20251027 Monday

Today

- Equals()
- Protected
- Final

Next:

- Test 2, next Monday (Nov. 3): online OR local
- Next class, Wednesday (Nov. 5): review for Test 2
- Friday (Nov. 7): Chapter 12 Exception Handling

Class Object

https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html#equals-java.lang.Object-

java.lang

Class Object

java.lang.Object

`public class Object`

Class `Object` is the root of the class hierarchy. Every class has `Object` as a superclass. All objects, including arrays, implement the methods of this class.

Modifier and Type	Method
`protected Object`	`clone()`
`boolean`	`equals(Object obj)`
`protected void`	`finalize()`
`Class<?>`	`getClass()`
`int`	`hashCode()`
`void`	`notify()`
`void`	`notifyAll()`
`String`	`toString()`
`void`	`wait()`
`void`	`wait(long timeout)`
`void`	`wait(long timeout, int nanos)`

Equals()

```
public boolean equals(Object obj)
```

Indicates whether some other object is "equal to" this one.

Parameters:

`obj - the reference object with which to compare.`

Returns:

`true if this object is the same as the obj argument; false otherwise.`

```
class MyClass{
    int a;
    public MyClass(int a){ this.a = a; }
    public boolean equals(MyClass obj){
        return ( this.a == obj.a );
    }
}
```

```
class Main {
    public static void main(String[] args) {
        MyClass x = new MyClass(123);
        MyClass y = new MyClass(234);
        MyClass z = new MyClass(123);
        System.out.println( x.equals(y) );
        System.out.println( x.equals(z) );
        System.out.println( z.equals(y) );
        System.out.println( z.equals(x) );
    }
}
```

- What if you do not define/override the equals()?

```
class MyClass{
    int a;
    public MyClass(int a){ this.a = a; }
    //public boolean equals(MyClass obj){
    //    return ( this.a == obj.a );
    //}
}
```

```
class Main {
    public static void main(String[] args) {
        MyClass x = new MyClass(123);
        MyClass y = new MyClass(234);
        MyClass z = new MyClass(123);
        System.out.println( x.equals(y) );
        System.out.println( x.equals(z) );
        System.out.println( z.equals(y) );
        System.out.println( z.equals(x) );
    }
}
```

```
false
false
false
false
```

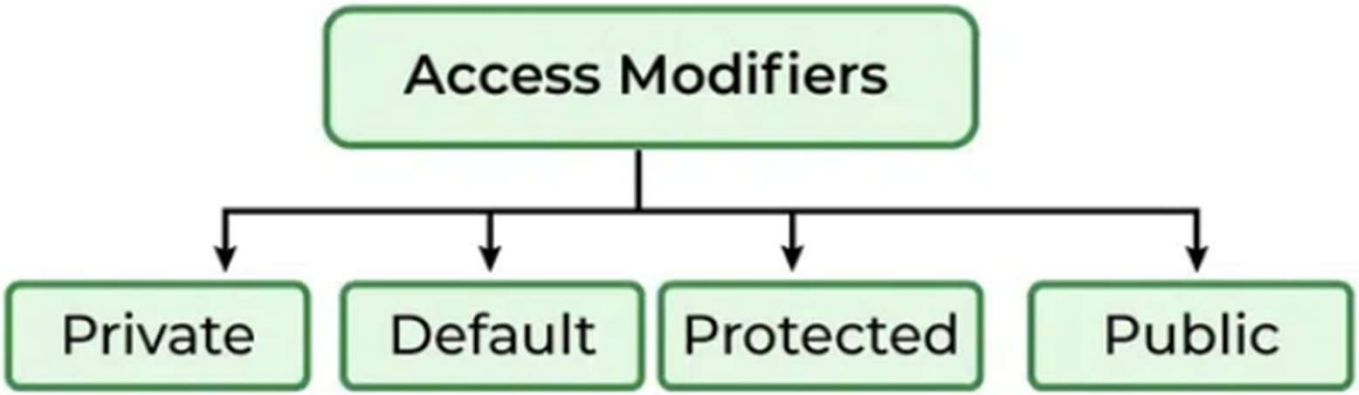
```
class MyClass{
    int a;
    public MyClass(int a){ this.a = a; }
    //public boolean equals(MyClass obj){
    //    return ( this.a == obj.a );
    //}
}
```

```
class Main {
    public static void main(String[] args) {
        MyClass x = new MyClass(123);
        MyClass y = x;
        MyClass z = new MyClass(123);
        System.out.println( x.equals(y) );
        System.out.println( x.equals(z) );
        System.out.println( z.equals(y) );
        System.out.println( z.equals(x) );
    }
}
```

```
true
false
false
false
```

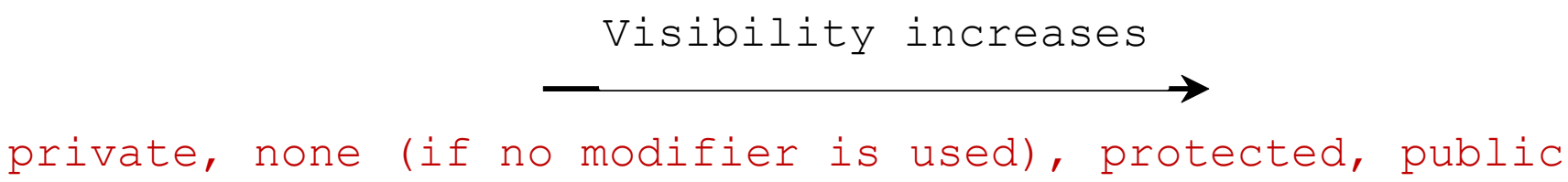
Access Modifiers in Java

- In Java, **access modifiers** are essential tools that define how the members of a class, like **variables, methods**, and even the **class** itself, can be accessed from other parts of our program.



The `protected` Modifier

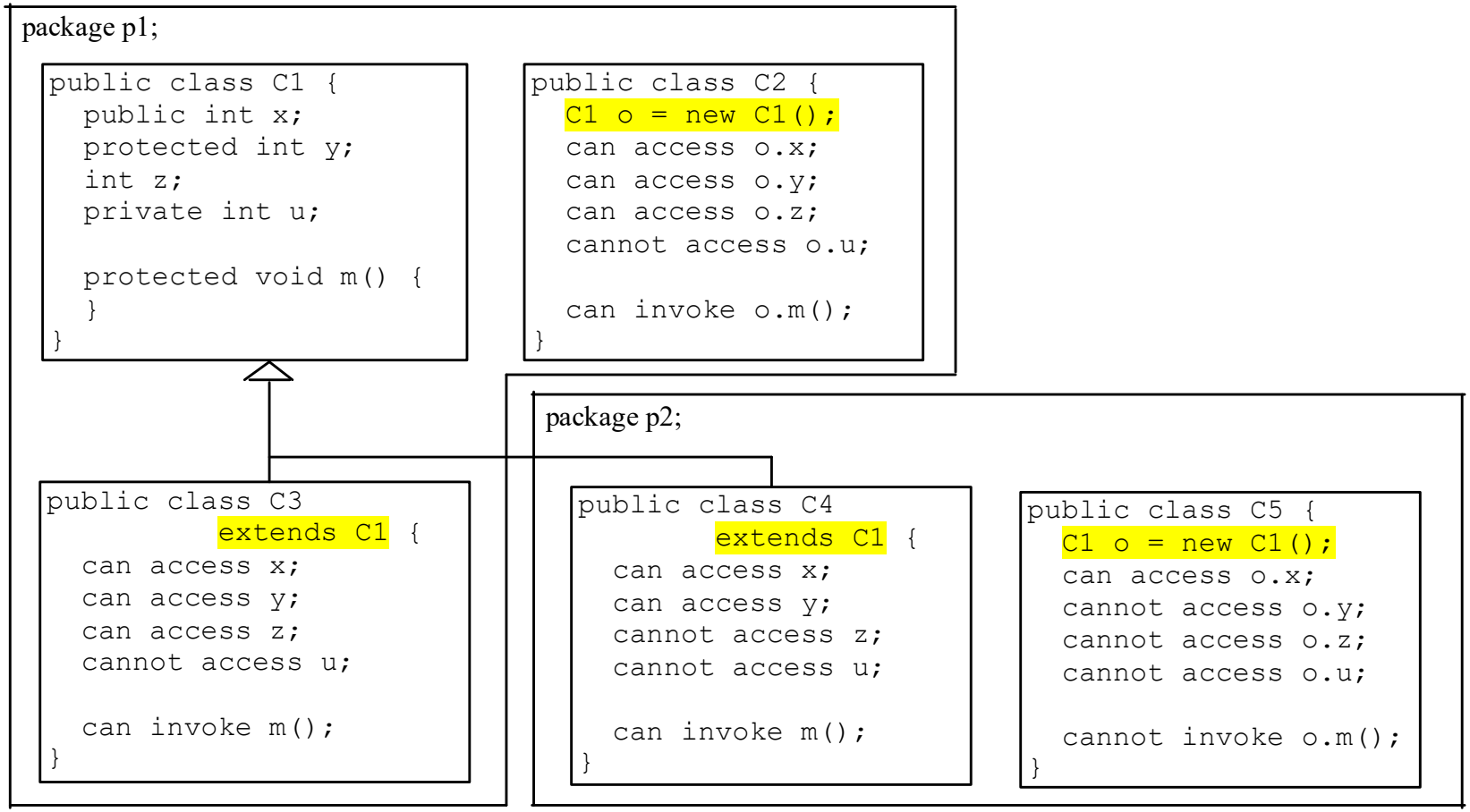
- The `protected` modifier can be applied on data and methods in a class. A protected data or a protected method in a public class can be accessed by any class in the same package or its subclasses, even if the subclasses are in a different package.
- private, default, protected, public



Accessibility Summary

Modifier on members in a class	Accessed from the same class	Accessed from the same package	Accessed from a subclass	Accessed from a different package
public	✓	✓	✓	✓
protected	✓	✓	✓	-
default	✓	✓	-	-
private	✓	-	-	-

Visibility Modifiers



Practice (later)

- Please practice/try,
- When there is error(s), think, and correct it

- You should design your own program to test your understanding of this topic and the other knowledge.

```
class MySuperClass{
    public int pub;
    protected int prt;
    int dft; //default
    private int prv;

    public MySuperClass(int a, int b, int c, int d){
        pub = a; prt = b; dft = c;  prv = d;
    }
    public void display(){
        System.out.println("pub: "+pub);
        System.out.println("prt: "+prt);
        System.out.println("dft: "+dft);
        System.out.println("prv: "+prv);
    }
}
```

```
class MySubClass extends MySuperClass{
    public MySubClass(int a, int b, int c, int d){
        super(a, b, c, d);
    }
    public void display(){
        System.out.println("pub: "+pub);
        System.out.println("prt: "+prt);
        System.out.println("dft: "+dft);
        System.out.println("prv: "+prv);
    }
}
```

```
class Main {
    public static void main(String[] args) {
        MySuperClass x = new MySuperClass(1, 2, 3, 4);
        MySubClass y = new MySubClass(5, 6, 7, 8);
        System.out.println(x.pub);
        System.out.println(x.prt);
        System.out.println(x.dft);
        System.out.println(x.prv);
        x.display();
        y.display();
    }
}
```

A Subclass Cannot Weaken the Accessibility

A subclass may override a protected method in its superclass and change its visibility to public. However, a subclass cannot weaken the accessibility of a method defined in the superclass. For example, if a method is defined as public in the superclass, it must be defined as public in the subclass.

Note (4 of 4)

The modifiers are used on classes and class members (data and methods), except that the final modifier can also be used on local variables in a method. A final local variable is a constant inside a method.

Last topic today: final

- The `final` variable is a constant:

```
final static double PI = 3.14159;
```

The `final` Modifier

- The `final` class cannot be extended:

```
final class Math {

   ...

}
```

- The `final` method cannot be overridden by its subclasses.

```
class Main {
    public static void main(String[] args) {
        MySubClass a = new MySubClass(123);
        System.out.println(a.x);
    }
}
class MySuperClass{
    public int x;
    MySuperClass(int a){ x = a; }
}
class MySubClass extends MySuperClass{
    MySubClass(int a){ super(a); }
}
```

- What if you add “final” to the super class as:

```
final class MySuperClass{
    public int x;
    MySuperClass(int a){ x = a; }
}
```

- Please practice “equals”/“protected”/“final” following my examples or writing your own

- Test 2: next Monday (Nov. 3)
- Next class: review for Test 2.