Inheritance & Polymorphism
Today's Goals

Our goals are to understand how methods get inherited and how Objects in a hierarchy interact.

- Clock c = new DigitalClock(true);
- AnalogClock ac = new DigitalClock(true);
- PreciseDigitalClock pdc = new DigitalClock(true);
- c.getTime(); ac.getTime(); pdc.getTime();
Clock

Clock Class

```java
public class Clock {
    private int hour;
    private int minute;

    public int getMinute() { return this.minute; }
    public int getHour() { return this.hour; }
    public String getTime() { return hour + " " + minute; }
}
```

OUTPUT

```java
>> Clock c = new Clock(); // hour = 4, minute = 12
>> System.out.println(c.getTime() + "..." + c.getHour() + "..." + c.getMinute());
>> 4 12...4...12
```

What specializations could we make to Clock?

- An “analog” clock with a face?
- A “digital” clock with military time?
- A clock with seconds?
For each of the following, is it always, sometimes, or never true:

- A DigitalClock is a Clock?
  - Always!
  - A DigitalClock is a Clock with digital features.

- An AnalogClock is a DigitalClock?
  - Never!
  - AnalogClock’s have a face; DigitalClock’s don’t.

- A PreciseDigitalClock is a DigitalClock?
  - Always!
  - A PreciseDigitalClock is a DigitalClock that includes seconds.

- A DigitalClock is a PreciseDigitalClock?
  - Sometimes!
  - Not all DigitalClocks have seconds, but those that do are PreciseDigitalClocks.

- A Clock is a DigitalClock?
  - Sometimes!
  - Not all Clocks have DigitalClock features, but those that do are DigitalClocks.
For each of the following, is it **always**, **sometimes**, or **never** true:

- A DigitalClock is a Clock?
  
  **Always!** A DigitalClock is a type of Clock with digital features.

- An AnalogClock is a DigitalClock?

- A PreciseDigitalClock is a DigitalClock?

- A DigitalClock is a PreciseDigitalClock?

- A Clock is a DigitalClock?
For each of the following, is it **always**, **sometimes**, or **never** true:

- A DigitalClock is a Clock?
  
  **Always!** A DigitalClock is a type of Clock with digital features.

- An AnalogClock is a DigitalClock?
  
  **Never!** AnalogClock's have a face; DigitalClock's don't.

- A PreciseDigitalClock is a DigitalClock?

- A DigitalClock is a PreciseDigitalClock?

- A Clock is a DigitalClock?
For each of the following, is it **always**, **sometimes**, or **never** true:

- A DigitalClock is a Clock?
  - **Always!** A DigitalClock is a type of Clock with digital features.

- An AnalogClock is a DigitalClock?
  - **Never!** AnalogClock's have a face; DigitalClock's don't.

- A PreciseDigitalClock is a DigitalClock?
  - **Always!** A PreciseDigitalClock is a DigitalClock that includes seconds.

- A DigitalClock is a PreciseDigitalClock?

- A Clock is a DigitalClock?
For each of the following, is it **always**, **sometimes**, or **never** true:

- A DigitalClock is a Clock?
  - **Always!** A DigitalClock is a type of Clock with digital features.

- An AnalogClock is a DigitalClock?
  - **Never!** AnalogClock’s have a face; DigitalClock’s don’t.

- A PreciseDigitalClock is a DigitalClock?
  - **Always!** A PreciseDigitalClock is a DigitalClock that includes seconds.

- A DigitalClock is a PreciseDigitalClock?
  - **Sometimes!** Not all DigitalClocks have seconds, but those that do are PreciseDigitalClocks.

- A Clock is a DigitalClock?
Clock Hierarchy

Clock Hierarchy Diagram

DigitalClock
- Digital Display ("5:45am")
- Optional Military Time ("11:05pm" or "23:05")

PreciseDigitalClock
- Everything a Digital Clock can do...
- Seconds Display ("5:45:11am")

AnalogClock
- "Clock Face Display"
- Hour Hand/Minute Hand

For each of the following, is it always, sometimes, or never true:
- A DigitalClock is a Clock?
  *Always!* A DigitalClock is a type of Clock with digital features.
- An AnalogClock is a DigitalClock?
  *Never!* AnalogClock's have a face; DigitalClock's don't.
- A PreciseDigitalClock is a DigitalClock?
  *Always!* A PreciseDigitalClock is a DigitalClock that includes seconds.
- A DigitalClock is a PreciseDigitalClock?
  *Sometimes!* Not all DigitalClocks have seconds, but those that do are PreciseDigitalClocks.
- A Clock is a DigitalClock?
  *Sometimes!* Not all Clocks have DigitalClock features, but those that do are DigitalClocks.
public class AnalogClock extends Clock {
    public static final int NUM_HOURS = 12;
    public static final int NUM_MINUTES = 60;

    public double getHourHandAngle() {
        return 360 * ((double) (this.getHour() % 12) / NUM_HOURS);
    }

    public double getMinuteHandAngle() {
        return 360 * ((double) this.getMinute() / NUM_MINUTES);
    }

    public String getTime() {
        return "Hour Hand: " + this.getHourHandAngle() + ", ", " + "Minute Hand: " + this.getMinuteHandAngle() + ", ";
    }
}

AnalogClock vs. Clock

- Is an AnalogClock a Clock?
- What is different about an AnalogClock?
AnalogClock Class

Class

```java
public class AnalogClock extends Clock {
    public static final int NUM_HOURS = 12;
    public static final int NUM_MINUTES = 60;

    public double getHourHandAngle() {
        return 360 * ((double) (this.getHour() % 12) / NUM_HOURS);
    }

    public double getMinuteHandAngle() {
        return 360 * ((double) this.getMinute() / NUM_MINUTES);
    }

    public String getTime() {
        return "Hour Hand: " + this.getHourHandAngle() + ", ", " + "Minute Hand: " + this.getMinuteHandAngle() + ";
    }
}
```

AnalogClock vs. Clock

- Is an AnalogClock a Clock?
  Always! An AnalogClock is a Clock with extra features.

- What is different about an AnalogClock?
public class AnalogClock extends Clock {
    public static final int NUM_HOURS = 12;
    public static final int NUM_MINUTES = 60;

    public double getHourHandAngle() {
        return 360 * ((double) (this.getHour() % 12) / NUM_HOURS);
    }

    public double getMinuteHandAngle() {
        return 360 * ((double) this.getMinute() / NUM_MINUTES);
    }

    public String getTime() {
        return "Hour Hand: " + this.getHourHandAngle() + ", " + "Minute Hand: " + this.getMinuteHandAngle() + ";";
    }
}

AnalogClock vs. Clock

- Is an AnalogClock a Clock?
  
  Always! An AnalogClock is a Clock with extra features.

- What is different about an AnalogClock?
  - It has new methods: getHourHandAngle, getMinuteHandAngle
  - It “overrides” getTime to do something different
public class AnalogClock extends Clock {
    
    public static final int NUM_HOURS = 12;
    public static final int NUM_MINUTES = 60;
    
    public double getHourHandAngle() {
        return 360 * ((double) (this.getHour() % 12) / NUM_HOURS);
    }
    
    public double getMinuteHandAngle() {
        return 360 * ((double) this.getMinute() / NUM_MINUTES);
    }
    
    public String getTime() {
        return "Hour Hand: " + this.getHourHandAngle() + "%, " + "Minute Hand: " + this.getMinuteHandAngle() + "%";
    }
}

AnalogClock c1 = new AnalogClock();
System.out.println(c1.getTime());
System.out.println(c1.getHourHandAngle());
System.out.println(c1.getMinuteHandAngle());

OUTPUT

>> Hour Hand: 180%, MinuteHand: 60%
>> 180
>> 60
public class AnalogClock extends Clock {
    public static final int NUM_HOURS = 12;
    public static final int NUM_MINUTES = 60;

    public double getHourHandAngle() {
        return 360 * ((double) (this.getHour() % 12) / NUM_HOURS);
    }

    public double getMinuteHandAngle() {
        return 360 * ((double) this.getMinute() / NUM_MINUTES);
    }

    public String getTime() {
        return "Hour Hand: " + this.getHourHandAngle() + ", 
               Minute Hand: " + this.getMinuteHandAngle() + ";";
    }
}

Clock c2 = new AnalogClock();
System.out.println(c2.getTime());
System.out.println(c2.getHourHandAngle());
System.out.println(c2.getMinuteHandAngle());

This doesn’t compile! Java treats c2 like a Clock. The second and third calls don’t make sense for a clock. If we remove the second and third lines, we get:

```
>> Hour Hand: 180%, MinuteHand: 60%
```
public class DigitalClock extends Clock {
    private boolean usingMilitaryTime;
    
    public DigitalClock(boolean usingMilitaryTime) {
        this.usingMilitaryTime = usingMilitaryTime;
    }
    public boolean isMilitaryTime() { return usingMilitaryTime; }
    public int getHour() {
        if (this.isMilitaryTime() || super.getHour() <= 12) {
            return super.getHour();
        } else { return super.getHour() - 12; }
    }
    public String getPeriod() {
        if (this.isMilitaryTime()) { return ""; }
        else if (super.getHour() <= 12) { return "am"; }
        else { return "pm"; }
    }
    public String getTime() {
        return this.getHour() + ":" + this.getMinutes() + this.getPeriod();
    }
}

public class PreciseDigitalClock extends DigitalClock {
    private int second;
    
    public PreciseDigitalClock() { super(false); }
    public int getSecond() { return this.second; }
    public String getTime() {
        return this.getHour() + ":" + this.getMinutes() + ":" + this.getSecond()
            + this.getPeriod();
    }
}
DigitalClock & PreciseDigitalClock

Clock vs. DigitalClock vs. PreciseDigitalClock

- Is a DigitalClock a Clock?

- Is a PreciseDigitalClock a DigitalClock?

- What is different about a DigitalClock (from a Clock)?

- What is different about a PreciseDigitalClock (from a DigitalClock)?
DigitalClock & PreciseDigitalClock

Clock vs. DigitalClock vs. PreciseDigitalClock

- Is a DigitalClock a Clock?
  
  *Always!* A DigitalClock is a Clock with extra features.

- Is a PreciseDigitalClock a DigitalClock?

- What is different about a DigitalClock (from a Clock)?

- What is different about a PreciseDigitalClock (from a DigitalClock)?
Clock vs. DigitalClock vs. PreciseDigitalClock

- Is a DigitalClock a Clock?
  
  *Always*! A DigitalClock is a Clock with extra features.

- Is a PreciseDigitalClock a DigitalClock?

  *Always*! A PreciseDigitalClock is a DigitalClock with extra features.

- What is different about a DigitalClock (from a Clock)?

- What is different about a PreciseDigitalClock (from a DigitalClock)?
Clock vs. DigitalClock vs. PreciseDigitalClock

- Is a DigitalClock a Clock?
  
  **Always!** A DigitalClock is a Clock with extra features.

- Is a PreciseDigitalClock a DigitalClock?
  
  **Always!** A PreciseDigitalClock is a DigitalClock with extra features.

- What is different about a DigitalClock (from a Clock)?
  
  - It has a new constructor
  - It has a new field: usingMilitaryTime
  - It has new methods: getPeriod, isMilitaryTime
  - It “overrides” getTime and getHour to do something different

- What is different about a PreciseDigitalClock (from a DigitalClock)?
Clock vs. DigitalClock vs. PreciseDigitalClock

- Is a DigitalClock a Clock?
  Always! A DigitalClock is a Clock with extra features.

- Is a PreciseDigitalClock a DigitalClock?
  Always! A PreciseDigitalClock is a DigitalClock with extra features.

- What is different about a DigitalClock (from a Clock)?
  - It has a new constructor
  - It has a new field: usingMilitaryTime
  - It has new methods: getPeriod, isMilitaryTime
  - It “overrides” getTime and getHour to do something different

- What is different about a PreciseDigitalClock (from a DigitalClock)?
  - It is missing the one argument constructor
  - It has a new field: second
  - It has a new method: getSecond
  - It “overrides” getTime to do something different
DigitalClock Puzzle #1

```java
DigitalClock c3 = new DigitalClock(false); //hour = 13, minute = 22
System.out.println(c3.getTime());
System.out.println(c3.getHour());
System.out.println(c3.getMinute());
System.out.println(c3.getPeriod());
```

OUTPUT

```
>> 1:22pm
>> 1
>> 22
>> pm
```

DigitalClock Puzzle #2

```java
Clock c4 = new DigitalClock(false); //hour = 13, minute = 22
System.out.println(c4.getTime());
System.out.println(c4.getHour());
System.out.println(c4.getMinute());
System.out.println(c4.getPeriod());
```

This doesn’t compile. Clock doesn’t have a getPeriod method!
DigitalClock Puzzle #3

```java
Clock c4 = new DigitalClock(false); // hour = 13, minute = 22
System.out.println(c4.getTime());
System.out.println(c4.getHour());
System.out.println(c4.getMinute());
```

```
OUTPUT
>> 1:22pm
>> 1
>> 22
```

Notice that Java knows that c4 is actually a DigitalClock.

DigitalClock Puzzle #4

```java
PreciseDigitalClock c5 = new PreciseDigitalClock(); // hour=13, minute=22, second=52
System.out.println(c5.getTime());
System.out.println(c5.getHour());
System.out.println(c5.getMinute());
System.out.println(c5.getSecond());
System.out.println(((DigitalClock)c5.getTime()));
System.out.println(((DigitalClock)c5.getSecond()));
```

```
OUTPUT
>> 1:22:52pm
>> 1
>> 22
>> 52
>> 1:22:52pm
>> This last one is a compilation error. (DigitalClock doesn’t have a getSecond() method)
```
DigitalClock Puzzle #5

```java
DigitalClock c6 = new DigitalClock(); //hour=13, minute=22
System.out.println(((PreciseDigitalClock)c6).getSecond());
System.out.println(((PreciseDigitalClock)c6).getTime());
System.out.println(((PreciseDigitalClock)c6).getSecond());
```

All of these are ClassCastException. A new DigitalClock() is NOT a PreciseDigitalClock

Now, we do the same idea with a mystery problem!
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");
    }
}
Keep the following rules in mind

- If the type on the left doesn't have a method, we can't call it.
- When calling a method, the **version** called is always the **actual type**.
- Casting **up** the tree is the only type that is okay.

What do each of the following do? (error? print what?)

- Snow var2 = **new** Rain();
  - var2.method2();
- Snow var2 = **new** Sleet();
  - var2.method2();
- Snow var2 = **new** Fog();
  - ((Sleet)var2).method2();
- Snow var2 = **new** Rain();
  - ((Sleet) var2).method2();
- Snow var2 = **new** Rain();
  - ((Rain) var2).method1();
- Snow var2 = **new** Rain();
  - ((Sleet) var2).method2();
Mystery Problem #1

Class Diagram

Snow
- created: method2()
- created: method3()

Rain
- created: method1()
- overridden: method2()
- inherited: method3()

Sleet
- overridden: method2()
- overridden: method3()

Fog
- created: method1()
- inherited: method2()
- overridden: method3()

var2 is a Sleet

var2 restricted to a Snow

Snow var2 = new Sleet();
var2.method2();

OUTPUT

>> Sleet 2
>> Snow 2
>> Sleet 3
Snow var2 = new Rain();
var2.method1();

OUTPUT

>> Rain 2
Snow var2 = new Rain();
((Rain) var2).method1();

>> Rain 1
Snow var2 = new Rain();
var2.method2();

OUTPUT

>> Rain 2
Snow var2 = new Rain();
((Sleet) var2).method2();

>> ClassCastException: *Error*
Mystery Problem #6

Class Diagram

Snow
- created: method2()
- created: method3()

Rain
- created: method1()
- overridden: method2()
- inherited: method3()

Sleet
- overridden: method2()
- overridden: method3()

Fog
- created: method1()
- inherited: method2()
- overridden: method3()

var2 is a valid cast!

Snow var2 = new Fog();
((Sleet)var2).method2();

OUTPUT

>> Sleet 2
>> Snow 2
>> Fog 3