Object oriented programming - Access Modifiers module 4

Encapsulation is separating <u>what</u> a class does from <u>how</u> it does it.

Encapsulation

```java
public class RightTriangle_v1 {
  private double area;
  private double base;
  private double height;

  public RightTriangle_v1(double base,
                          double height) {
    if (base <= 0 || height <= 0) {
      this.base = 0;
      this.height = 0;
    } else {
      this.base = base;
      this.height = height;
      this.area = base * height / 2;
    }
  }

  public double getArea() {
    return this.area;
  }
}
```

```java
public class RightTriangle_v2 {
  private double base;
  private double height;

  public RightTriangle_v2(double base,
                          double height) {
    if (base <= 0 || height <= 0) {
      this.base = 0;
      this.height = 0;
    } else {
      this.base = base;
      this.height = height;
    }
  }

  public double getArea() {
    return this.base * this.height / 2;
  }
}
```

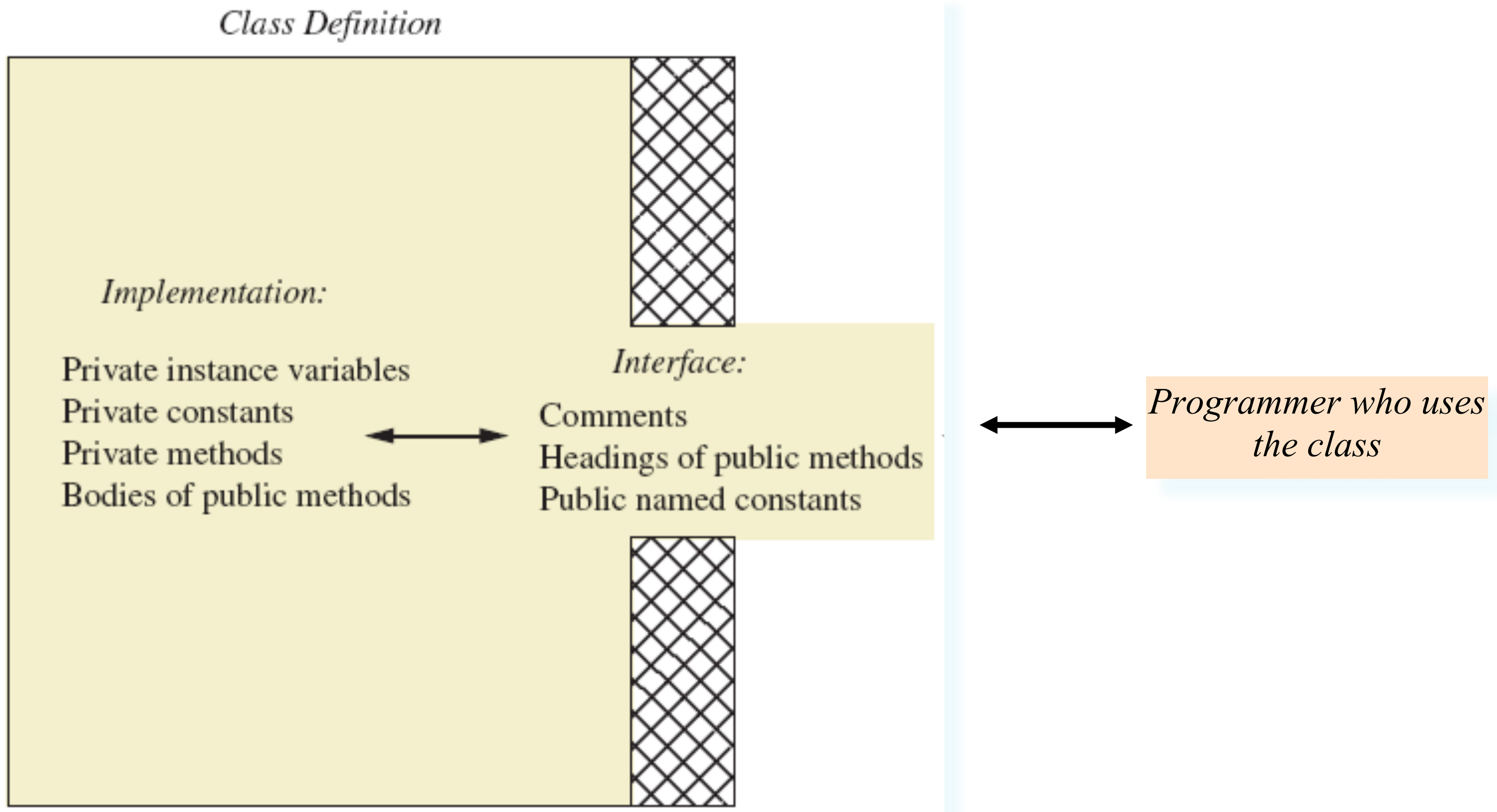
Encapsulación

```
public class EncapsulationDemo {

  public static void main(String[] args) {

    RightTriangle_v1 triangle1 = new RightTriangle_v1(10, 5);
    RightTriangle_v2 triangle2 = new RightTriangle_v2(10, 5);

    double out1 = triangle1.getArea();
    double out2 = triangle2. getArea();

    System.out.println("Class V1: " + out1);
    System.out.println("Class V2: " + out2);
  }
}
```

getArea() in both implementations produces the same results

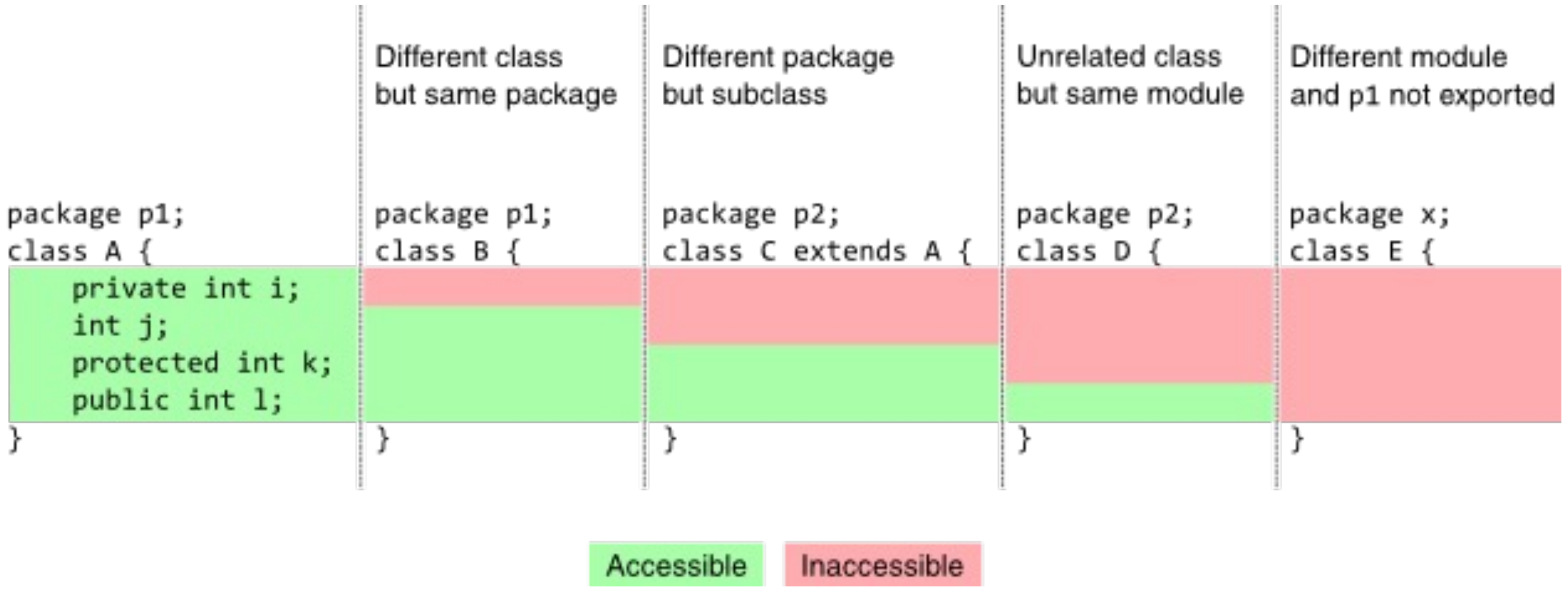
How do you achieve encapsulation?

1. Through access modifiers
2. Public methods to interact with the class


Class Definition
Implementation:
Private instance variables
Private constants
Private methods
Bodies of public methods
Interface:
Comments
Headings of public methods
Public named constants
Programmer who uses the class

Access Modifiers

An access modifier limits the scope of a variable, method or class. There are 4 access modifiers in java, but the most common are `private` and `public`.


Different class but same package
Different package but subclass
Unrelated class but same module
Different module and p1 not exported
Accessible
Inaccessible

```
package p1;
class A {
    private int i;
    int j;
    protected int k;
    public int l;
}
```

```
package p1;
class B {
}
```

```
package p2;
class C extends A {
}
```

```
package p2;
class D {
}
```

```
package x;
class E {
}
```

Getter

The getter method, also known as accessor, is a type of method that allows you to read the contents of an instance variable in a class.

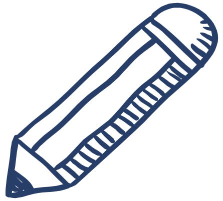
```
public class RightTriangle_v1 {
  //...
  //rest of the variables and methods
  //...
  private double height;

  public double getHeight() {
    return this.height;
  }
}
```

Note that **`height`** is a private variable, hence it is impossible to read its contents directly from outside the class. That is what the public `getHeight()` method is used for

Setter

Setter methods, also known as mutator, allows you to modify the contents of an instance variable in a class.

```
public class RightTriangle_v1 {
  //...
  //rest of the variables and methods
  //...
  private double height;
  private double area;

  public void setHeight(double height) {
    if (height > 0) {
      this.height = height;
      this.area = this.base * this.height / 2;
    }
  }
}
```



If `height` is modified, then we should modify the contents of variable `area` too.
This internal congruency can be achieved through the setter method `setHeight( )`.

It is a good practice to declare all instance variables as private, and use setter and getter methods to interact with them.

```java
public class RightTriangle_v1 {

  private double area;
  private double base;
  private double height;

  public RightTriangle_v1(double base,
                          double height) {
    if (base <= 0 || height <= 0) {
      this.setBase(0);
      this.setHeight(0);
    } else {
      this.setBase(base);
      this.setHeight(height);
    }
  }

  private void updateArea() {
    this.area = this.getBase() * this.getHeight() / 2;
  }

  public double getBase() {
    return this.base;
  }
```

```java
  public void setBase(double base) {
    if (base >= 0) {
      this.base = base;
      this.updateArea();
    }
  }

  public double getHeight() {
    return this.height;
  }

  public void setHeight(double height) {
    if (height >= 0) {
      this.height = height;
      this.updateArea();
    }
  }

  public double getArea() {
    return this.area;
  }
```

```java
private void updateArea() {
    this.area = this.getBase() * this.getHeight() / 2;
  }
```

Why is updateArea() a private method?

When we use instance variable **`area`**, we expect its contents to be updated and congruent with the rest of its attributes. The class code is responsible to guarantee that `updateArea()` is called every time one of its dimensions are changed.

Why don’t we create a `setArea()` method?

If the contents of `area` is modified, the class might loose internal congruency. The class should ensure that the area is always congruent with the following formula:

$$Area = \frac{base \times height}{2}$$

Area should only be updated when base and height are updated