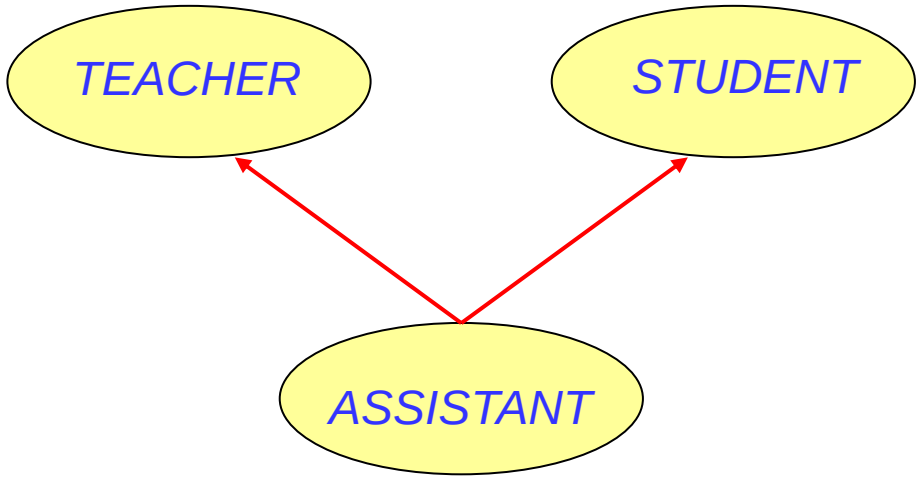
```
public clase Shape {
      public String getName() { return “Shape”; }
      public final double getArea() { return 0;}
}
public class Square extends Shape {
      private int x,y,length;
      public Square(int x, int y, int length) {...}
      public String getName() { return “Square”; }
}
```

super, this жишээ

```
public class A { ...
    public A() {}
    public A(int) { this();}
    public int M (float f, String s) { M(f); }
    public int M (float f) { bodyA }
}
public class B extends A { ...
    public B() { super(); }
    public B(int ) { this(); }
    public int M (float t, String l) { this.M(f);
                                       super.M(f,s); bodyB }
    public int M (float k) { bodyA }
}
```

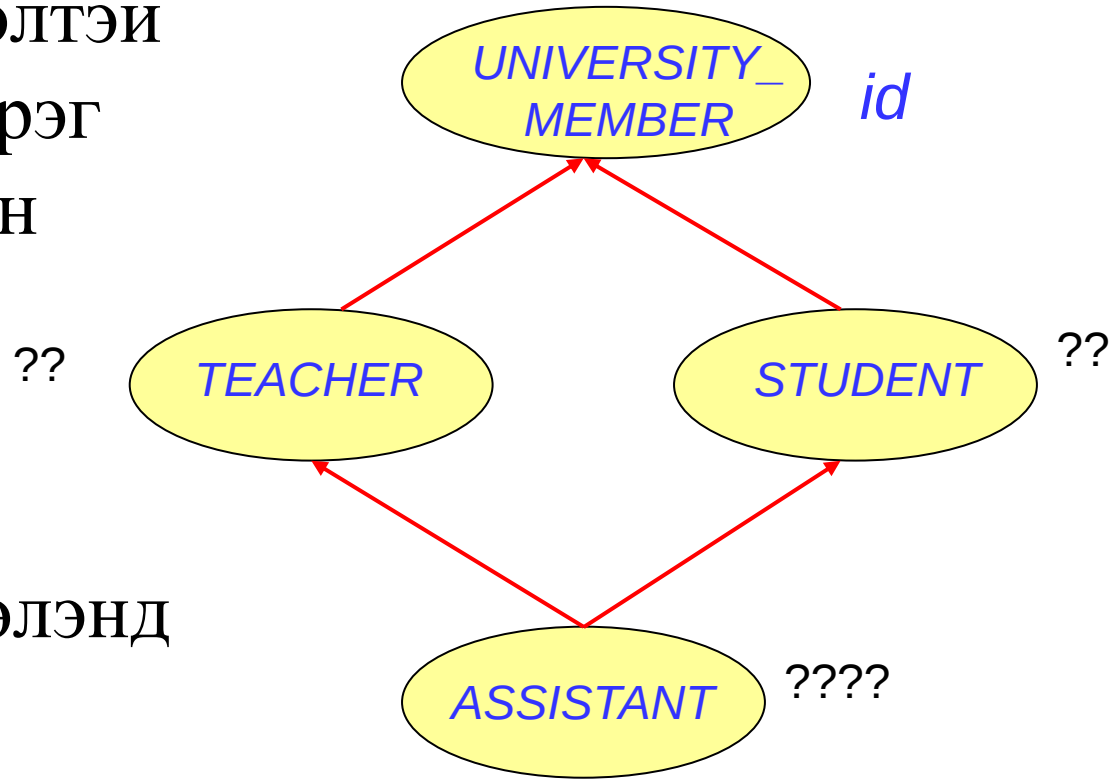
Нийлмэл удамшил

- 2 ба түүнээс олон классаас шинэ класс удамшиж гарахыг нийлмэл удамшил гэнэ.



Нийлмэл удамшлыг хэрэгжүүлэх хүндрэл

- Нийлмэл удамшлыг хэрэгжүүлэхэд дараах хүндрэлтэй байдал үүсдэг тул Java, C# зэрэг хэлүүдэд нийлмэл удамшлийн боломжийг хязгаарлан хэрэгжүүлдэг.
- Иймээс дараах хэлбэрийн нийлмэл удамшил Java, C# хэлэнд боломжгүй.



ДҮГНЭЛТ

- Удамшлыг UML диаграм дээр дүрслэх
- Эцэг, хүү класс, Object класс
- Удамшлын харьцааг шалгах instanceof үйлдэл
- Хүү классын байгуулагч функц
- Удамшилд public, private болон protected харагдалтууд
- Дарж тодорхойлох, түүнээс татгалзах, overloading
- Нийлмэл удамшил.