MET CS665 – Software Design and Patterns

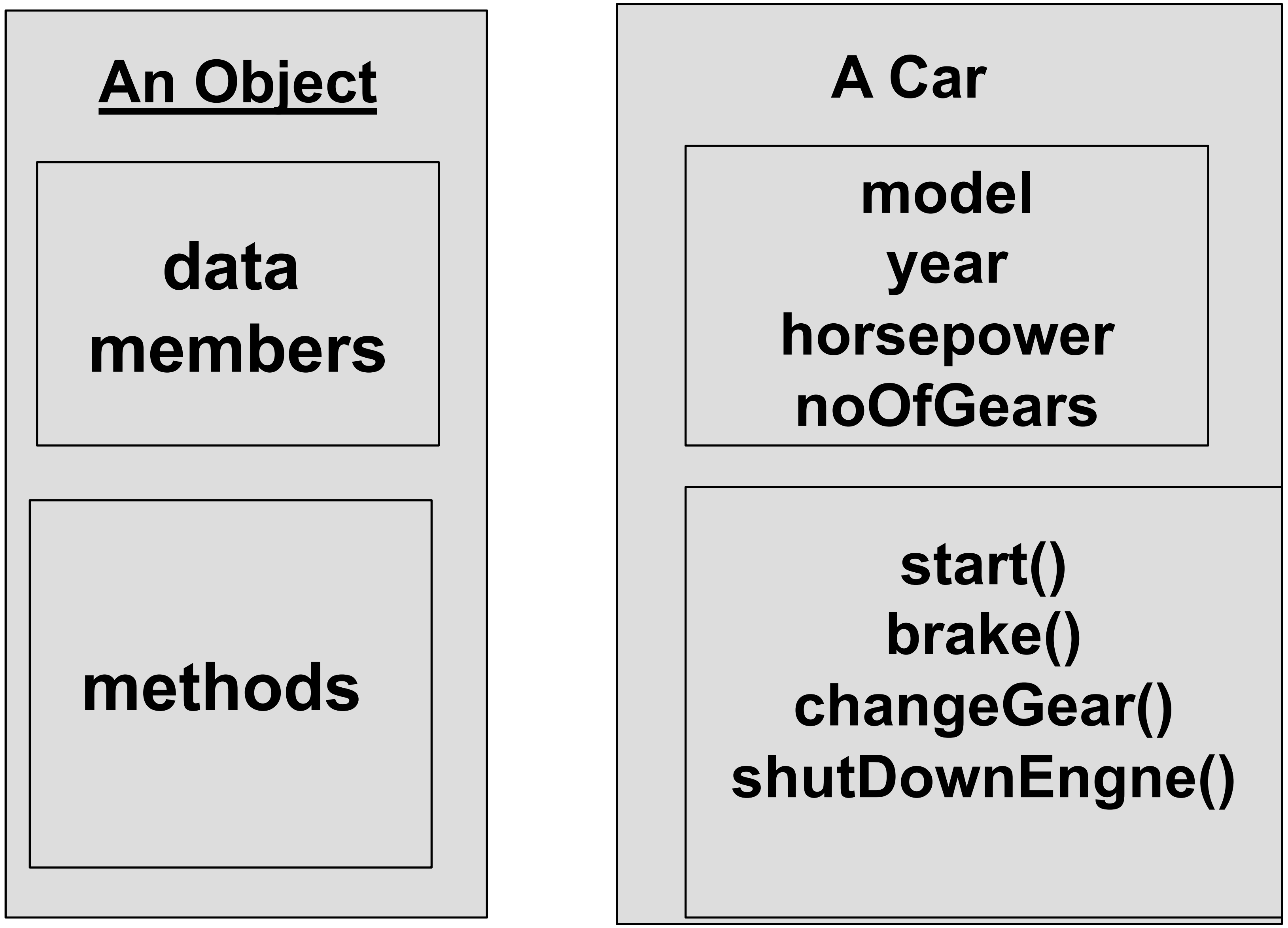
Object-Oriented Features

Object-Oriented Programming (OOP)

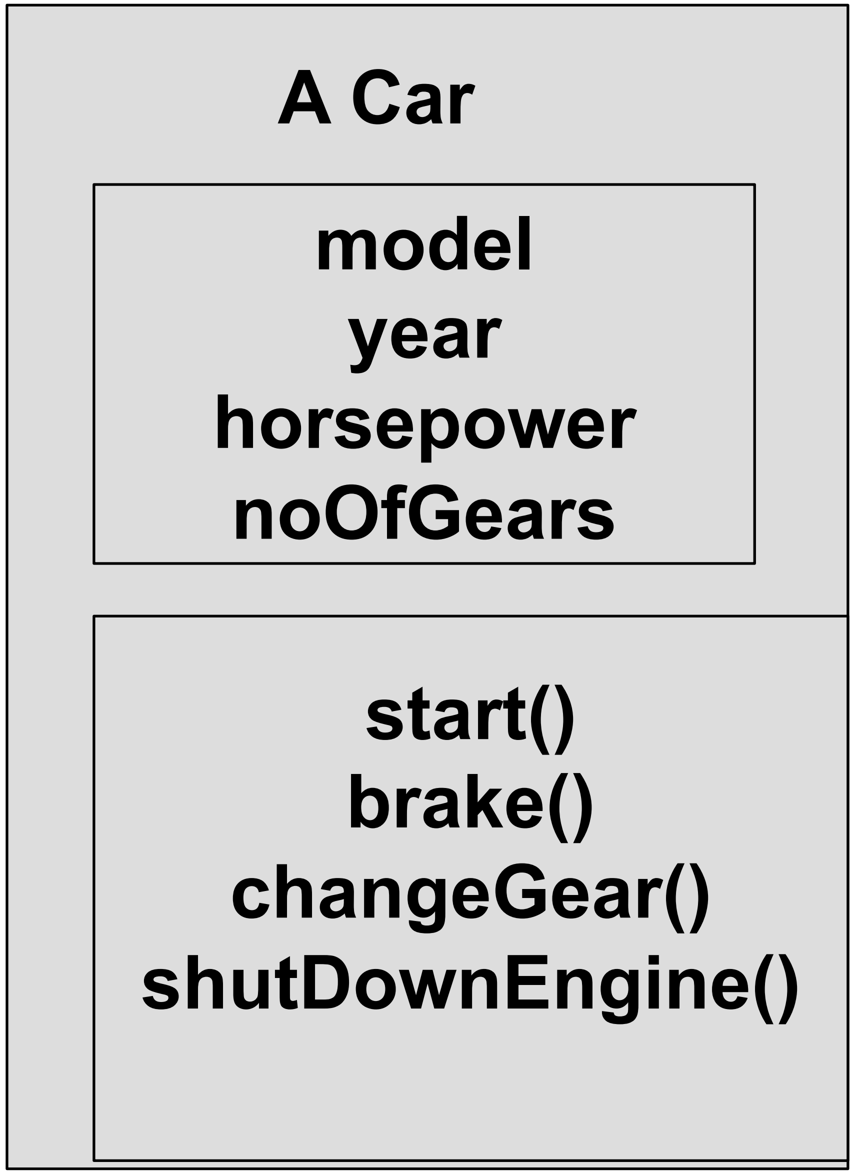
- OOP is a programming technique organized based on using objects to design and develop applications.

- OOP combines data and computation for processing the data into encapsulated objects.

Classes and Objects



An Object

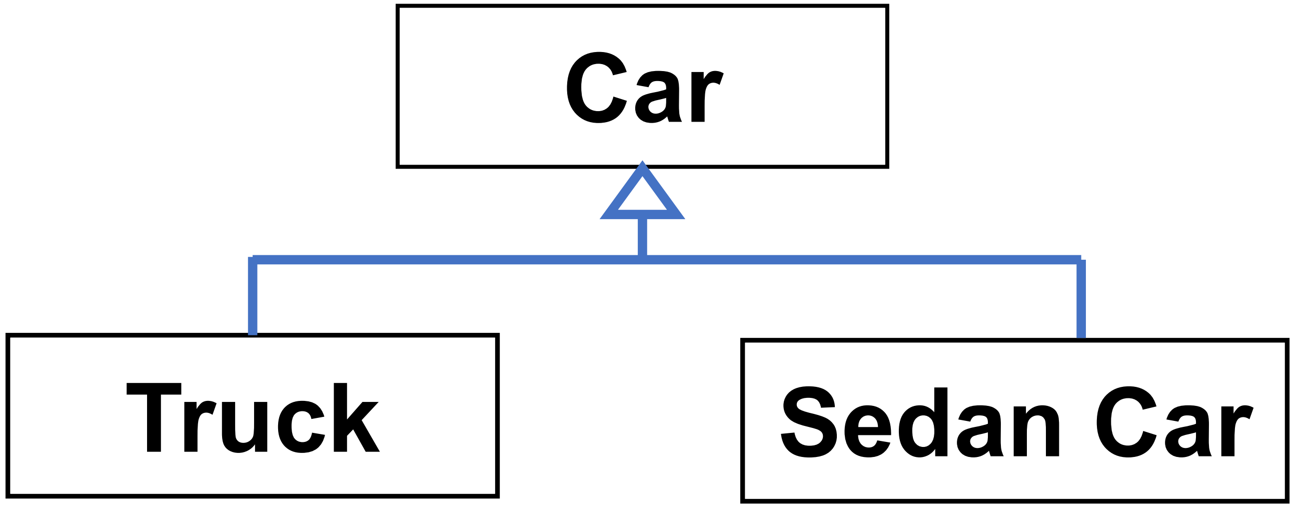


```
Car_No123

model="Toyota Corrola"
year=2010
horsepower=132
noOfGears=4

start() {

Steps to start the engine …

}
```

Object-Oriented Features

- Inheritance
- Polymorphism
- Encapsulation
- Abstraction

Inheritance

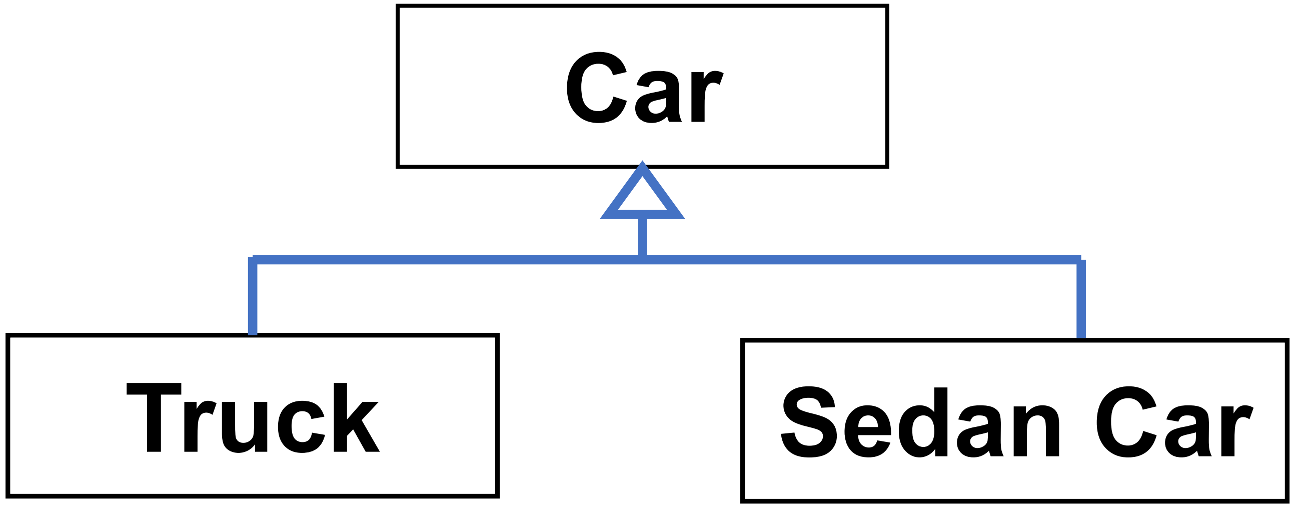


- Is-A-Type-Of Relation

A Truck is type of a Car

A Sedan is a type of a Car

Inheritance

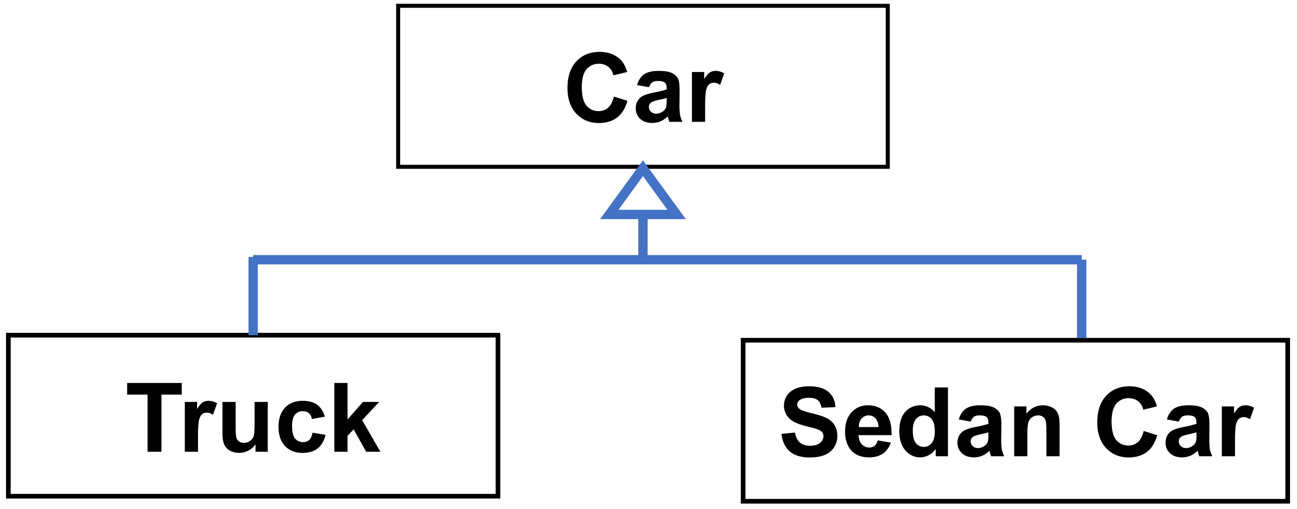


- Is-A-Type-Of Relation

A Truck is type of a Car

A Sedan is a type of a Car

Inheritance



```
public class Truck extends Car {
    ...
}
public class SedanCar extends Car {
    ...
}
```

Polymorphism

- Closely related to Inheritance
- Multiple forms/types of the same class/objects
- Dynamic Binding

```
Car myToyota=new Sedan();
Shop myCarShop =new Shop();

myShop.repair(myToyota);
```

Encapsulation

- Goal is to bind the data with the computation that manipulates it.
- Restrict the access to Object’s data from external interference.
- We can control and check the input values

Encapsulation

```
public class Car {

private int year;
  private int noOfGears;

public void setYear(int year) {
// we set the input only if it is correct. Year of a car must be between 1885 and 2018.  Benz ran his first car in
1885, Daimler in 1886.
if (year > 1885 || year < 2018)
          this.year = year;
else
          // If the check failed we log it and send a message.
          System.out.println("Year must be between 1900 and 2018");
}

public int getYear() {
          return year;
}

public void setNoOfGears(int noOfGears) {
          // Input for the number of gears must be correct.
          if (noOfGears < 14)
                    this.noOfGears = noOfGears;
          else
          System.out.println("No of gears of a car can not be larger than 14");
}

public int getNoOfGears() {
          return noOfGears;
}
}
```

Abstraction

- Hiding the implementation complexity

- Offering computation services by providing over Application Programing Interfaces (API)

Abstraction - Example

```
public class Carshop {
// This method repairs your car if pay enough.
public boolean repair(Car yourCar, double money){
        . . .
   }
}
```

```
public class Main {

public static void main(String[] args) {
       Car myCar = new Sedan();
       CarShop myFavoriteCarshop =new CarShop();

       if(myCar.isDamaged && iHaveMoney) {
          myFavoriteCarshop.repair(myCar, money);
         }
    }
}
```

Object-Oriented Features

- Inheritance
- Polymorphism
- Encapsulation
- Abstraction