CSCI 132:
Basic Data Structures and Algorithms

Interfaces

Reese Pearsall

Fall 2023

Announcements

Program 1 due on September 18th @ 11:59 PM

Lab 3 due on Thursday @ 11:59 PM

Wednesday is Rubber Duck day!!

- You can also grab a rubber duck from my office

Interfaces are abstract classes that only contain methods with no body

```
public interface Vehicle {
        void accelerate(int a);
        void slowdown(int a);
        void refuel(int a);
}
```

Accelerate, Slow down, and refuel are all common behavior that all vehicles will have

However, the specifics of *how* they accelerate, slow down, refuel will be different between vehicles (ie the body of the methods will be slightly different)

Interfaces can be used to specify what a class *must do*, but not *how*

Interfaces are abstract classes that only contain methods with no body

```java
public interface Vehicle {
        void accelerate(int a);
        void slowdown(int a);
        void refuel(int a);
}
```

```java
public class Ferrari implements Vehicle {
}
```

For a Java class to use an interface, it must use the `implements` keyword

We can implement multiple interfaces (unlike inheritance)

Interfaces are abstract classes that only contain methods with no body

```java
public interface Vehicle {
        void accelerate(int a);
        void slowdown(int a);
        void refuel(int a);
}
```

```java
public class Ferrari implements Vehicle {

@Override
public void accelerate(int a) { ✓
        …
}
@Override
public void slowdown(int a) { ✓
        …
}
@Override
public void refuel(int a) { ✓
}
        …
}
```

Now, any Class that also has the behavior of `accelerating`, `slowdown`, and `refuel` can implement our interface, and those classes are **forced** to write the body of the methods

The code of the method body is omitted, but that is where the programmer can put the specific behavior of:

- how a Ferrari will accelerate
- how a Ferrari will slow down
- how a Ferrari will refuel

Interfaces are abstract classes that only contain methods with no body

```java
public interface Vehicle {
        void accelerate(int a);
        void slowdown(int a);
        void refuel(int a);
}
```

You can not create an instance of an interface

In the interface, the method bodies must be empty

- (Remember, the classes that *use* our interface will have the method bodies)

```java
public class Ferrari implements Vehicle {

@Override
public void accelerate(int a) { ✓
        …
}
@Override
public void slowdown(int a) { ✓
        …
}
@Override
public void refuel(int a) { ✓
}
        …
}
```

The code of the method body is omitted, but that is where the programmer can put the specific behavior of:

- how a Ferrari will accelerate
- how a Ferrari will slow down
- how a Ferrari will refuel

Interfaces are abstract classes that only contain methods with no body

Why use interfaces?

Interfaces are great when you need **multiple implementations** of the **same behavior**

It forces classes to implement X methods that might not logically belong to them *(more control)*

It provides **abstraction**
(ie the details of how things are implemented are not revealed in an interface)