9. Inheritance :: Overriding Methods

A subclass inherits the **public** and **protected** methods of its superclass. In many cases, the behavior of those methods is exactly what the subclass needs:

```
public class Super {
  private int mX;
  ...
  public int getX() { return mX; }
  public void setX(int pNewX) { mX = pNewX; }
}
```

```
public class Sub extends Super {
  // getX() and setX() are inherited and do exactly what Sub needs, i.e., given
  // a Sub object, we can call getX() and setX() on that object to read and write
  // the mX instance variables of the Sub object.
  //
  // Sub sub = new Sub();
  // sub.setX(100);        // Sets mX of sub to 100.
  // ...
  // int x = sub.getX();   // Sets x to mX of sub.
}
```

9. Inheritance :: Overriding Methods (continued)

However, in other situations the behavior of the inherited method is not exactly what the subclass requires and in this case, the subclass can provide its own implementation of the method which exhibits the behavior required by the subclass.

For example, suppose undergraduate students at a university are assessed tuition per this table:

$\textit{credit hours} < 12$	$\textit{tuition} = \$500 \times \textit{credit hours}$
$12 \leq \textit{credit hours} \leq 18$	$\textit{tuition} = \$5{,}000$
$\textit{credit hours} > 18$	$\textit{tuition} = \$5{,}000 + \$750 \times (\textit{credit hours} - 18)$

We write the code for the *Student* class implementing a *calcTuition*() method:

```
public class Student {
  public double calcTuition() {
    double tuition;
    if (creditHours() < 12) {
      tuition = 500 * creditHours();
    } else if (creditHours() > 18)
      tuition = 5000 + 750 * (creditHours() - 18);
    } else {
      tuition = 5000;
    }
    return tuition;
  }
}
```

9. Inheritance :: Overriding Methods (continued)

At the same university, a graduate student's tuition is assessed differently:

$\textit{credit hours} < 6$	$\textit{tuition} = \$750 \times \textit{credit hours}$
$6 \le \textit{credit hours} \le 9$	$\textit{tuition} = \$7{,}500$
$\textit{credit hours} > 9$	$\textit{tuition} = \$7{,}500 + \$1{,}250 \times (\textit{credit hours} - 9)$

Graduate students are like undergraduate students in many ways (they're poor, they stay up late and sleep late, they start every assignment at the last minute, etc) so we can create a *GraduateStudent* class that inherits many of the attributes and behaviors of the *Student* class. However, a *GraduateStudent* is different than a *Student* in one way: their tuition is calculated using a different algorithm. Consequently, we can **override** the *calcTuition*() method in the *GraduateStudent* subclass by writing a new implementation of *calcTuition*().

9. Inheritance :: Overriding Methods (continued)

```
public class GraduateStudent extends Student {
  // An inherited method which is "re-implemented" in a subclass is said to be
  // overridden in the subclass. Subclasses override methods when the behavior
  // of the method in the subclass differs from the behavior of the method in
  // the superclass. If the behavior of the method in superclass already does
  // exactly what the subclass needs, then the subclass would not override the
  // method.
  public double calcTuition() {
    double tuition;
    if (creditHours() < 6) {
      tuition = 750 * creditHours();
    } else if (creditHours() > 9) {
      tuition = 7500 + 1250 * (creditHours() - 9);
    } else {
      tuition = 7500;
    }
    return tuition;
  }
}
```