Pokémon
&
Inheritance

(PI)

OOP

- Object-oriented programming is another paradigm that makes objects its central players, not functions.
- Objects are pieces of data and the associated behavior.
- Classes define an object, and can inherit methods and instance variables from each other.

Inheritance

Occasionally, we find that many abstract data types are related.

For example, there are many different kinds of people, but all of them have similar methods of eating and sleeping.

Inheritance

We would like to have different kinds of Pokémon, which differ (among other things) in the amount of points lost by its opponent during an attack.

The only method that changes is `attack`. All the other methods *remain the same*. Can we avoid *duplicating code* for each of the different kinds?

Inheritance

Key OOP Idea: Classes can inherit methods and instance variables from other classes

```
public class WaterPokemon extends Pokemon
{
    …
    void attack(Pokemon other)
    {
        other.decrease_hp(75);
    }
}
```

Inheritance

Key OOP Idea: Classes can inherit methods and instance variables from other classes

```
public class WaterPokemon extends Pokemon
{
    …
    void attack(Pokemon other)
    {
        other.decrease_hp(75);
    }
}
```

The Pokemon class is the ***superclass*** of the WaterPokemon class.

The WaterPokemon class is the ***subclass*** of the Pokemon class.

Inheritance

Key OOP Idea: Classes can inherit methods and instance variables from other classes

```
public class WaterPokemon extends Pokemon
{
    …
    void attack(Pokemon other)
    {
        other.decrease_hp(75);
    }
}
```

The `attack` method from the Pokemon class is ***overridden*** by the `attack` method from the `WaterPokemon` class.

Inheritance

```
WaterPokemon ashs_squirtle = new
      WaterPokemon(“Squirtle”,”Ash”, 314);
Pokemon mistys_togepi = new Pokemon(“Togepi”,
      “Misty”, 245);
mistys_togepi.attack(ashs_squirtle);
System.out.println(ashs_squirtle.getHitPts());
264
ashs_squirtle.attack(mistys_togepi);
System.out.println(mistys_togepi.getHitPts());
170
```

Inheritance

```
WaterPokemon ashs_squirtle = new
      WaterPokemon(“Squirtle”,”Ash”, 314);
Pokemon mistys_togepi = new Pokemon(“Togepi”,
      “Misty”, 245);
mistys_togepi.attack(ashs_squirtle);
System.out.println(ashs_squirtle.getHit
264
ashs_squirtle.attack(mistys_togepi);
System.out.println(mistys_togepi.getHitPts());
170
```

mistys_togepi uses the attack method from the Pokemon class.

Inheritance

```
WaterPokemon ashs_squirtle = new
      WaterPokemon(“Squirtle”,”Ash”, 314);
Pokemon mistys_togepi = new Pokemon(“Togepi”,
      “Misty”, 245);
mistys_togepi.attack(ashs_squirtle);
System.out.println(ashs_squirtle.getHitPts());
264
ashs_squirtle.attack(mistys_togepi);
System.out.println(mistys_togepi.getHi
170
```

ashs_squirtle uses the attack method from the WaterPokemon class.

Inheritance

```
WaterPokemon ashs_squirtle = new
      WaterPokemon(“Squirtle”,”Ash”, 314);
Pokemon mistys_togepi = new Pokemon(“Togepi”,
      “Misty”, 245);
mistys_togepi.attack(ashs_squirtle);
System.out.println(ashs_squirtle.getHitPts());
264
ashs_squirtle.attack(mistys_togepi);
System.out.println(mistys_togepi.getHit
170
```

The WaterPokemon class does not have a getHitPts method, so it uses the method from its superclass.

Inheritance: What Happens Here?

```
public class ElectricPokemon extends Pokemon
{
    String origin;
    public ElectricPokemon(String name, String owner, int
                                       hp, String origin)
    {
        this.origin = origin;
    }
}

ElectricPokemon ashs_pikachu = new
      ElectricPokemon(“Pikachu”, “Ash”, 300, “Pallet
                                              Town”);

System.out.println(ashs_pikachu.getHitPts());
```

Inheritance: What Happens Here?

One fix is to first call the constructor of the superclass. The constructor of the subclass overrode the constructor of the superclass, which is why the other instance variables were never assigned (and gave a compile error).

```
public class ElectricPokemon extends Pokemon
{
    String origin;
    public ElectricPokemon(String name, String owner, int
                                        hp, String origin)
    {
        super(name, owner, hp);
        this.origin = origin;
    }
}
```