# CSCI 132:
# Basic Data Structures and Algorithms

Interfaces

Reese Pearsall

Spring 2024

# Announcements

Program 1 due on February 16th @ 11:59 PM

Friday and Monday is Rubber Duck day!!

- You can also grab a rubber duck from my office

**Inheritance** is a mechanism that allows a *parent* class to pass on public and protected instance fields and methods to a *child* class

```
public class Programmer extends Employee {
}
```

Inheritance is great when you have **shared behaviors and attributes** across classes with the **same implementation**

Programmer

```
public String getName() {
    return this.name;
}
```

Salesperson

```
public String getName() {
    return this.name;
}
```

Lawyer

```
public String getName() {
    return this.name;
}
```

Same code (same implementation!!!)

**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

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

```
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

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

```
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

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

# **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);
}
```

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)

```
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 **shared behavior** with **different implementations**

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)

Given a side length **a**

| **Shape** | **Perimeter Formula** | **Area Formula** |
|---|---|---|
| Square | a * 4 | $a^2$ |
| Equilateral Triangle | a * 3 | $\frac{\sqrt{3}}{4}a^2$ |
| Regular Pentagon | a * 5 | $\frac{1}{4}\sqrt{5(5+2\sqrt{5})}\,a^2$ |
| Regular Hexagon | a * 6 | $\frac{3\sqrt{3}}{2}a^2$ |

Given a side length **a**

| **Shape** | **Perimeter Formula** | **Area Formula** |
|---|---|---|
| Square | a * 4 | $a^2$ |
| Equilateral Triangle | a * 3 | $\frac{\sqrt{3}}{4}a^2$ |
| Regular Pentagon | a * 5 | $\frac{1}{4}\sqrt{5(5+2\sqrt{5})}\,a^2$ |
| Regular Hexagon | a * 6 | $\frac{3\sqrt{3}}{2}a^2$ |

Shared behaviors with different implementations