Object-Oriented Programming: Interfaces

IST 256

Application Programming for Information Systems

Interfaces

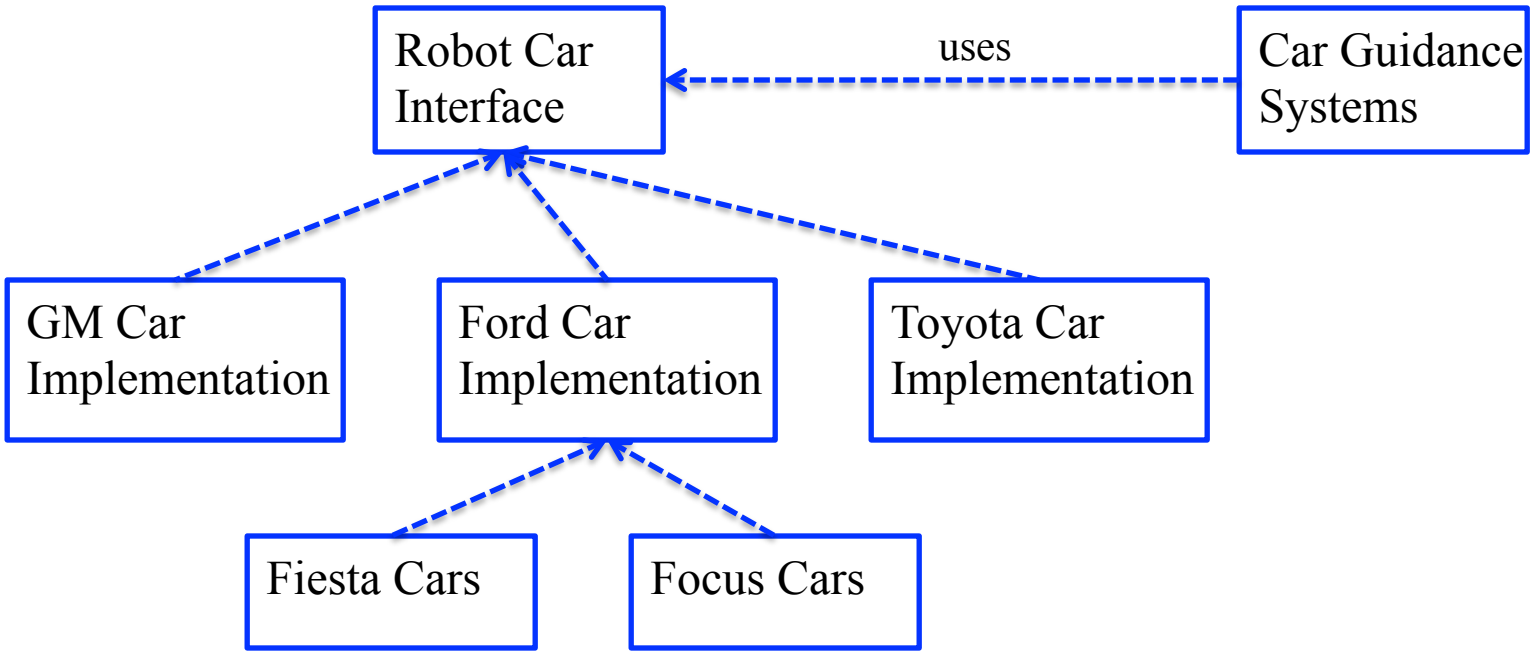
- A special type of class used in Object Oriented Programming is the interface.
- An interface has declarations for
  - Fields
  - Methods, but only the headers and no actual method bodies
    - These are called *abstract* methods
- A class can implement an interface by providing the actual method bodies
- The interface functions as a kind of contract, because every class that implements it is guaranteed to have those methods

Example from the Java Tutorial

- Suppose that in the future, guidance companies want to write software for robotic cars that drive themselves
- There should be a standard automotive interface that spells out in detail the commands that make the car move: start, stop, turn, accelerate, etc.
  - In Java, this would be an interface with abstract method
- The guidance companies could count on using these standard methods to control the car, and they could get input from GPS satellites and sensor data to write guidance algorithms
- The automotive companies would implement these methods to make their brand of car respond to the commands
  - e.g. GM would have their own (proprietary) software to make their cars move.

Example Java Hierarchy

- The Interface is the parent class of all the class that provide implementations of the interface.
  - Implementation hierarchy functions in the same way as the inheritance hierarchy for parent and child classes.



How to Write an Interface

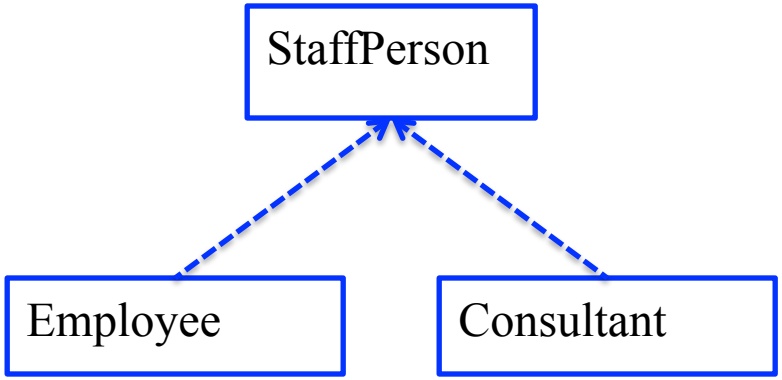
- Use the interface keyword in the place of class
- Provide the abstract methods by giving method headers,
  - also known as method signatures
- No fields are needed because no instances are ever created

```
public interface StaffPerson
{
  // constant declarations, if any

  // method signatures
  String getName();
  void raiseSalary(double rate);

  // more method signatures
  ......

}
```



Employee Class can implement the interface

- Use the key word implements
- Protected fields to hold data
- Provide methods for the signatures in the interface

```
public class Employee implements StaffPerson
{       // employee fields (visible to subclasses)
        protected String name;
        protected double salary;

        // constructor with initial values
        public Employee(String n, double s)
        {       name = n;
                salary = s;
        }

        // methods of implementation
        public String getName()
        {   return name;    }
        public void raiseSalary(double percent)
        {   salary = … ;  }
        . . .
}
```