**Administrative**

**Today’s session**

BigDecimal data type

Inheritance

**Session Topics**

**BigDecimal data type**

● Data type **double** stores numbers in eight bytes and shows up to ~17 significant digits.

● Data type double represents real numbers approximately.

● Data type **BigDecimal** represents real numbers much more precisely.

● Data type BigDecimal consists of:

✓ A number

✓ A 32-bit integer scale. If the scale is zero or positive, the scale is the number of digits to the right of the decimal point. If the scale is negative, the unscaled value of the number is multiplied by ten to the power of -scale.

● Data type BigDecimal should be used for financial and scientific applications where precision is very important.

● See **BigDecimal data type** sample application on Blackboard.

**Inheritance**

● **Inheritance** is one class having access to one or more members of another class.

● Inheritance enables a hierarchy of classes to be defined.

● Classes at the top of the hierarchy are more general while classes at the bottom of the hierarchy are more specific.

● Inheritance enables the same code to be used in multiple classes such that the overall amount of code is smaller.

● Inheritance is accomplished with keyword **extends**.

● Example of keyword extends:

public class Class2 extends Class1

{

…

}

● There are multiple ways to describe Class2 and Class1 from the example:

✓ Class2 is the **subclass**, **child class**, or **derived class**.

✓ Class1 is the **superclass**, **parent class**, or **base class**.

● Both Java classes and custom classes may be extended and modified.

● A superclass may have one or more subclasses.

● Every class, except the **Object** class, has exactly one superclass.

● An object of a subclass is also an object of its superclass.

**Inherited members**

● The subclass inherits the **public** and **protected** members of the superclass.

● If the subclass and superclass are in the same package, the subclass also inherits the **package-private**members of the superclass.

● Although a subclass does not inherit the constructor method of the superclass, it may call the method using keyword **super**.

● If no call to the super constructor is made in the subclass, the default (no values) super constructor is called.

**Overridden and hidden members**

● A subclass may declare an instance variable with the same name, or a instance method with the same signature, as one in the superclass. This **overrides**, and makes unavailable, the variable or method in the superclass.

● A subclass may declare a static variable with the same name, or a static method with the same signature, as one in the superclass. This **hides** thevariable ormethod in the superclass.

● To access the superclass version of a static field or method, we precede it with the superclass name.

● This table summarizes the result of defining a subclass field with the same name, or a subclass method with the same signature, as a superclass method:

|  |  |
| --- | --- |
|  | Superclass field or method |
| Subclass instance field or method | Overrides; the superclass version may not be accessed. |
| Subclass static field or method | Hides; the superclass version may be accessed with a superclass prefix. |

● Ken exception: whether overriding or hiding, the child member and the parent member must both be either instance or static. Otherwise, a syntax error will occur.

● An overridden method may have a less restrictive access modifier:

Superclass

public class Vehicle

{

…

protected Vehicle getVehicle()

{

…

}

…

}

Subclass

public class Car extends Vehicle

{

…

public Vehicle getVehicle()

{

…

}

…

}

● Modifier **final** may be used in the following declarations:

✓ Field – the field value may not be modified (it becomes a constant).

✓ Method – the method may not be overridden.

✓ Class – the class may not be extended.

● See **Inheritance** sample application on Blackboard.