Inheritance

- Allows programmer to derive a class from an existing one
- Existing class is called the parent class, or superclass
- Derived class is called the child class or subclass
- The child class inherits the methods and data defined for the parent class
- Inherited trait can be accessed as though it were locally declared (defined)

Inheritance, cont’d

Inheritance relationships often shown graphically in a class diagram, with the arrow pointing to the parent class

![Class Diagram]

Create an is-a relationship, meaning the child is a more specific version of the parent

Single inheritance: one parent only

Deriving a subclass

Reserved word extends establishes an inheritance relationship:

```java
class Vehicle {
    // class contents
}

class Car extends Vehicle {
    // class contents
}
```
protected visibility

- Visibility modifiers control which members get inherited
  - **private**
    - Not inherited, can be accessed by local class only
  - **public**
    - Inherited, can be accessed by all classes
  - **protected**
    - Inherited, can be accessed by subclasses

- **Access**: access as though declared locally
- All variables from a superclass exist in the subclass, but some (private) cannot be accessed directly

```java
public class Vehicle {
    protected int plateNum; // plate #
    private int numWheels;  // # of wheels

    public void writeOut() {
        System.out.println("Plate number:" + plateNum);
    }
} // class Vehicle

public class Plane extends Vehicle {
    protected double wingSpan;
    private boolean hasPropeller;

    public void writeProperties() {
        System.out.println(plateNum); // ?
        System.out.println(numWheels); // ?
        if (hasPropeller) // ?
            System.out.println("Prop plane");
    }
} // class Plane
```

Reserved word super

- Invoke constructor of superclass
  ```java
  super(parameter-list);
  ```
  - parameter-list must match that in superclass' constructor

- Access methods and variables from superclass

```java
// New definition of class Vehicle
public class Vehicle {
    protected int plateNum; // plate #
    private int numWheels;  // # of wheels

    public Vehicle(int plate, int wheels) {
        plateNum = plate;
        numWheels = wheels;
    }

    public void writeOut() {
        System.out.println("Plate number:" + plateNum);
    }
} // class Vehicle
```
// New definition of class Plane
public class Plane extends Vehicle {
  protected double wingSpan;
  private boolean hasPropeller;

  public Plane(int plate, int wheels, double span, boolean prop) {
    super(plate, wheels);
    wingSpan = span;
    hasPropeller = prop;
  }

  public void writeProperties() {
    System.out.println(plateNum);
    //System.out.println(numWheels);
    if (hasPropeller)
      System.out.println("Prop plane");
  }
}

// Better definition of class Plane
public class Plane extends Vehicle {
  protected double wingSpan;
  private boolean hasPropeller;

  public Plane(int plate, int wheels, double span, boolean prop) {
    super(plate, wheels);
    wingSpan = span;
    hasPropeller = prop;
  }

  // Override method writeOut
  // Also access method from superclass
  public void writeOut() {
    super.writeOut();
    System.out.println("Wing Span: "+
    wingSpan);
    if (hasPropeller)
      System.out.println("Prop plane");
  }
}

### Overriding methods

- Subclass can override definition of inherited method in favor of its own
- New method in subclass must have same signature as superclass (but has different method body)
- Which method gets used??
  *The object that is used to invoke a method determines which version is used*
- Method declared to be final cannot be overridden
- Do not confuse overriding with overloading!

### Important ideas in inheritance

- Use different hierarchies for different problems
- Single inheritance
- Keep common features as high in the hierarchy as reasonably possible
- Inherited features are continually passed down the line
- Use the superclass’ features as much as possible
- "Inherited" ⇒ “can be accessed as though declared locally”
  *(private variables in superclass exists in subclasses; they just cannot be accessed directly)*