# Using Static Members

**Jim Wilson**
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim  jwhh.com

# Overview

- Static member overview
- Static fields
- Static methods
- Static import statement
- Static initialization blocks

# Static Members

**Static members are shared class-wide**

- Not associated with individual instance

**Declared using the static keyword**

- Accessible using the class name

# Static Members

**Field**

A value not associated with
a specific instance

All instances access the same value

Flight.java

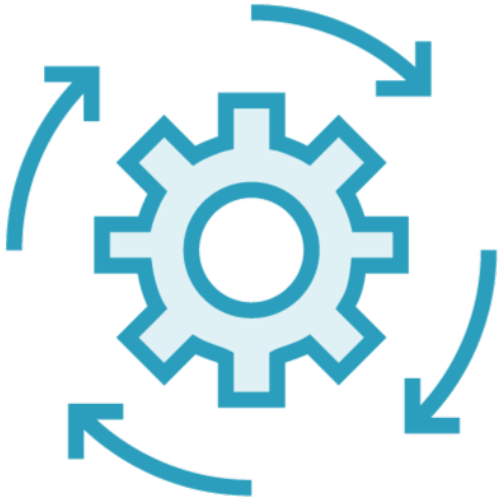
```
public class Flight {
  private int passengers, seats = 150;
  private static int allPassengers;
  public void add1Passenger() {
    if(passengers < seats) {
      passengers += 1;
      allPassengers += 1;
    }
  }
  // other members elided
}
```

# Static Members

## Field

A value not associated with a specific instance

All instances access the same value



## Method

Performs an action not tied to a specific instance

Has access to static members only

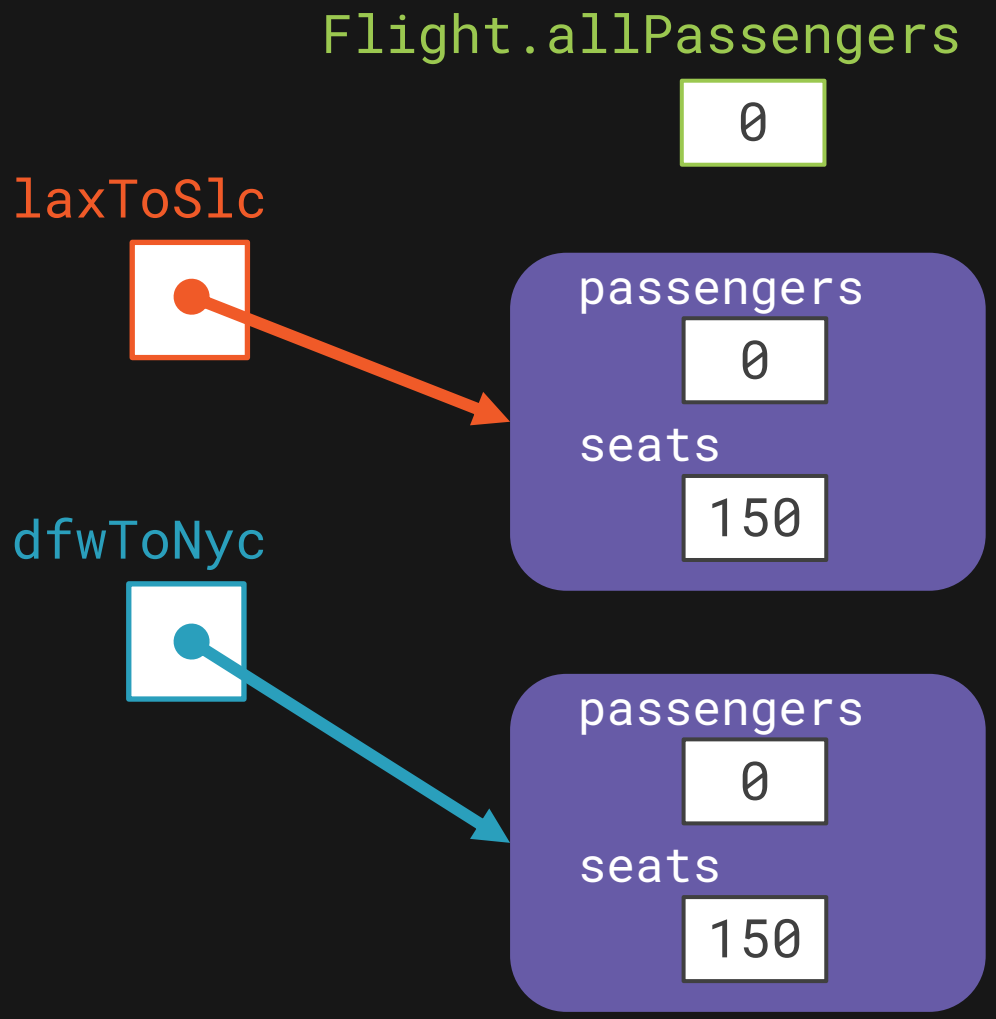
Flight.java

```
public class Flight {
  private int passengers, seats = 150;
  private static int allPassengers;
  public static int getAllPassengers() {
    return allPassengers;
  }
  public static void resetAllPassengers() {
    allPassengers = 0;
  }
  // other members elided
}
```

Main.java

```
Flight laxToSlc = new Flight();

Flight dfwToNyc = new Flight();
```



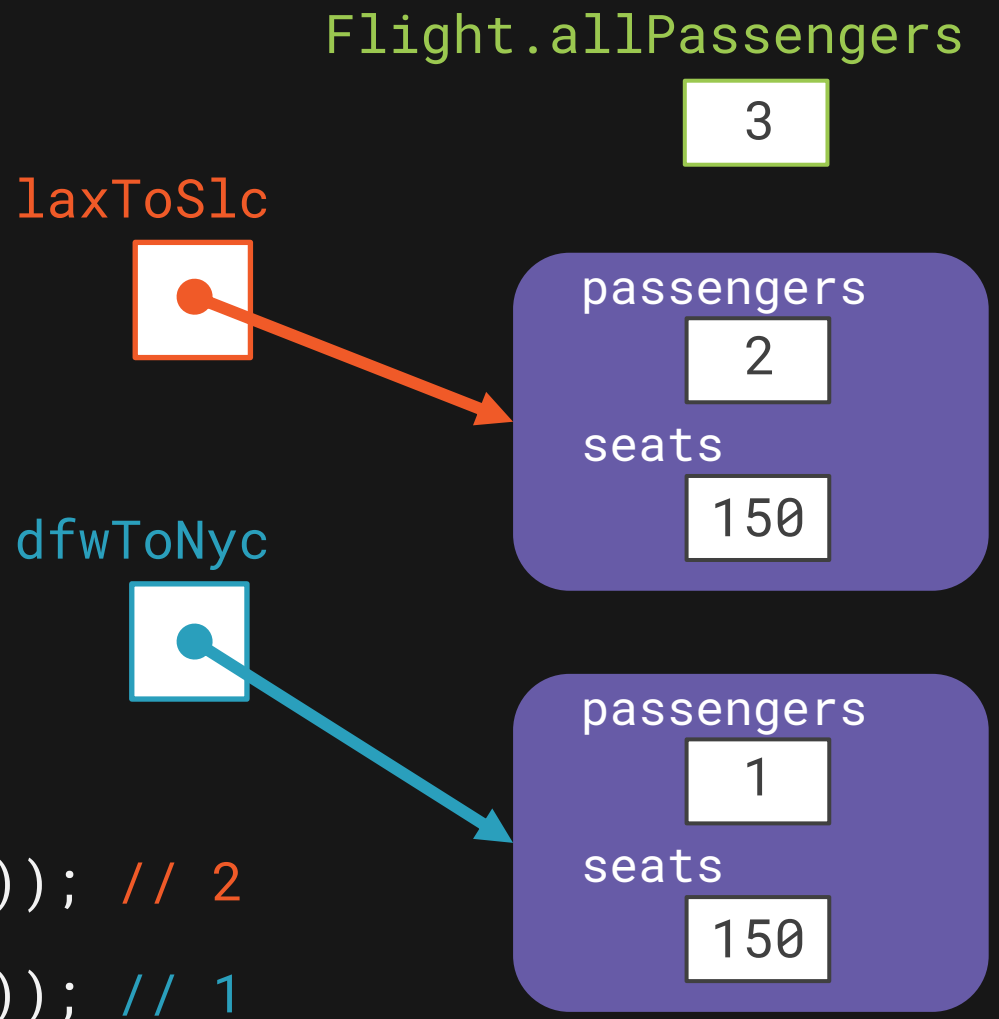
Main.java

```
Flight.resetAllPassengers();

Flight laxToSlc = new Flight();
laxToSlc.add1Passenger();
laxToSlc.add1Passenger();

Flight dfwToNyc = new Flight();
dfwToNyc.add1Passenger();

System.out.println(laxToSlc.getPassengers()); // 2
System.out.println(dfwToNyc.getPassengers()); // 1
System.out.println(Flight.getAllPassengers()); // 3
```



# Static Import Statement

**Import statement**

- Allows a type name to be used without being package-qualified

**Static import statement**

- Used with static methods
- Allows method name to be used without being class-qualified

Main.java

```
import static com.pluralsight.flightapp.Flight.resetAllPassengers;


Flight.resetAllPassengers();

Flight laxToSlc = new Flight();

laxToSlc.add1Passenger();

laxToSlc.add1Passenger();

Flight dfwToNyc = new Flight();

dfwToNyc.add1Passenger();

System.out.println(Flight.getAllPassengers());
```

Main.java

```
import static com.pluralsight.flightapp.Flight.resetAllPassengers;

import static com.pluralsight.flightapp.Flight.getAllPassengers;

resetAllPassengers();

Flight laxToSlc = new Flight();

laxToSlc.add1Passenger();

laxToSlc.add1Passenger();

Flight dfwToNyc = new Flight();

dfwToNyc.add1Passenger();

System.out.println(Flight.getAllPassengers());
```

Main.java

```
import static com.pluralsight.flightapp.Flight.resetAllPassengers;
import static com.pluralsight.flightapp.Flight.getAllPassengers;

resetAllPassengers();

Flight laxToSlc = new Flight();
laxToSlc.add1Passenger();
laxToSlc.add1Passenger();

Flight dfwToNyc = new Flight();
dfwToNyc.add1Passenger();

System.out.println(getAllPassengers());
```

Main.java

```
import static com.pluralsight.flightapp.Flight.*;


resetAllPassengers();

Flight laxToSlc = new Flight();
laxToSlc.add1Passenger();
laxToSlc.add1Passenger();

Flight dfwToNyc = new Flight();
dfwToNyc.add1Passenger();

System.out.println(getAllPassengers());
```

# Static Initialization Blocks

**Perform one-time type initialization**

- Execute before type's first use
- Has access to static members only

**Statements enclosed in brackets**

- Preceded by static keyword
- Outside of any method or constructor

Flight.java

```
public class Flight {
  private int passengers, seats = 150;
  private static int allPassengers, maxPassengersPerFlight;

  static {
    AdminService admin = new AdminService();
    admin.connect();
    maxPassengersPerFlight = admin.isRestricted() ?
        admin.getMaxFlightPassengers() : Integer.MAX_VALUE;
    admin.close();
  }
```

Flight.java

```
  public void add1Passenger() {
    if(passengers < seats) {
      passengers += 1;
      allPassengers += 1;
    }
  }

  // other members elided
}
```

Flight.java

```
  public void add1Passenger() {
    if(passengers < seats && passengers < maxPassengersPerFlight) {
      passengers += 1;
      allPassengers += 1;
    }
  }

  // other members elided
}
```

# Summary

**Static members**

- Shared class-wide
- Declared using the static keyword

# Summary

**Static fields**

- Values not associated with an instance
- All instances access the same value

**Static method**

- Perform action not tied to an instance
- Can only access static members

# Summary

**Static import statement**

- Allows static methods to be used without being class-qualified

**Static initialization blocks**

- Perform one-time type initialization
- Execute before type's first use