Class Members and Data Encapsulation

Fields

Earlier in this course we added a **memory** field to a **Calculator** class.

```
public class Calculator {
    private int memory;

    public void keepInMemory(int number) {
        memory = number;
    }

    public int readFromMemory() {
        return memory;
    }

    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }

    ...
}
```

This is the field. As it's private, it's only accessible inside this class by other members, such as methods

Public methods for reading and writing to the private variable

The value of "memory" is unique for every instance of a Calculator

Fields are per-instance

Declaring a **field** in a **class** doesn't mean that the instance of that field belong to the class:

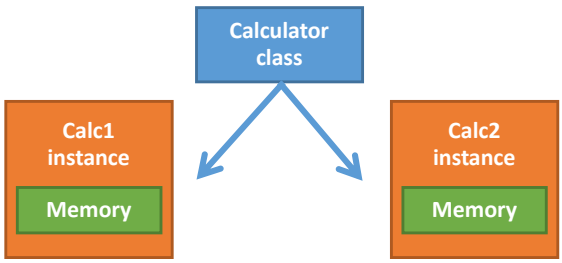
```
// We'll start by creating two instances of the same class:
Calculator calc1 = new Calculator();
Calculator calc2 = new Calculator();

// We'll save a number in the memory of the calculator
calc1.keepInMemory(17);

// We'll do the same with the other instance with a different number
calc2.keepInMemory(4711);

// And then we print out the result
System.out.println("calc1 memory: " + calc1.readFromMemory());
System.out.println("calc2 memory: " + calc2.readFromMemory());
```

Output

```
calc1 memory: 17
calc2 memory: 4711
```


Fields are per-instance

Each instance of the class is independent.



Any instance of the type Calculator will always have a field named **memory**, as specified in the class declaration.

However, the **value** of that field can differ in each Calculator instance!

Access Modifiers

Public
Private

Public

Access modifiers are used to control what can access classes and class members.

The **public** modifier is the most permissive of the access modifiers.

Indicates that this class can be instantiated anywhere in your code, without restriction.

Indicates that this method can be reached outside of this class. In other words, you can call it from an instance variable of this type.

```
public class Greeter {
    public String greet() {
        return "Welcome!";
    }
}
```

Public

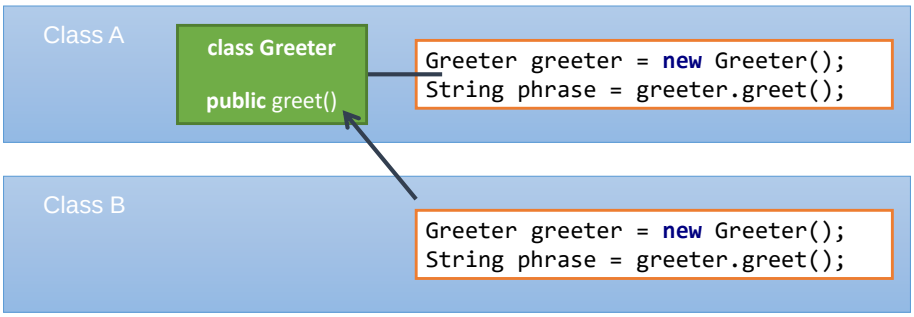
This allows us to call the method in the following way.

```
public class Greeter {
    public String greet() {
        return "Welcome!";
    }
}


Greeter greeter = new Greeter();

String phrase = greeter.greet();
// The string phrase contains "Welcome!"
```

Had the greet() method been declared as private, this would not compile!

Public

The **public** modifier allows anyone to consume this member from the outside.



Private

The **private** modifier is the least permissive.

Private members of a class cannot be accessed outside of the class.

Essentially the same example, but we are passing a field variable instead.

The field variable cannot be directly accessed outside the class

```
public class Greeter {
    private String phrase = "Welcome";

    public String greet() {
        return phrase;
    }
}
```

Private

Some methods are meant for internal use inside a class. These should be **private**.

```
public class MyDataType {
    // This method is externally exposed
    public boolean doStuff() {
        // Calling the internal function
        String internalStuff = doInternalStuff();

        // Do some work on the string we received and return a boolean
        return (internalStuff.length() > 5);
    }

    // This method is only accessible inside the class
    private String doInternalStuff() {
        // ... The code that goes here will return a string
    }
}
```

Private

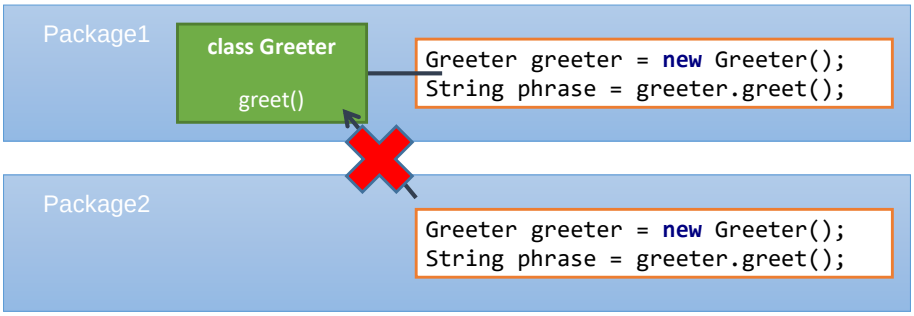
Private members can only be called from inside the class.

```
// Declare and instantiate
MyDataType myData = new MyDataType();

// This works because doStuff is public

myData.doStuff();

// This does not work because doInternalStuff is private
myData.doInternalStuff();
```

```
'doInternalStuff()' has private access in 'com.foo.Program'
```

No access modifier

If you leave off the access modifier, the member is as accessible as public inside the same package, but is inaccessible externally.




```
class MyClass {
    int Sum(int x, int y) {
        return x + y;
    }
}
```

Exercise 12

Lets do exercises 12