CS161
Intro to Computer Science I
Inheritance & Interacting w/ the Superclass
Chap. 9.1 – 9.2

Inheritance
• Code Reuse
• "Is A" relationship
• Superclass vs. Subclass
• Examples

Extending a Class

```java
public class Employee {
    public int getHours() {
        return 40;
    }
    public double getSalary() {
        return 40000.0;
    }
    public int getVacationDays() {
        return 10;
    }
}
```
Extending a Class

public class Secretary {
    public int getHours() {
        return 40;
    }
    public double getSalary() {
        return 40000.0;
    }
    public int getVacationDays() {
        return 10;
    }
    public void takeDictation (String text) {
        System.out.println("Dictating text: " + text);
    }
}

• What is different from the Employee Class?

Extending a Class

• We'd like to say:
  public class Secretary {
      copy all the contents from the Employee class;
      public void takeDictation(String text) {
          System.out.println("Taking dictation of text: " + text);
      }
  }
  public class Secretary extends Employee {
      public void takeDictation (String text) {
          System.out.println("Dictating text: " + text);
      }
  }

• What does this mean for an instance of the Secretary class?

Extending a Class

public class EmployeeMain {
    public static void main (String[] args) {
        Employee jen = new Employee();
        Secretary brian = new Secretary();
        System.out.println("Hours: " + jen.getHours());
        System.out.println("Hours: " + brian.getHours());
        System.out.println(brian.takeDictation());
    }
}

• Why can't jen access takeDictation()?
Overriding Methods

```java
public class Secretary extends Employee {
    public double getSalary() {
        return 25000.0;
    }
    public void takeDictation(String text) {
        System.out.println("Dictating text: " + text);
    }
}
```

Overriding Methods

```java
public class EmployeeMain{
    public static void main (String[] args) {
        Employee jen = new Employee();
        Secretary brian = new Secretary();

        System.out.println("Salary: " + jen.getSalary());
        System.out.println("Salary: " + brian.getSalary());
    }
}
```

• What does brian.getSalary() print?

Override vs. Overload

• What is the difference?
• How are they similar?
Problems w/ Inheritance Example

public class Employee {
    public int getHours() {
        return 40;
    }
    public double getSalary() {
        return 40000.0;
    }
    public int getVacationDays() {
        return 10;
    }
}

Problems w/ Inheritance Example

public class Secretary extends Employee {
    public double getSalary() {
        return 25000.0;
    }
    public void takeDictation (String text) {
        System.out.println(" Dictating text: "+text);
    }
}

public class LegalSecretary extends Secretary {
    public double getSalary() {
        return 30000.0;
    }
}

Problems w/ Inheritance Example

• What if we want to give everyone a raise?
• How can we fix this?
  – Use the Superclass's getSalary()!
  – Why is this better?
Fixing the Problem

public class Secretary extends Employee {
    public double getSalary() {
        return (getSalary() – 15000);
    }
    public void takeDictation(String text) {
        System.out.println("Dictating text: " + text);
    }
}

• Why is this BAD?

Fixing the Problem

public class Secretary extends Employee {
    public double getSalary() {
        return (Employee.getSalary() – 15000);
    }
    public void takeDictation(String text) {
        System.out.println("Dictating text: " + text);
    }
}

• Why is this BAD?

Fixing the Problem

public class Secretary extends Employee {
    public double getSalary() {
        return (super.getSalary() – 15000);
    }
    public void takeDictation(String text) {
        System.out.println("Dictating text: " + text);
    }
}

public class LegalSecretary extends Secretary {
    public double getSalary() {
        return (super.getSalary()) + 5000;
    }
}

• Why do we subtract for Secretary and add for LegalSecretary?
Employee Class with Attribute and Constructor

```java
class Employee {
    private int years;
    public Employee(int initialYears) {
        years = initialYears;
    }
    public int getHours() {
        return 40;
    }
    public double getSalary() {
        return 40000.0;
    }
    public int getVacationDays() {
        return 10 + 2 * years;
    }
}
```

Employee Class with Attribute and Constructor

• Do subclasses inherit the constructor?
  — No, it won’t compile...  
  Secretary.java:2: cannot find symbol
  symbol  : constructor Employee()
  location: class Employee
  public class Secretary extends Employee {

• What should you do?

Employee Class with Attribute and Constructor

```java
public class Secretary extends Employee {
    public Secretary(int years) {
        super(years);
    }
}
```

• The super call must be the first statement in the constructor.

Employee Class with Attribute and Constructor

```java
```
Employee Class with Attribute and Constructor

• What happens to subclass of subclass, i.e. LegalSecretary?
  – Depends on the superclass of the subclass
    public class Secretary extends Employee {
      public Secretary(int years) {
        super(years);
      }
    }
    ... 
    Either/or...
    public class LegalSecretary extends Secretary {
      public LegalSecretary(int years) {
        super(years);
      }
    }
    ...
    public class LegalSecretary extends Secretary {
      public LegalSecretary() {
        super(0);
      }
    }
    ...

Inheriting Fields/Attributes

• Try to give Secretary $5000 for each year at the company:
  public class Secretary extends Employee {
    ... 
    public double getSalary() {
      return super.getSalary() + 5000 * years;
    }
    ...
  }
• Doesn’t compile:
  Secretary.java:7: years has private access in Employee 
  return super.getSalary() + 5000 * years;
  ^
Inheriting Fields/Attributes

• Private fields cannot be directly accessed from subclasses.
  – Subclasses can’t break encapsulation.
  – What do we do?
    public class Employee{
      private int years;
      public Employee(int initialYears){
        years = initialYears;
      }
      public int getYears(){
        return years;
      }
    }

    public class Secretary extends Employee{
      public Secretary(int years){
        super(years);
      }
      public double getSalary(){
        return super.getSalary() + 5000 * getYears();
      }
    }

• Why don’t we need to have super.getYears()?