Class Inheritance

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim   jwhh.com

Overview

Inheritance overview

Derived class relationship to base class

Member hiding and overriding

Role of the Object class

Implementing support for equality checks

Class Inheritance

Use extends keyword

Derived has characteristics of base

Derived can add specialization

CargoFlight.java

```java
class CargoFlight extends Flight {
  float maxCargoSpace = 1000.0f;
  float usedCargoSpace;
  public void add1Package(float h, float w, float d) {
    float size = h * w * d;
    if (hasCargoSpace(size))
      usedCargoSpace += size;
    else
      handleNoSpace();
  }
```

CargoFlight.java

```java
  private boolean hasCargoSpace(float size) {
    return usedCargoSpace + size <= maxCargoSpace;
  }
  private void handleNoSpace() {
    System.out.println("Not enough space");
  }
}
```

Class Inheritance

Main.java

```
CargoFlight cf =
  new CargoFlight();
cf.add1Package(1.0f, 2.5f, 3.0f);

Passenger jack =
  new Passenger(0, 2);
cf.add1Passenger(jack);
```

CargoFlight.java

```
class CargoFlight extends Flight {
  public add1Package(
   float h, float w, float d){...}

  // other members elided
}
```

References to derived class instances

- Can be assigned to base class references

Available features

- Dictated by the type of reference being used to access the instance

References to Derived Class Instances

Main.java

```
            new CargoFlight();

Passenger jack =
  new Passenger(0, 2);

f.add1Passenger(jack);

f.~~add1Package~~(1.0f, 2.5f, 3.0f);
```

CargoFlight.java

```
class CargoFlight extends Flight {
  public add1Package(
   float h, float w, float d){...}

  // other members elided
}
```

References to Derived Class Instances

Main.java

```java
Flight[] squadron = new Flight[5]
squadron[0] = new Flight();
squadron[1] = new CargoFlight();
squadron[2] = new CargoFlight();
squadron[3] = new Flight();
squadron[4] = new CargoFlight();
```

CargoFlight.java

```java
class CargoFlight extends Flight {
  public add1Package(
   float h, float w, float d){...}

  // other members elided
}
```

Derived Class Members

Fields

Hide base class fields with same name

Flight.java

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

Field Hiding

Flight.java

```
class Flight {
  int seats = 150;
  private void hasSeating() {
    return passengers < seats;
  }
  // other members elided
}
```

CargoFlight.java

```
class CargoFlight extends Flight {
  int seats = 12;
  // other members elided
}
```

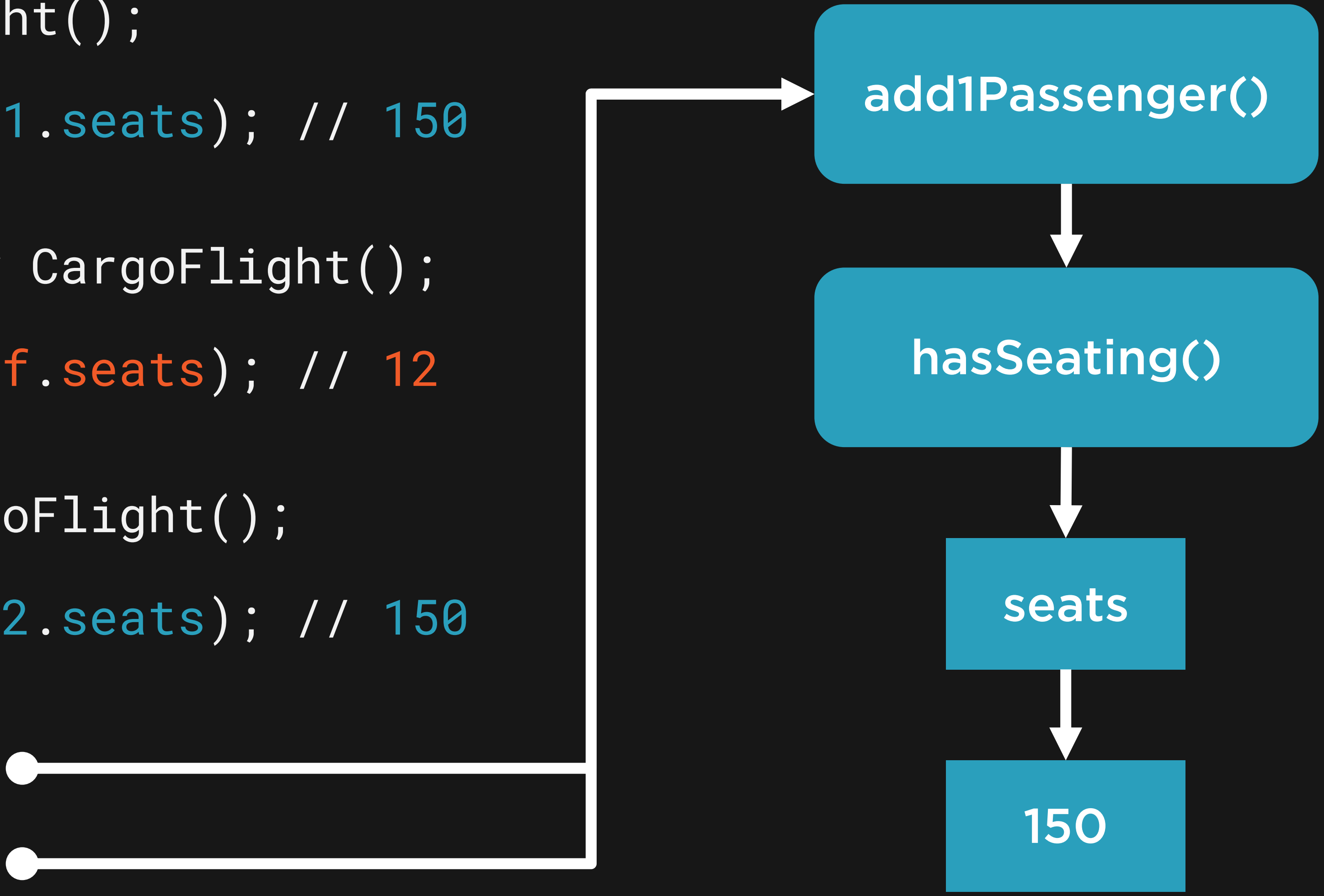
Main.java

```
Flight f1 = new Flight();
System.out.println(f1.seats); // 150

CargoFlight cf = new CargoFlight();
System.out.println(cf.seats); // 12

Flight f2 = new CargoFlight();
System.out.println(f2.seats); // 150

f2.add1Passenger();
cf.add1Passenger();
```



Derived Class Members

Fields

Hide base class fields with same name

Methods

Override base methods with same signature

Method Overriding

Flight.java

```
class Flight {
  int seats = 150;
  private void hasSeating() {
    return passengers < seats;
  }
  // other members elided
}
```

CargoFlight.java

```
class CargoFlight extends Flight {
  int seats = 12;
  // other members elided
}
```

Method Overriding

Flight.java

```
class Flight {
  int getSeats() { return 150; }
  private void hasSeating() {
    return passengers < seats;
  }
  // other members elided
}
```

CargoFlight.java

```
class CargoFlight extends Flight {
  int seats = 12;
  // other members elided
}
```

Method Overriding

Flight.java

```
class Flight {
  int getSeats() { return 150; }
  private void hasSeating() {
    return passengers < getSeats();
  }
  // other members elided
}
```

CargoFlight.java

```
class CargoFlight extends Flight {
  int seats = 12;
  // other members elided
}
```

Method Overriding

Flight.java

```java
class Flight {
  int getSeats() { return 150; }
  private void hasSeating() {
    return passengers < getSeats();
  }
  // other members elided
}
```

CargoFlight.java

```java
class CargoFlight extends Flight {
  int getSeats() { return 12; }
  // other members elided
}
```

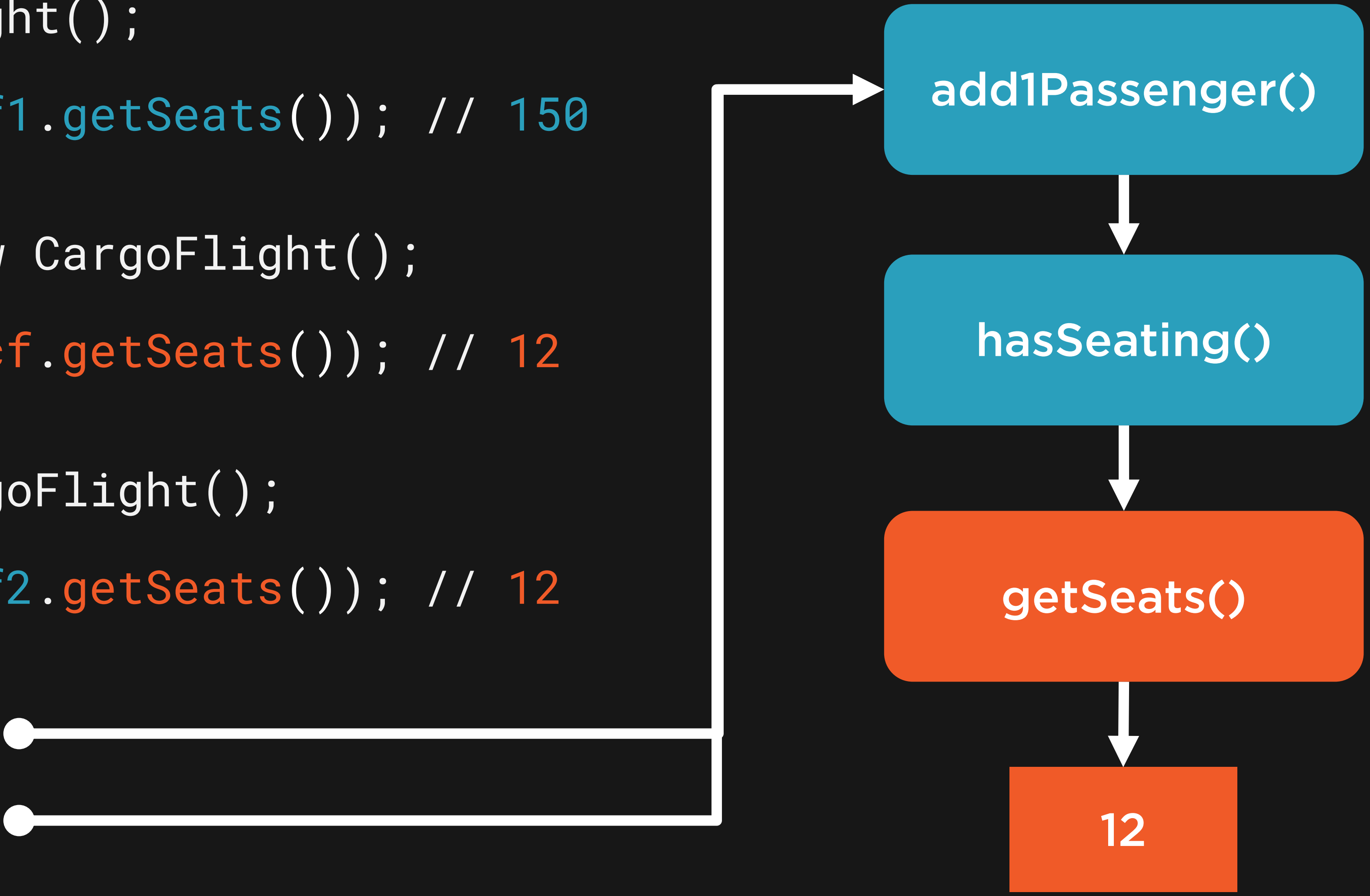
Main.java

```java
Flight f1 = new Flight();
System.out.println(f1.getSeats()); // 150

CargoFlight cf = new CargoFlight();
System.out.println(cf.getSeats()); // 12

Flight f2 = new CargoFlight();
System.out.println(f2.getSeats()); // 12

f2.add1Passenger();
cf.add1Passenger();
```



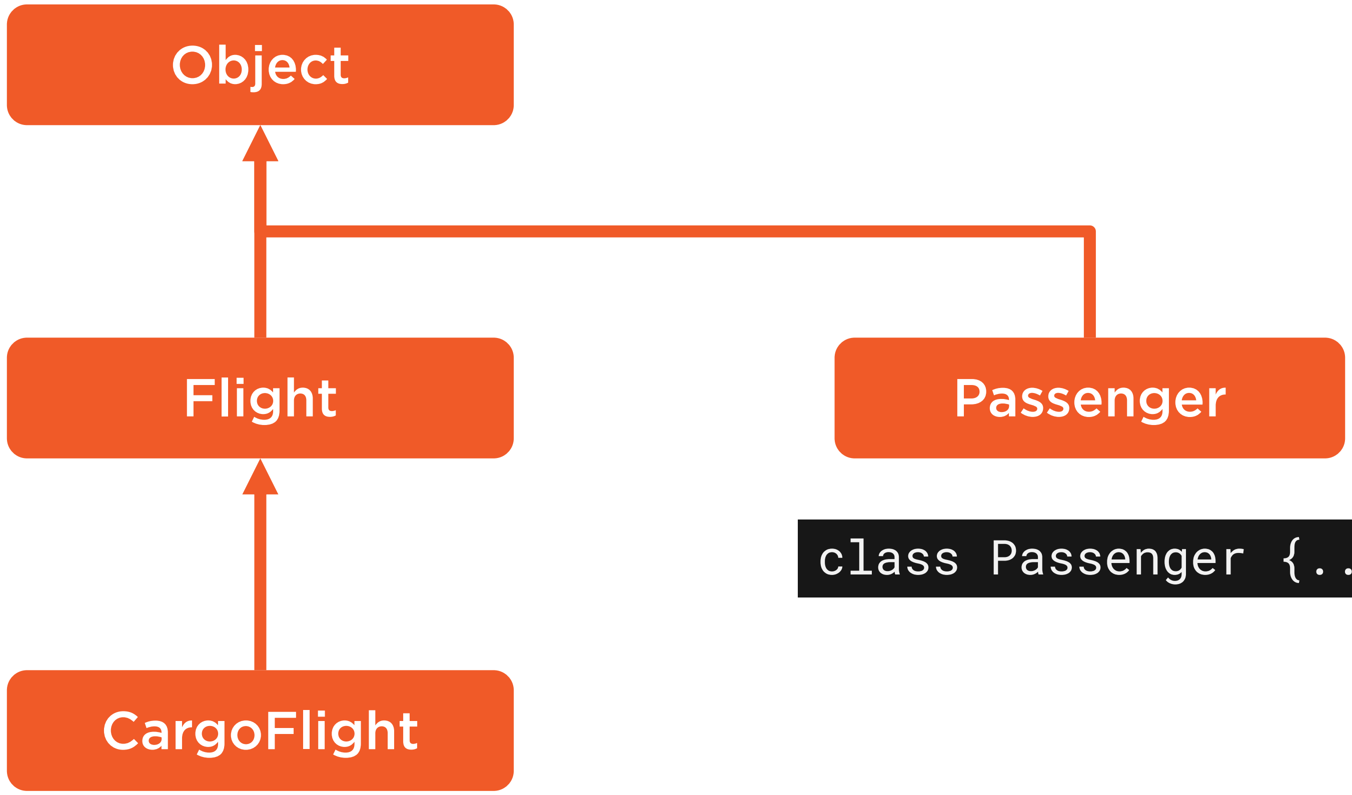
Object Class

Root of the Java class hierarchy

- Every class has characteristics of Object
- Object references can reference any array or class instance

Inheriting from Object

Every class inherits directly or indirectly from the Object class



```
class Flight {...}
```

```
class Passenger {...}
```

```
class CargoFlight extends Flight {...}
```

Object References

Main.java

```java
Object[] stuff = new Object[3];
stuff[0] = new Flight();
stuff[1] = new Passenger(0, 2);
stuff[2] = new CargoFlight();
```

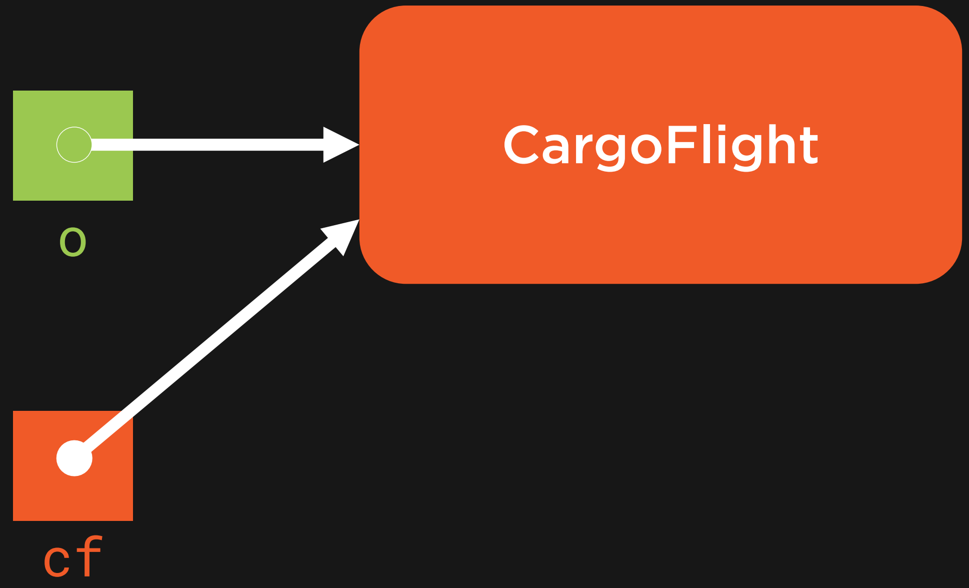
Main.java

```java
Object o = new Passenger();
o = new Flight[5];
```

Object References

```
Object o = new CargoFlight();
o.add1Package(1.0f, 2.5f, 3.0f);

if(o instanceof CargoFlight) {
   CargoFlight cf =               o;
   cf.add1Package(1.0f, 2.5f, 3.0f);
}
```



Object Class Methods

Method	Description
clone	Create a new object instance that duplicates the current instance
hashCode	Get a hash code for current instance
getClass	Return type information for the current instance
finalize	Handle special resource cleanup scenarios
toString	Return a string value representing the current instance
equals	Compare another object to the current instance for equality

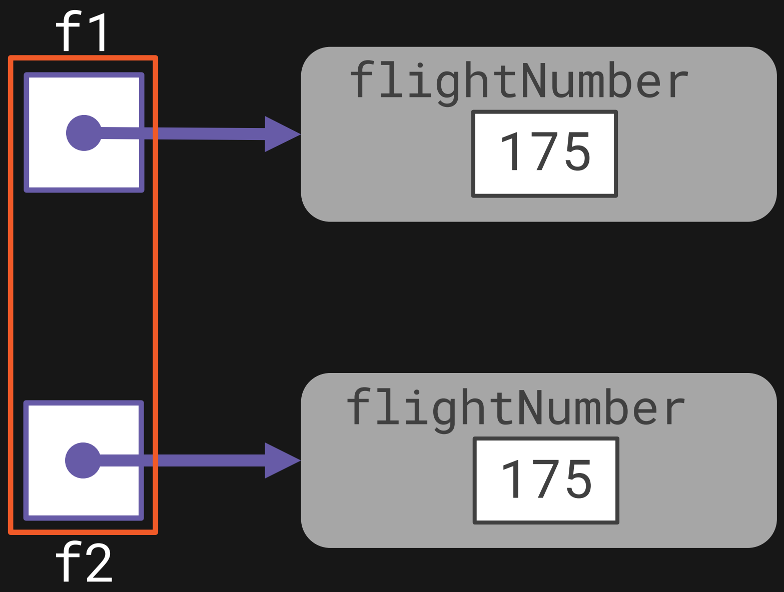
Equality

What does it mean to be equal? ... It depends.

```
Flight f1 = new Flight(175);
Flight f2 = new Flight(175);

if(f1 == f2) // false
    // do something

if(f1.equals(f2)) // false
    // do something
```



Flight.java

```java
public class Flight {
  private int flightNumber;
  private char flightClass;
  @Override
  public boolean equals(Object o) {
    Flight flight = (Flight) o;
    return flightNumber == flight.flightNumber
            flightClass == flight.flightClass;
  }
} // other members elided
```

Equality

What does it mean to be equal? ... It depends.

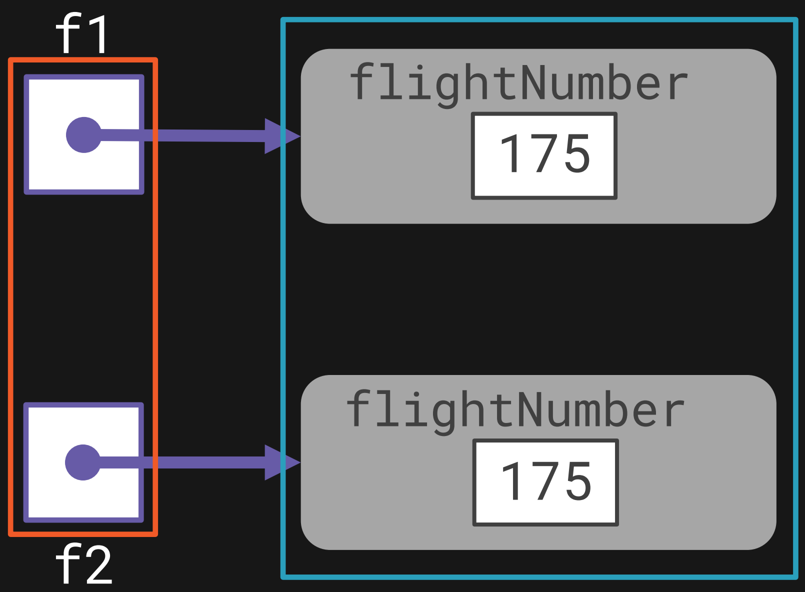
```
Flight f1 = new Flight(175);
Flight f2 = new Flight(175);

if(f1 == f2) // false
    // do something

if(f1.equals(f2)) // true
    // do something

Passenger p = new Passenger();

if(f1.equals(p)) // will crash at runtime
  // do something
```



Flight.java

```
public class Flight {
  private int flightNumber;
  private char flightClass;

  @Override
  public boolean equals(Object o) {
    if (   o instanceof Flight )
        return false;
    Flight flight = (Flight) o;
    return flightNumber == flight.flightNumber &&
            flightClass == flight.flightClass;
  }
} // other members elided
```

Equality

What does it mean to be equal? ... It depends.

```
Flight f1 = new Flight(175);
Flight f2 = new Flight(175);

if(f1 == f2) // false
    // do something

if(f1.equals(f2)) // true
    // do something

Passenger p = new Passenger();
if(f1.equals(p)) // false
  // do something
```

Summary

Inherit one class from another

- Derived has characteristics of base
- Derived can add specialization

Summary

Inheritance and type of reference

- Can assign derived class instance to base class reference
- Available features limited by reference

Derived class can override methods

- Must have same signature
- Derived class method used even when using a base class reference

Summary

Object class

- Root of Java class hierarchy
- Every class has Object characteristics
- Provides methods that classes commonly override

Checking for equality

- Equality operator checks references
- Override equals method to provide class specific equality comparisons