Class Design

Creating Classes

```
// Mammal.java
public class Mammal {
  private int age;
  protected String name;

  public int getAge() {
    return age;
  }
  public setAge(int theAge) {
    age = theAge;
  }
}
```

```java
// Dog.java

public class Dog extends Mammal {

  protected void setNameAndAge(String dogName, int age) {

    name = dogName;

    setAge(age);

  }

  public void barks() {

    System.out.println("Dog " + name + " (" + getAge() + ") says: woof!");

  }


  // main method

}
```

inherited from Mammal class

inherited from Mammal class

```java
// Mammal.java
public class Mammal {
  private int age;
  protected String name;
   ...
}
```

```
// main method
public static void main(String[] args) {
  Dog dog = new Dog();
  dog.setNameAndAge("Rex", 5);
  dog.barks();
}
```

```
Dog Rex (5) says: woof!
```

```
// if one of the classes is not public, you can have both classes
// in the same file (e.g. Dog.java, if Mammal is not public)
```

```
// keyword "this" is used to access the members of the class you're in


// setter method without using "this"
public setAge(int theAge) {
  age = theAge;
}


// setter method using "this"
public setAge(int age) {
  this.age = age;
}
```

```
// Mammal.java
public class Mammal {
  private int age;
  protected String name;
  ...
}
```

Java looks if there is a local variable with specific name,

if not it will use instance variable

```
// keyword "super" is used to access the members of the superclass


// setNameAndAge() method without using "super"
protected void setNameAndAge(String dogName, int age) {
  name = dogName;
   setAge(age);
}


// setNameAndAge() method using "super"
protected void setNameAndAge(String name, int age) {
  super.name = name;
   setAge(age);
}
```

```
// Mammal.java
public class Mammal {
  private int age;
  protected String name;
  ...
}
```