Inheritance

Ch 5.5
Highlights

- Extending classes (inheritance)

```java
public class Human extends Mammal {

    public Human() {
        super();
    }
}
```

- super keyword
Story time

A long time ago in a galaxy far, far away....
Story time
Story time
Story time
I ARE DUNECAT

I CONTROLS THE SPICE,
I CONTROLS THE UNIVERSE.

DUNE CAT

haz no fear, fear iz mindkillerz
Extending classes

(See: DuneCat.java and ArrakianSandworm.java)

Extended (child) classes get objects and methods from their parent class

For example, DuneCat (child) got the roar() method from ArrakianSandworm (parent)
Child classes get all objects and (public) methods from the parent class.

Remember: private means they can only be used inside the class where it is defined.

To access and modify private data, public methods are needed (get___() and set___()).

(See: Parent.java and Child.java)
Extending classes can be quite useful

If multiple things are very similar, you can put the shared information in one spot (instead of having multiple copies in both classes)
  - Easier to update/modify code
  - Makes code smaller (easier to understand)
### Slime Devil

Medium immortal humanoid (devil, ooze)

<table>
<thead>
<tr>
<th>HP 123; Bloodied 61</th>
<th>Initiative +18</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC 30, Fortitude 28, Reflex 29, Will 28</td>
<td>Perception +13</td>
</tr>
<tr>
<td>Speed 6, swim 6</td>
<td>Darkvision</td>
</tr>
<tr>
<td>Resist 20 acid</td>
<td></td>
</tr>
</tbody>
</table>

**Traits**
- Mercurial Body
  - The slime devil ignores difficult terrain and does not provoke opportunity attacks by moving.

**Standard Actions**
1. **Caustic Slam (acid) ♦ At-Will**
   - Attack: Melee 1 (one creature); +19 vs. Fortitude
   - Hit: 3d8 + 11 acid damage.
2. **Diabolical Envelopment (acid) ♦ At-Will**
   - Attack: Melee 1 (one Medium or smaller enemy); +19 vs. Reflex
   - Hit: The devil grabs the target and shifts 1 square into the target’s square. Until the grab ends, the target is dazed and takes ongoing 10 acid damage. While the devil has the target grabbed, attacks against the devil deal half damage to it and half damage to the grabbed creature. When the devil moves, it pulls the target with it. In addition, the target remains grabbed, and the devil does not provoke an opportunity attack from the target.

### Level 16 Lurker

XP 1,400

### Herald of Colorless Fire

Medium natural animate (construct, fire)

<table>
<thead>
<tr>
<th>HP 244; Bloodied 122</th>
<th>Initiative +25</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC 41, Fortitude 37, Reflex 40, Will 37</td>
<td>Perception +19</td>
</tr>
<tr>
<td>Speed 8, fly 6</td>
<td></td>
</tr>
<tr>
<td>Resist 15 fire</td>
<td></td>
</tr>
</tbody>
</table>

**Traits**
- Frozen in Place
  - Whenever the herald of colorless fire takes cold damage, it cannot use flickering flame until the end of its next turn.

**Standard Actions**
1. **Caress of Flame (fire, force) ♦ At-Will**
   - Attack: Melee 1 (one creature); +32 vs. AC
   - Hit: 3d10 + 19 fire and force damage.
2. **Storm of Colorless Fire (fire, force) ♦ Recharge [XX] [!!]**
   - Effect: The herald makes the following attack twice, shifting half its speed between the attacks. The herald cannot target the same creature with both attacks.
   - Attack: Close burst 1 (creatures in burst); +30 vs. Reflex
   - Hit: 4d10 + 16 fire and force damage, and ongoing 15 fire damage (save ends).
Derived classes

Parent:

(Internal combustion engine)

Children:
Phone
Using child and parent classes

You can think of a child class as a more specific instance of a parent class.

```java
public class WheeledTransport{};
public class Car extends WheeledTransport{};
public class Truck extends WheeledTransport{};
```

This means you can think of a child object as type parent (but not parent as type child).

```java
WheeledTransport toot = new Truck();
WheeledTransport henry = new Car();
```

(See: UsingInheritance.java)
super

super is a keyword that lets you access the parent class (constructor/methods/objects)

In a constructor you can use super() to run the parent's default constructor

Rule: the super() command must happen on the first line of a constructor

(See: SuperParent.java and SuperChild.java)
Oddities of super() constructor

If you do not code super() on the first line of a child class's constructor, then...

- The parent's default constructor will run
- If the parent does not have a default constructor, there will be an error

This is a reason why it is normally useful to have a default constructor defined, even if it does nothing
We introduced `this` a while back, but you can also use `this` to run constructors (like super)

If you use `this` as a constructor (like super), then it must happen on the first line of code (Note: you can only use `this` in this manner inside a constructor)

(See: ThisConstructor.java)
Override

Child classes can **override** methods from their parent's class.

This means a child can redefine a method to work in a completely different way than the parent's method.

Can use `super` to get the parent's version of an overridden method.

(See: OverrideSandworm.java)
You should always add @Override before every method you override (very important!)

When overriding, you can change modifiers and return types under these restrictions:
  1. The modifier must be more “open” (e.g. private -> public)
  2. The return type is a descendant (child) of the original return type (e.g. Parent -> Child)

(See: OverrideChild.java and OverrideParent.java)
Override vs Overload

When you override, you redefine an existing method (name) (same name, number of arguments and types)

When you overload, you add a new way of using a method (name) (same name, but different arguments)