Object-Oriented Programming

Ehsan Noei, PhD

e.noei@queensu.ca

- Slides
- Codes
- Assignments
- Supplementary materials

Are available before the class.

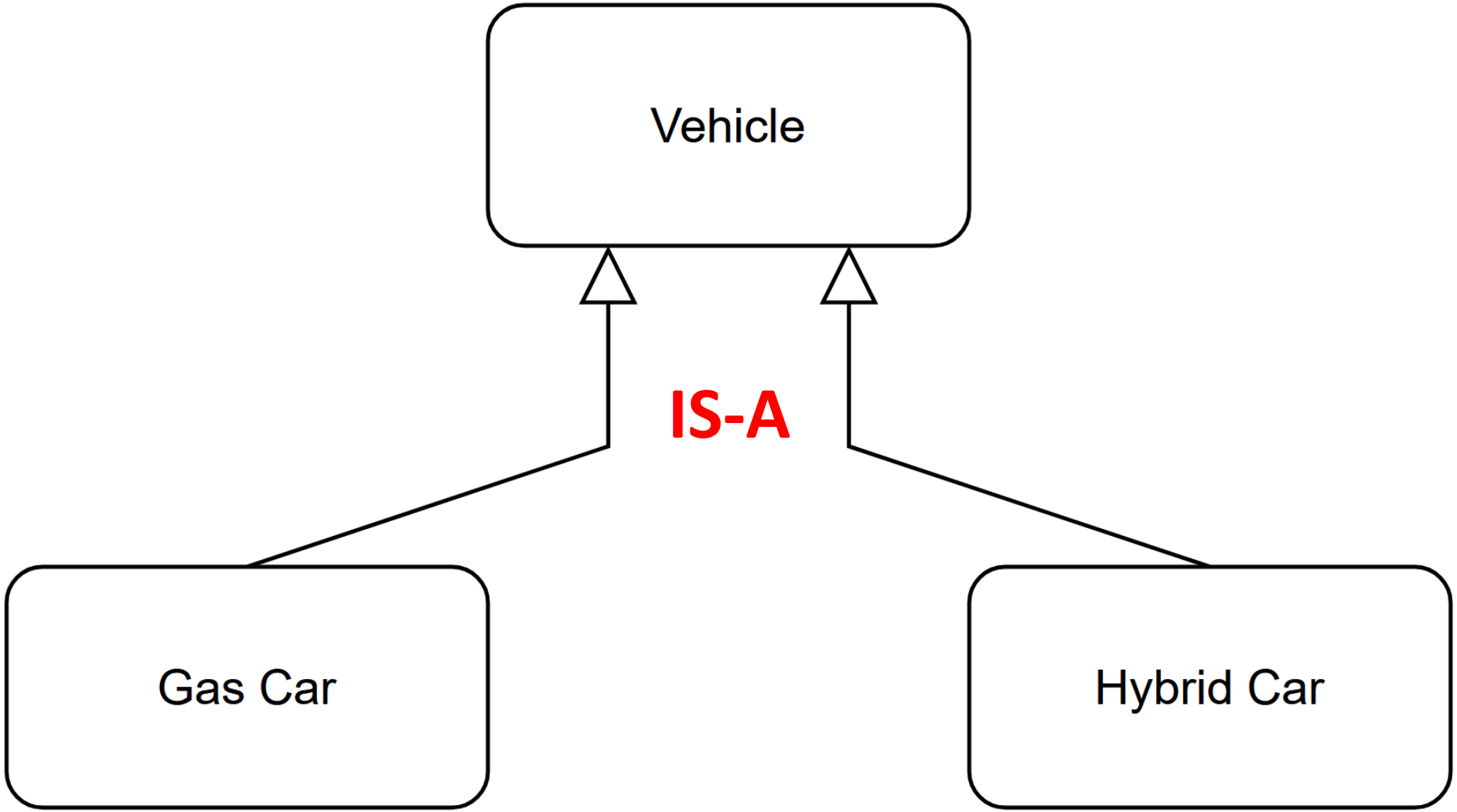
Four Pillars of OOP

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

Subtype Polymorphism

Subtype Polymorphism

- Subtype polymorphism allows a function to operate on objects of different classes as long as they share a common superclass.


Vehicle
IS-A
Gas Car
Hybrid Car

```
public class Vehicle {
    protected Integer fuel;
    public Vehicle(Integer fuel) {
        this.fuel = fuel;
    }
    public void move(Integer fuel) {
        this.fuel -= fuel;
    }
}
```

```
public class GasCar extends Vehicle {
    public GasCar (Integer fuel) {
        super(fuel);
    }
}
```

```
public class HybridCar extends Vehicle {

    public HybridCar (Integer fuel) {
        super(fuel);
    }

}
```

```
Vehicle gasCar = new GasCar(100);
gasCar.move(10);

Vehicle hybridCar = new HybridCar(100);
hybridCar.move(5);
```

Data vs. Behaviour

```
public class Vehicle {
    protected Integer fuel;
    public Vehicle(Integer fuel) {
        this.fuel = fuel;
    }
    public void move(Integer fuel) {
        this.fuel -= fuel;
    }
}
```

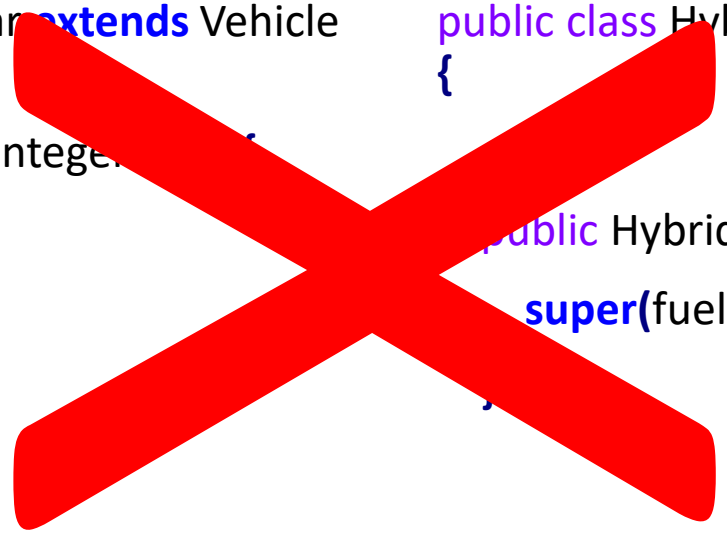
```
public class GasCar extends Vehicle
{
    public GasCar (Integer
        super(fuel);
    }

}
```

```
public class HybridCar extends Vehicle
{

    public HybridCar (Integer fuel) {
        super(fuel);


}
```



```
Vehicle gasCar = new Vehicle(100);
gasCar.move(10);

Vehicle hybridCar = new Vehicle(100);
hybridCar.move(5);
```

```
public class Vehicle {
    protected Integer fuel;
    public Vehicle(Integer fuel) {
        this.fuel = fuel;
    }
    public void move(Integer fuel) {

    }
}
```

```
public class GasCar extends Vehicle {
    public GasCar (Integer fuel) {
        super(fuel);
    }
@Override
public void move (Integer fuel) {
        this.fuel -= fuel;
        pushGas();
    }
}
```

```
public class HybridCar extends Vehicle {
    private  Integer electric;
    public HybridCar (Integer fuel) {
        super(fuel);
    }
```

```
@Override public void move (Integer fuel) {
        this.fuel -= fuel;
        this. electric -= fuel / 10;
        if(this.speed < 80 &&
            this.electric > 10) eVMode();
        else pushGas();
    }
}
```

```
Vehicle gasCar = new GasCar(100);
gasCar.move(10);

Vehicle hybridCar = new HybridCar(100);
hybridCar.move(5);
```

Summary

- Subtype polymorphism allows a function to operate on objects of different classes as long as they share a common superclass.
- Subtype polymorphism
  - Reuse of pieces of code for hierarchical objects
  - Differences in behaviours

Next Session

- More examples of subtype polymorphism
- Another kind of polymorphism