

Abstract Classes

by Mr. F

A Tale of Two Class Types

- A **CONCRETE** class is one with precisely defined behaviors
- An **ABSTRACT** class is one where the behaviors are declared but not implemented fully

Example

- A WritingUtensil class should have a write method... but how it writes is not well defined!
 - Lead?
 - Ink?
 - Hammer on stone?

Example

- A Pencil class should have a write method...
and it will be well defined!
- Uses lead

WritingUtensil

```
public abstract class WritingUtensil  
{  
}
```

- Abstract classes have the word `abstract` in their class declaration

WritingUtensil

```
public abstract class WritingUtensil
{
    private int usesLeft;
}
```

- Abstract classes may have fields

WritingUtensil

```
public abstract class WritingUtensil
{
    private int usesLeft;

    public WritingUtensil(int ul) {
        usesLeft = ul;
    }
}
```

- They may have constructors

WritingUtensil

```
public abstract class WritingUtensil
{
    private int usesLeft;

    public WritingUtensil(int ul) {
        usesLeft = ul;
    }

    public int getUsesLeft() {
        return usesLeft;
    }

    public void setUsesLeft(int ul) {
        usesLeft = ul;
    }
}
```

- They may have regular methods

WritingUtensil

```
public abstract class WritingUtensil
{
    private int usesLeft;

    public WritingUtensil(int ul) {
        usesLeft = ul;
    }

    public int getUsesLeft() {
        return usesLeft;
    }

    public void setUsesLeft(int ul) {
        usesLeft = ul;
    }

    public abstract void write();
}
```

- They may have incomplete methods!

WritingUtensil Issues

- You can't do any of the following:

```
WritingUtensil wu = new WritingUtensil(5);
```

```
wu.write();
```

not well defined



abstract idea



The Point

- You can make classes that extend WritingUtensil

```
public class Pencil extends WritingUtensil
{
    public Pencil(int ul) {
        super(ul);
    }

    public void write() {
        setUsesLeft(getUsesLeft()-1);
        //do other stuff
    }
}
```

here we implement
the write method



The Point

- In fact, Pencil **MUST** have a write() method... otherwise you'll get an error!
- Concrete classes cannot have incomplete parts

```
public class Pencil extends WritingUtensil
{
    public Pencil(int ul) {
        super(ul);
    }

    public void write() {
        setUsesLeft(getUsesLeft()-1);
        //do other stuff
    }
}
```

Pencil uses

- You can do all of the following

```
Pencil p = new Pencil(5);
```

```
WritingUtensil wu = new Pencil(6);
```

```
p.write();
```

```
wu.write();
```

Recap

- Abstract classes are just like regular classes except
 - They have abstract in the class declaration
 - They can have instance variables
 - They have constructors
 - They can have regular methods
 - They can have abstract methods
 - You cannot construct them

Recap

- Concrete classes may
 - Extend an abstract class
 - But then they must implement any abstract methods