Inheritance + Polymorphism

Hitesh Boinpally
Summer 2023
Agenda

• Final Logistics
• Inheritance
• Different Errors
• Polymorphism
Inheritance

- **Inheritance**: Forming hierarchial relationships between classes
  - Allows for sharing / reusing of code between classes
  - **Superclass**: The class being extended
  - **Subclass**: The class that inherits behavior from superclass
    - Gains copy of every method
Inheritance

- **Inheritance**: Forming hierarchial relationships between classes
  - Allows for sharing / reusing of code between classes
  - **Superclass**: The class being extended
  - **Subclass**: The class that inherits behavior from superclass
    - Gains copy of every method
- Inheritance forms an “is-a” relationship
  - Tiger extends Cat
  - Means that Tiger “is-a” Cat
public class MusicPlayer {
    public void m1() {
        S.o.pln("MusicPlayer1");
    }
}

public class TapeDeck extends MusicPlayer {
    public void m3() {
        S.o.pln("TapeDeck3");
    }
}

public class iPod extends MusicPlayer {
    public void m2() {
        S.o.pln("IPod2");
        m1();
    }
}

public class IPhone extends iPod {
    public void m1() {
        S.o.pln("IPhone1");
        super.m1();
    }
    public void m3() {
        S.o.pln("IPhone3");
    }
}
public class MusicPlayer {
    public void m1() {
        s.o.pln("MusicPlayer1");
    }
}

public class TapeDeck extends MusicPlayer {
    public void m3() {
        s.o.pln("TapeDeck3");
    }
}

public class iPod extends MusicPlayer {
    public void m2() {
        s.o.pln("IPod2");
        m1();
    }
}

public class IPhone extends iPod {
    public void m1() {
        s.o.pln("IPhone1");
        super.m1();
    }

    public void m3() {
        s.o.pln("IPhone3");
    }
}
<table>
<thead>
<tr>
<th></th>
<th>m1()</th>
<th>m2()</th>
<th>m3()</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MusicPlayer</strong></td>
<td>MP1</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td><strong>TapeDeck</strong></td>
<td>MP1</td>
<td>/</td>
<td>TD3</td>
</tr>
<tr>
<td><strong>IPod</strong></td>
<td>MP1</td>
<td>IPod2</td>
<td>/</td>
</tr>
<tr>
<td><strong>IPhone</strong></td>
<td>IPhone1</td>
<td>IPod2</td>
<td>IPhone3</td>
</tr>
</tbody>
</table>

```javascript
MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new iPod();
MusicPlayer var3 = new IPhone();
IPod var4 = new IPhone();
Object var5 = new iPod();
Object var6 = new MusicPlayer();

var1.m1();
var3.m1();
var4.m2();
var3.m2();
var5.m1();
```
<table>
<thead>
<tr>
<th>Class</th>
<th>m1()</th>
<th>m2()</th>
<th>m3()</th>
</tr>
</thead>
<tbody>
<tr>
<td>MusicPlayer</td>
<td>MP1</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>TapeDeck</td>
<td>MP1</td>
<td>/</td>
<td>TD3</td>
</tr>
<tr>
<td>Ipod</td>
<td>MP1</td>
<td>Ipod2 m1()</td>
<td>/</td>
</tr>
<tr>
<td>Iphone</td>
<td>IPhone1 MP1</td>
<td>Ipod2 m1()</td>
<td>IPhone3</td>
</tr>
</tbody>
</table>

```javascript
MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new Ipod();
MusicPlayer var3 = new Iphone();
Ipod var4 = new Iphone();
Object var5 = new Ipod();
Object var6 = new MusicPlayer();

var1.m1();
MusicPlayer1
var3.m1();
IPhone1 / MusicPlayer1
var4.m2();
IPod2 / IPhone1 /
MusicPlayer1
var3.m2();
Compiler Error (CE)
var5.m1();
Compiler Error (CE)
```
<table>
<thead>
<tr>
<th>Class</th>
<th>m1()</th>
<th>m2()</th>
<th>m3()</th>
</tr>
</thead>
<tbody>
<tr>
<td>MusicPlayer</td>
<td>MP1</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>TapeDeck</td>
<td>MP1</td>
<td>/</td>
<td>TD3</td>
</tr>
<tr>
<td>iPod</td>
<td>MP1</td>
<td>IPod2</td>
<td>/</td>
</tr>
<tr>
<td>IPhone</td>
<td>IPhone1</td>
<td>IPod2</td>
<td>IPhone3</td>
</tr>
<tr>
<td></td>
<td>MP1</td>
<td>m1()</td>
<td></td>
</tr>
</tbody>
</table>

MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new IPod();
MusicPlayer var3 = new IPhone();
IPod var4 = new IPhone();
Object var5 = new IPod();
Object var6 = new MusicPlayer();

((TapeDeck) var1).m2();
((IPod) var3).m2();
((IPhone) var2).m1();
((TapeDeck) var3).m2();
MusicPlayer var1 = new TapeDeck();
MusicPlayer var2 = new IPod();
MusicPlayer var3 = new IPhone();
IPod var4 = new IPhone();
Object var5 = new IPod();
Object var6 = new MusicPlayer();

((TapeDeck) var1).m2();
Compiler Error (CE)
((IPod) var3).m2();
Runtime Error (RE)
((TapeDeck) var3).m2();
Compiler Error (CE)
The Rules

First we define a few things with a color code:

```java
DeclaredType name = new ObjectType();  // declare variable
name.method();                         // call method
((CastToType)name).method();          // cast object, then call a method
```

When we try to execute one of the latter two, we follow this progression:

1. CASTING
   - **Yes**
     - **Does** CastToType **contain method?**
       - **Yes**
         - **Can cast** ObjectType to CastToType?
           - **Yes**
             - **Execute** method of ObjectType
           - **No**
             - **Runtime Error**
       - **No**
         - **Compiler Error**
   - **No**
     - **Does** DeclaredType **contain method?**
       - **Yes**
         - **Execute** method of ObjectType
       - **No**
         - **Compiler Error**