Inheritance Mechanics

1. Consider the following classes:

```java
public class Snow {
    public void method2() {
        System.out.println("Snow 2");
    }
    public void method3() {
        System.out.println("Snow 3");
    }
}

public class Rain extends Snow {
    public void method1() {
        System.out.println("Rain 1");
    }
    public void method2() {
        System.out.println("Rain 2");
    }
}

public class Sleet extends Snow {
    public void method2() {
        System.out.println("Sleet 2");
        super.method2();
        method3();
    }
    public void method3() {
        System.out.println("Sleet 3");
    }
}

public class Fog extends Sleet {
    public void method1() {
        System.out.println("Fog 1");
    }
    public void method3() {
        System.out.println("Fog 3");
    }
}
```

Suppose the following variables are defined:

```java
Snow var1 = new Sleet();
Snow var2 = new Rain();
Sleet var3 = new Fog();
```

In the table below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, indicate the line breaks with slashes as in "a / b / c" to indicate three lines of output with "a" followed by "b" followed by "c". If the statement causes an error, fill in the right-hand column with the phrase "error" to indicate this.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) var1.method2();</td>
<td>Snow 2</td>
</tr>
<tr>
<td>b) var2.method1();</td>
<td>Rain 1</td>
</tr>
<tr>
<td>c) var2.method2();</td>
<td>Rain 2</td>
</tr>
<tr>
<td>d) var3.method2();</td>
<td>Sleet 2 / Sleet 3</td>
</tr>
<tr>
<td>e) var3.method3();</td>
<td>Sleet 3</td>
</tr>
<tr>
<td>f) ((Rain) var2).method1();</td>
<td>Rain 1</td>
</tr>
<tr>
<td>g) ((Sleet) var1).method3();</td>
<td>Sleet 3</td>
</tr>
<tr>
<td>h) ((Sleet) var1).method1();</td>
<td>Fog 1</td>
</tr>
<tr>
<td>i) ((Sleet) var2).method3();</td>
<td>Fog 3</td>
</tr>
</tbody>
</table>

(Solutions on Lecture Slides)