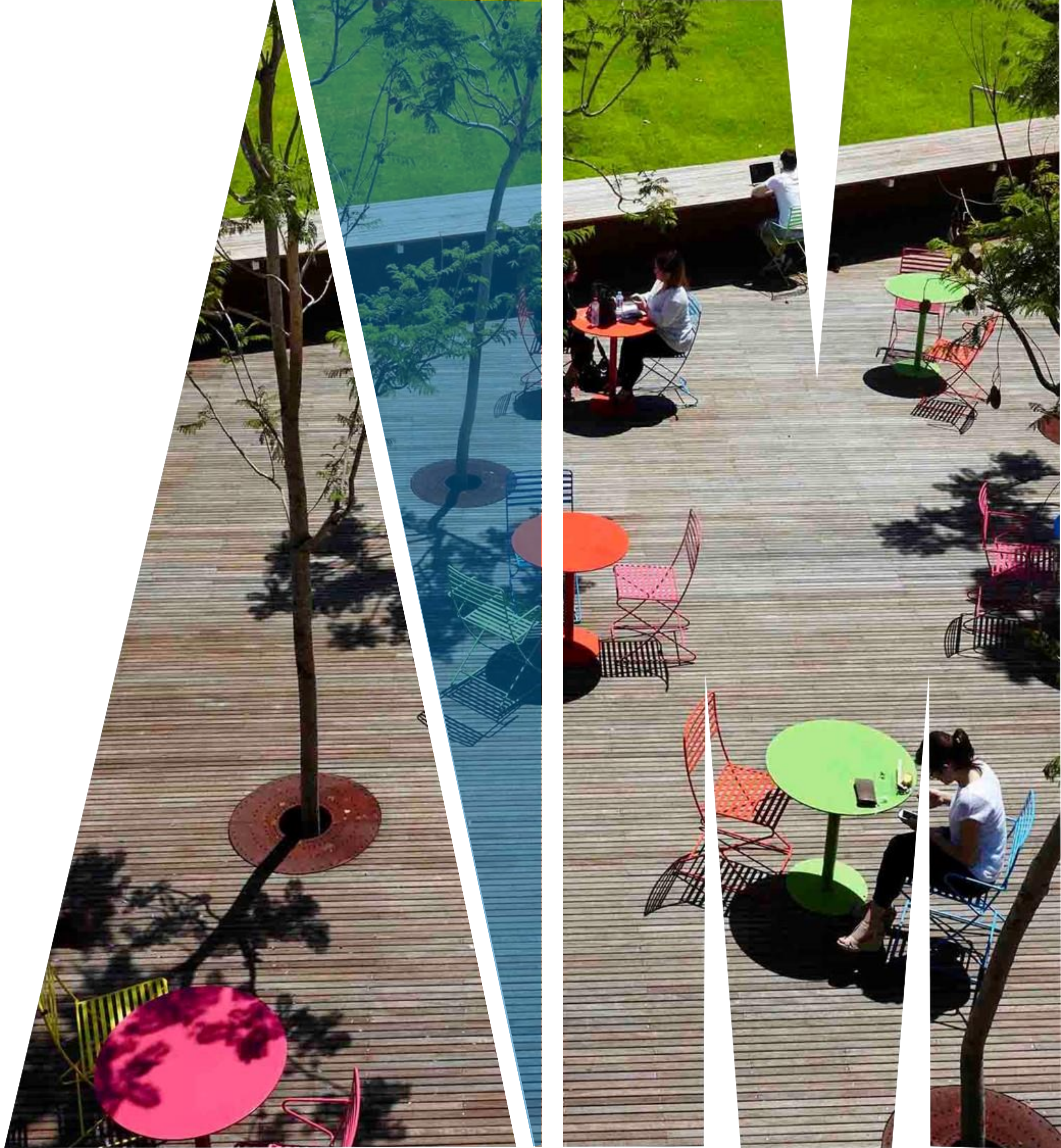
MONASH
University

FIT2099 Object-Oriented Design and Implementation

Defensive copying



MONASH
University

THE
DEFENSIVE COPYING

When getters return **a reference** to a private object it is said that this object is ***mutable***

i.e. with public attributes or mutator methods (setters) other than constructor

A **mutable object** is simply an object which can change its state after construction. For example, StringBuilder and Date are mutable objects, while String and Integer are immutable objects.

HOW A
PRIVACY LEAK OCCURS

1- If the mutable object is a private attribute (field) of another (native) class**, its state should be changed only by the native class**.

2- Then, a **defensive copy** of the mutable object must be made any time it's passed into (constructors and set methods) or out of (get methods) the class.

3- If this is not done, then it's simple for the caller to **break encapsulation**, by changing the state of an object which is simultaneously visible to both the class and its caller.

WHEN DOES A
PRIVACY LEAK OCCUR?

When getters return a reference to a private object that is *mutable*

i.e. with public attributes or mutator methods other than constructor

Generally, you should **make a copy and return that.**

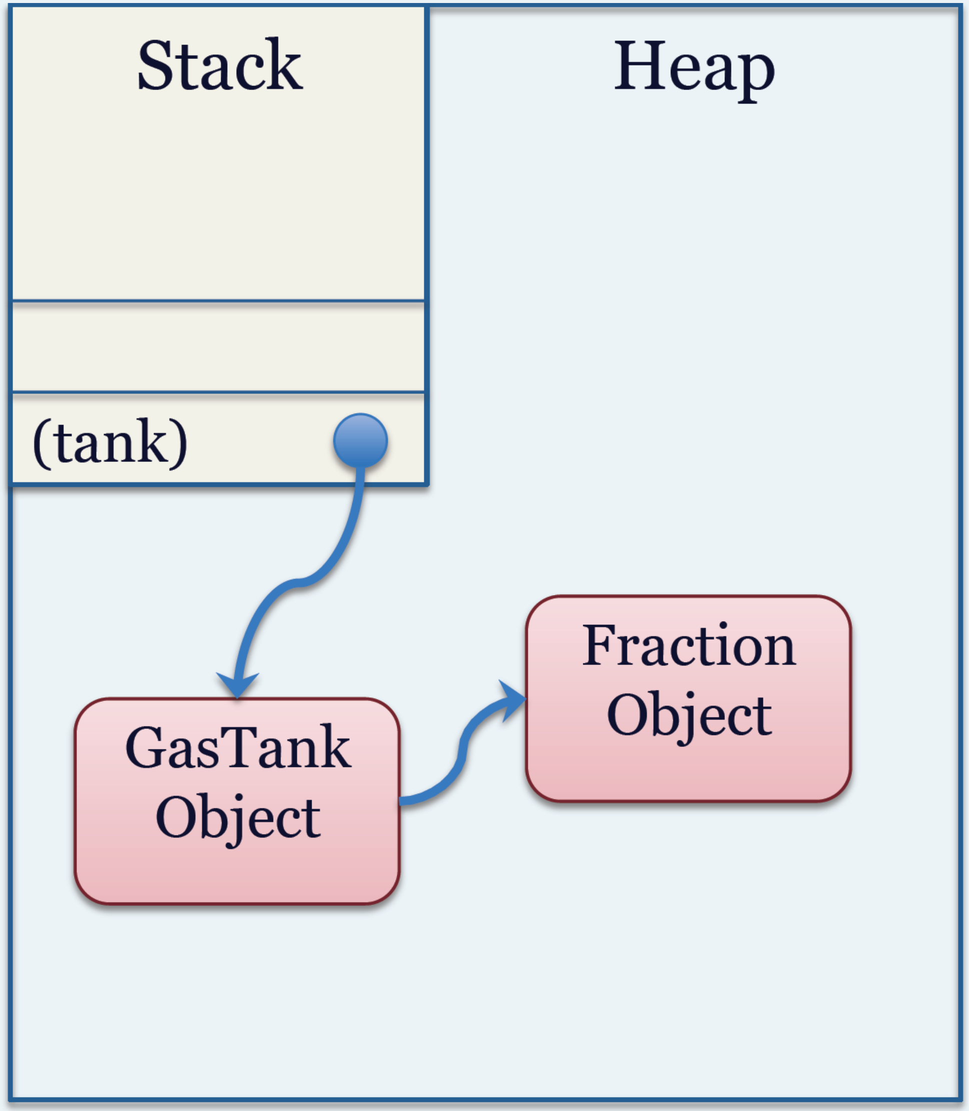
Otherwise, you lose benefit of encapsulation

this is called a ***privacy leak***

Lose control of connascence

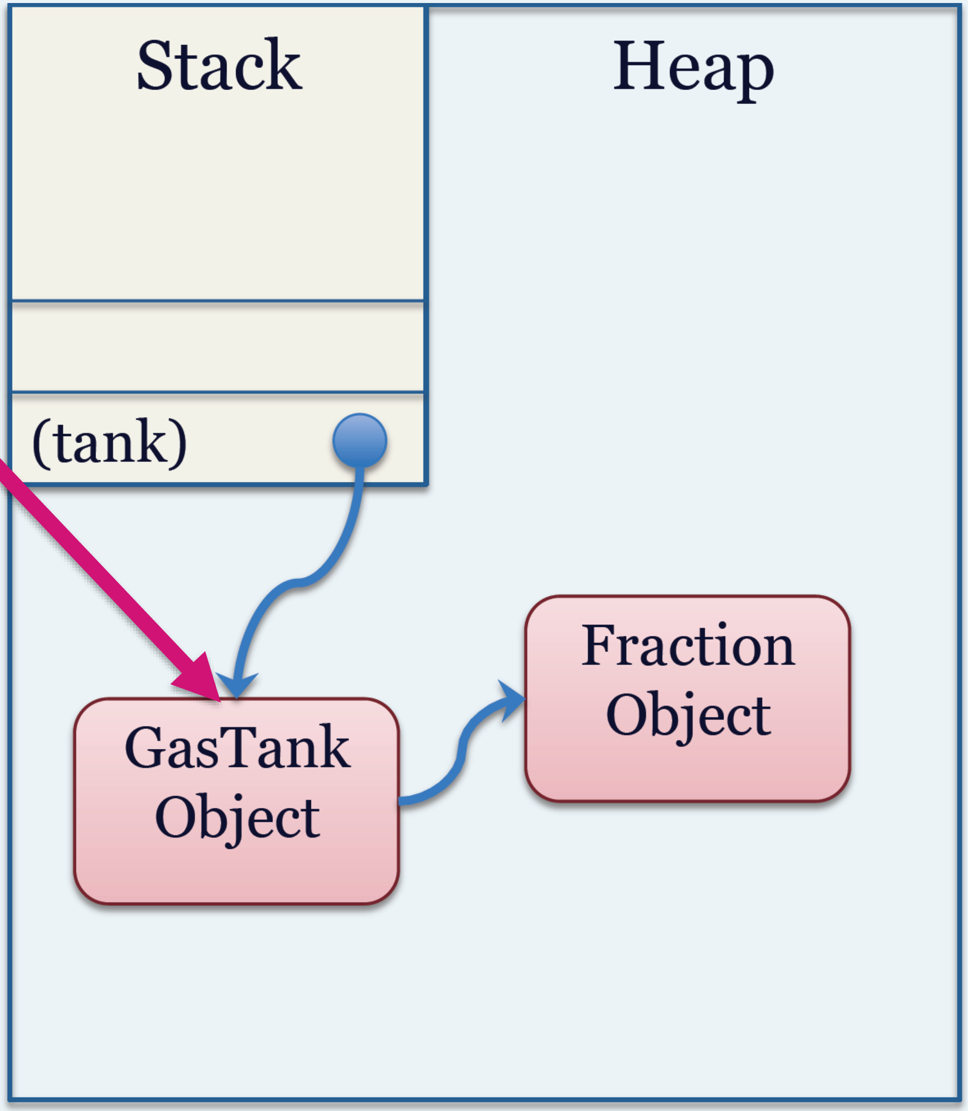
PRIVACY LEAK
EXAMPLE

```
Public class GasTank {
        private Fraction fuel;
        public GasTank() {
                fuel = new Fraction(1, 1);
        }



}
```



PRIVACY LEAK
EXAMPLE

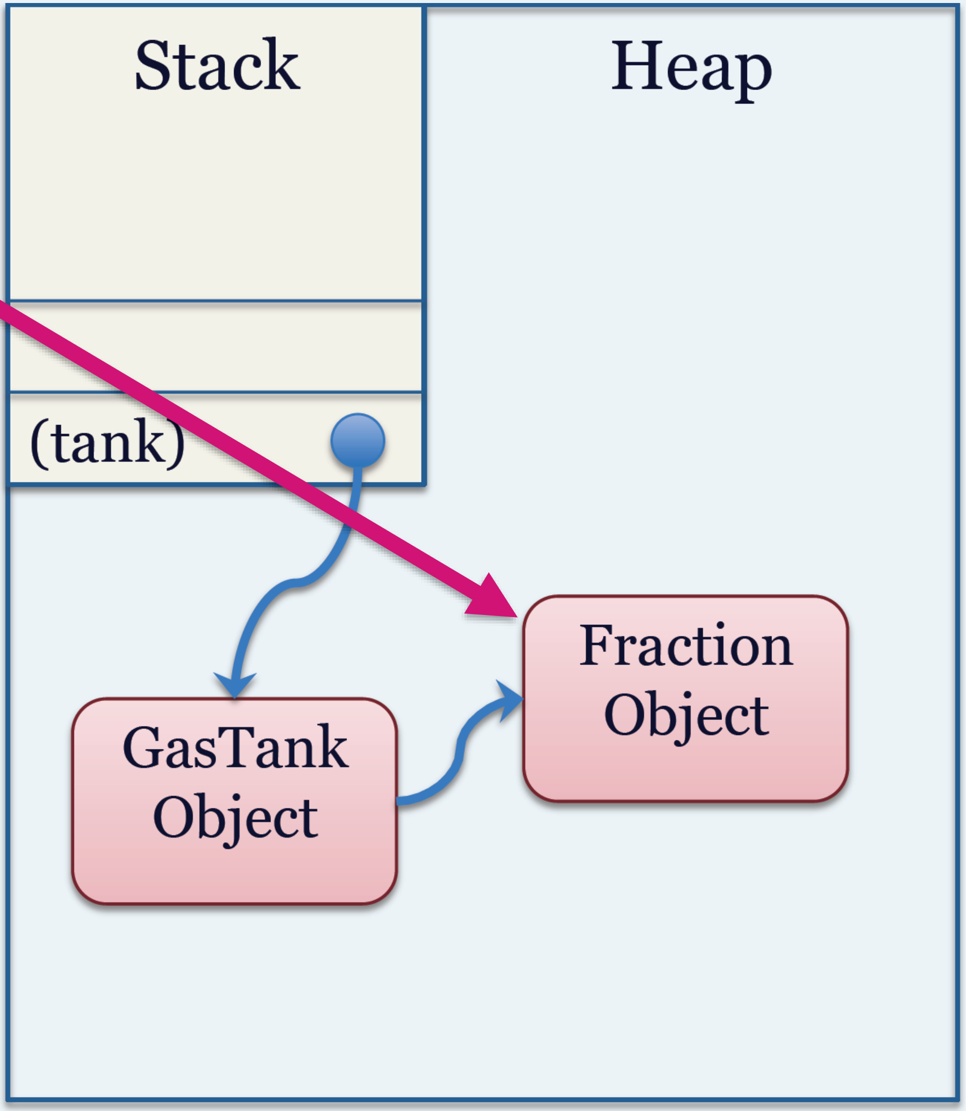
```
Public class GasTank {
        private Fraction fuel;
        public GasTank() {
                fuel = new Fraction(1, 1);
        }




}
```



PRIVACY LEAK
EXAMPLE

Private

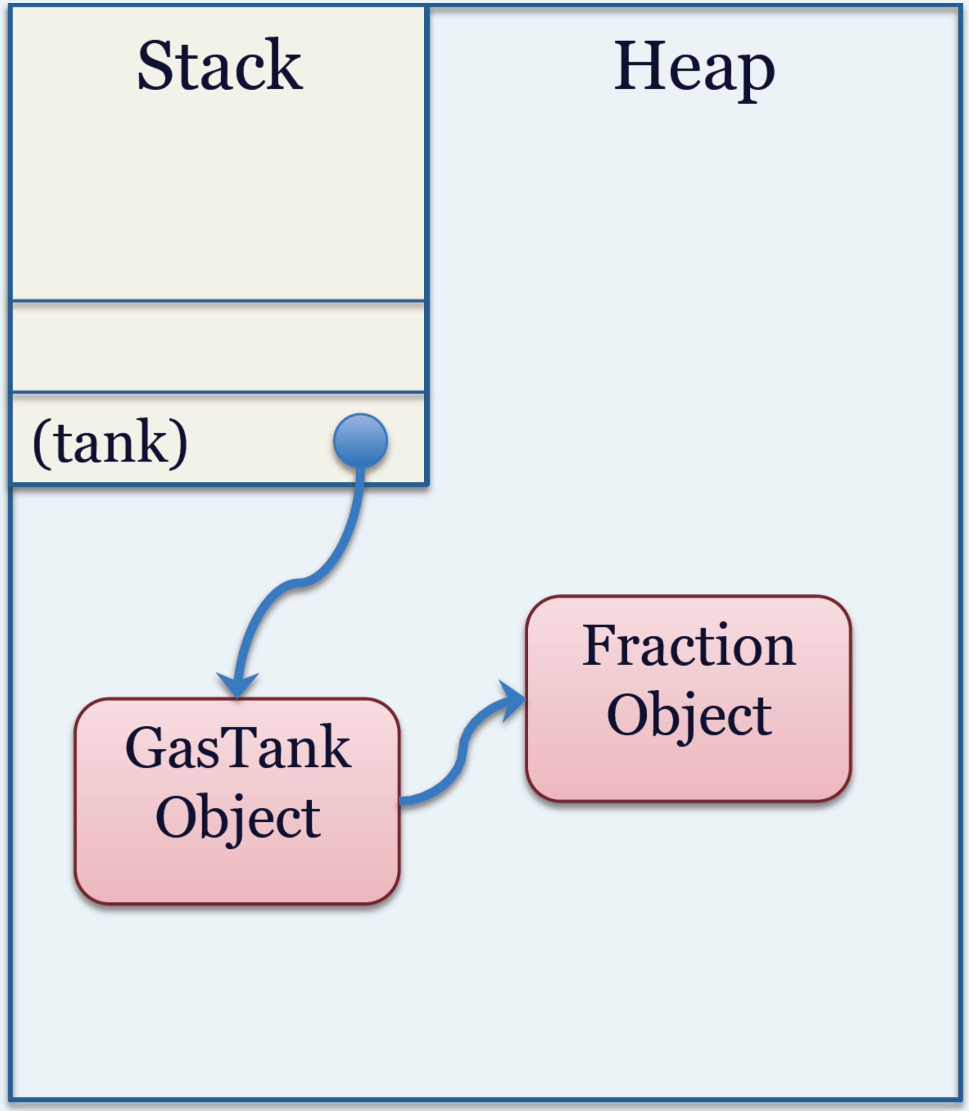
```
Public class GasTank {
        private Fraction fuel
        public GasTank() {
                fuel = new Fraction(1, 1);
        }



}
```



PRIVACY LEAK
EXAMPLE

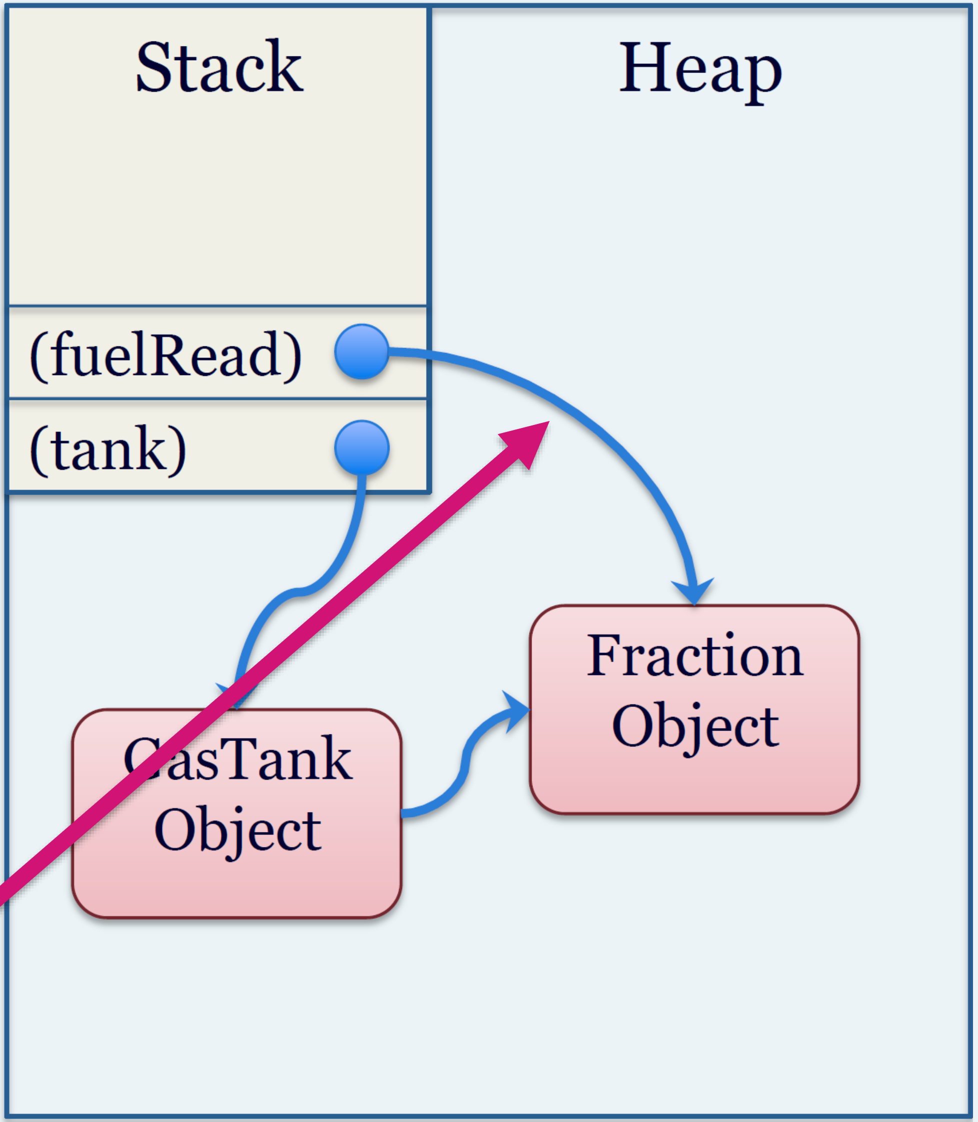
The getter returns a reference to the object

```
Public class GasTank {
        private Fraction fuel;
        public GasTank() {
                fuel = new Fraction(1, 1);
        }
        public Fraction getFuel() {
                return fuel;
        }
        public void setFuel(Fraction f) {
                if (f.asDouble <= 1)
                        fuel = f;
        }
}

// What does the following code do?
GasTank tank = new GasTank();
Fraction fuelRead = tank.getFuel();
fuelRead.setNumerator(2);
```



PRIVACY LEAK
EXAMPLE

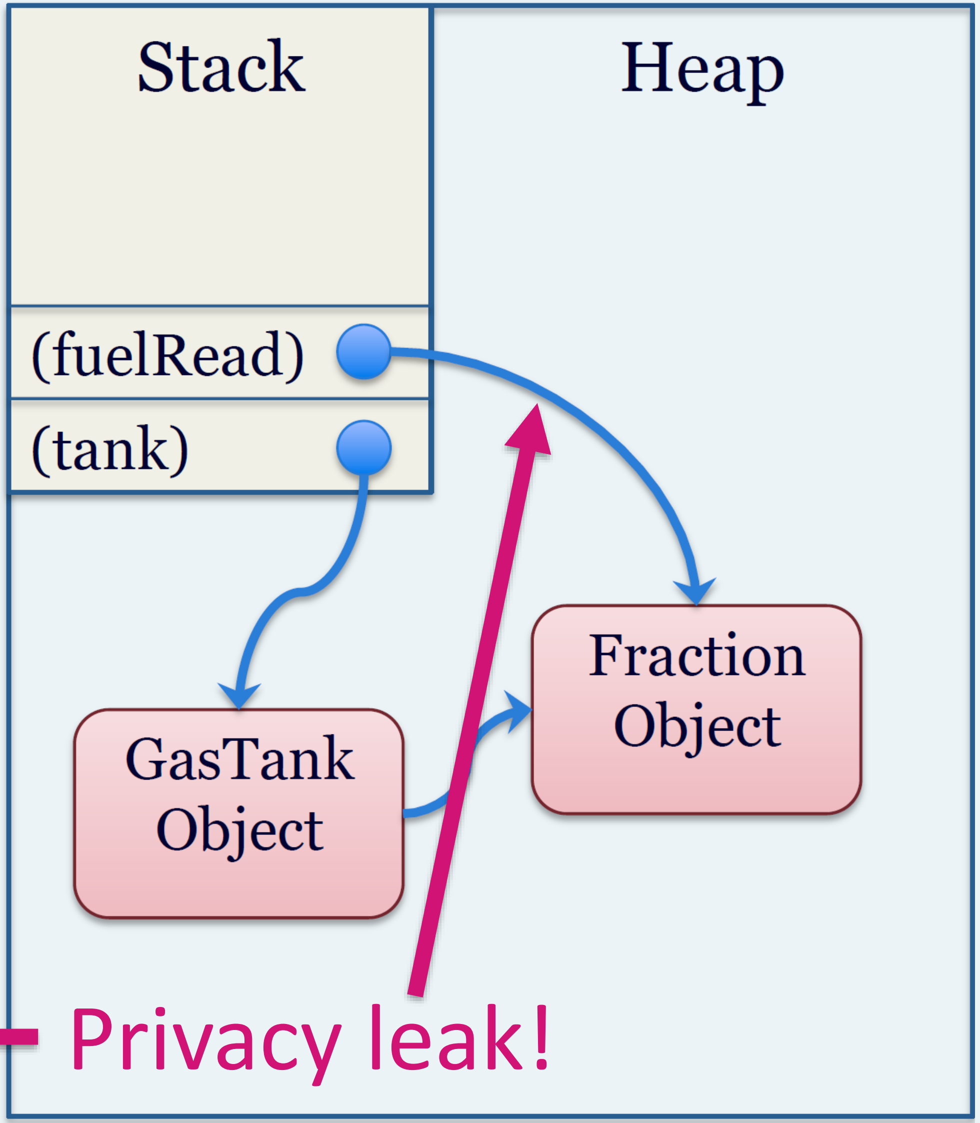
```
Public class GasTank {
    private Fraction fuel;
    public GasTank() {
        fuel = new Fraction(1, 1);
    }
    public Fraction getFuel() {
        return fuel;
    }
    public void setFuel(Fraction f) {
        if (f.asDouble <= 1)
            fuel = f;
    }
}

// What does the following code do?
GasTank tank = new GasTank();
Fraction fuelRead = tank.getFuel();
fuelRead.setNumerator(2);
```



PRIVACY LEAK
EXAMPLE

Private

```
Public class GasTank {
        private Fraction fuel;
        public GasTank() {
                fuel = new Fraction(1, 1);
        }
        public Fraction getFuel() {
                return fuel;
        }
        public void setFuel(Fraction f) {
                if (f.asDouble <= 1)
                        fuel = f;
        }
}

// What does the following code do?
GasTank tank = new GasTank();
Fraction fuelRead = tank.getFuel();
fuelRead.setNumerator(2);
```



Privacy leak!

THE SOLUTION VIA
DEFENSIVE COPYING

A privacy leak occurs when a private instance variable can be modified outside of its class

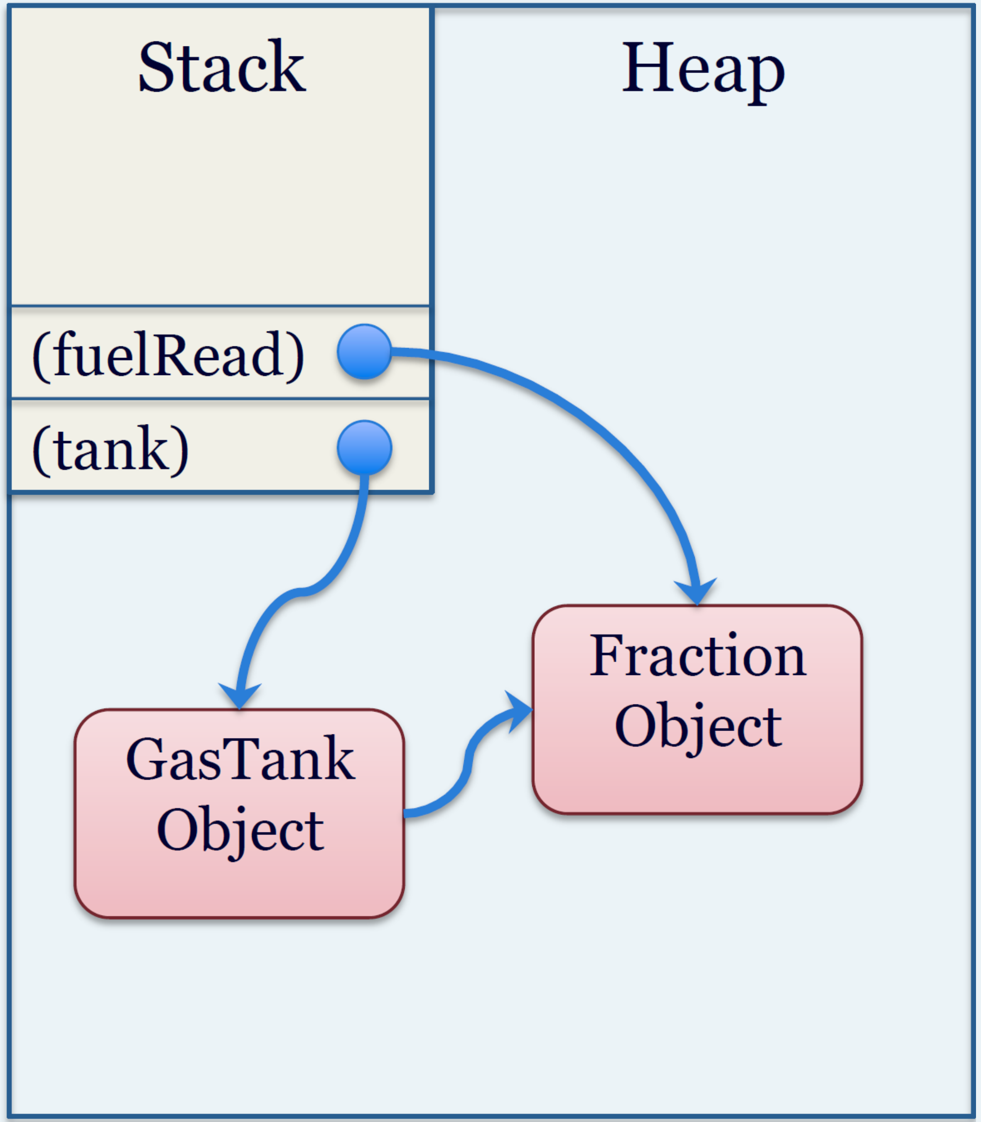
This happens because of aliasing, two references to the same object

What if we rewrite getFuel()?

```
public Fraction getFuel() {
        return new Fraction(fuel);
}
```

Returns a copy of fuel

Changes to this copy won't affect the original



THE SOLUTION VIA
DEFENSIVE COPYING

A privacy leak occurs when a private instance variable can be modified outside of its class

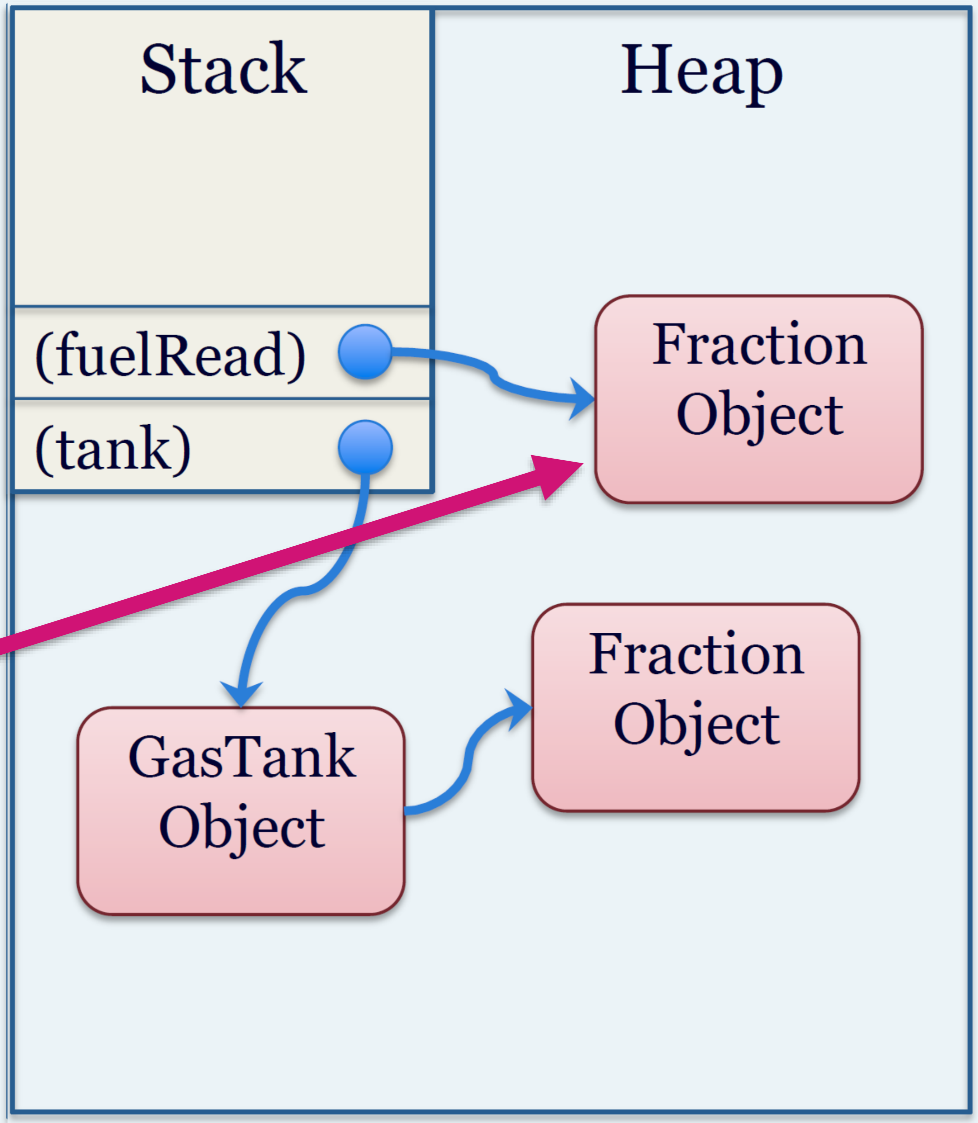
This happens because of aliasing, two references to the same object

What if we rewrite getFuel()?

```
public Fraction getFuel() {
        return new Fraction(fuel);
}
```

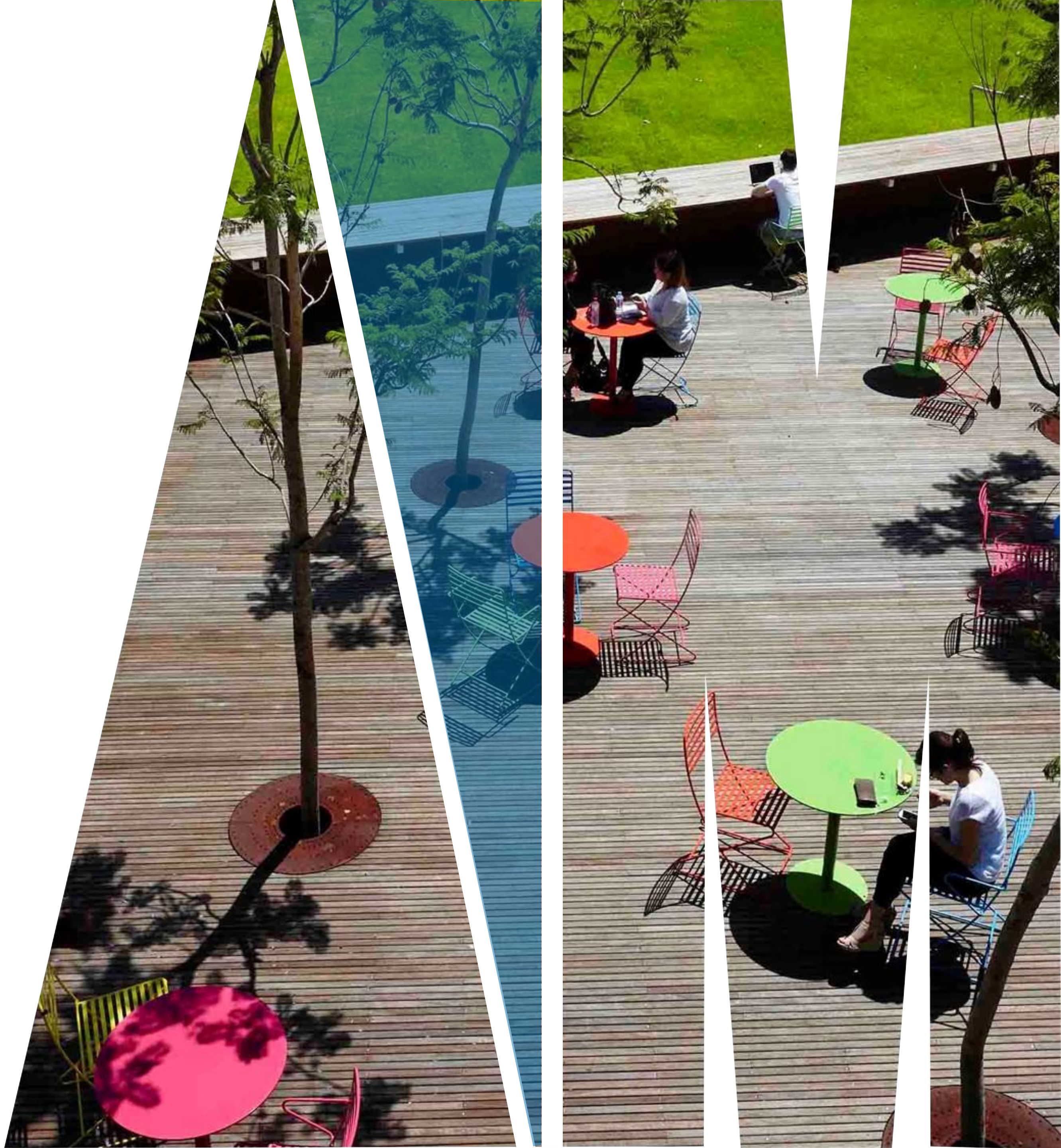
Returns a copy of fuel

Changes to this copy won't affect the original



MONASH University

Thanks



MONASH University