Scope

Scope of Local Variables

- **Scope of variable:** Region of program in which the variable can be accessed
- Scope of a local variable extends from its declaration to end of the block that encloses it

Scope of Local Variables

- Sometimes the same variable name is used in two methods:

```
public class RectangleTester
{
   public static double area(Rectangle rect)
   {
      double r = rect.getWidth() * rect.getHeight();
   return r;
   }
   public static void main(String[] args)
   {
      Rectangle r = new Rectangle(5, 10, 20, 30);
      double a = area(r);
      System.out.println(r);
   }
}
```

Scope of Local Variables

- These variables are independent from each other; their scopes are disjoint

Scope of Local Variables

- Scope of a local variable cannot contain the definition of another variable with the same name

```
Rectangle r = new Rectangle(5, 10, 20, 30);
if (x >= 0)
{
   double r = Math.sqrt(x);
   // Error-can't declare another variable called r here
   . . .
}
```

Scope of Local Variables

- However, can have local variables with identical names if scopes do not overlap

```
if (x >= 0)
{
   double r = Math.sqrt(x);
   . . .
} // Scope of r ends here
else
{
   Rectangle r = new Rectangle(5, 10, 20, 30);
   // OK-it is legal to declare another r here
   . . .
}
```

Scope of Class Members

- Private members have class scope: You can access all members in any method of the class
- Must qualify public members outside scope

```
Math.sqrt
harrysChecking.getBalance
```

Scope of Class Members

- Inside a method, no need to qualify fields or methods that belong to the same class
- An unqualified instance field or method name refers to the `this` parameter

```
public class BankAccount
{
   public void transfer(double amount, BankAccount other)
   {
      withdraw(amount); // i.e., this.withdraw(amount);
      other.deposit(amount);
   }
   ...
}
```

Overlapping Scope

- A local variable can *shadow* a field with the same name
- Local scope wins over class scope

```
public class Coin
{
   . . .
   public double getExchangeValue(double exchangeRate)
   {
      double value; // Local variable
      . . .
      return value;
   }
   private String name;
   private double value; // Field with the same name
}
```

Overlapping Scope

- Access shadowed fields by qualifying them with the this reference

```
value = this.value * exchangeRate;
```