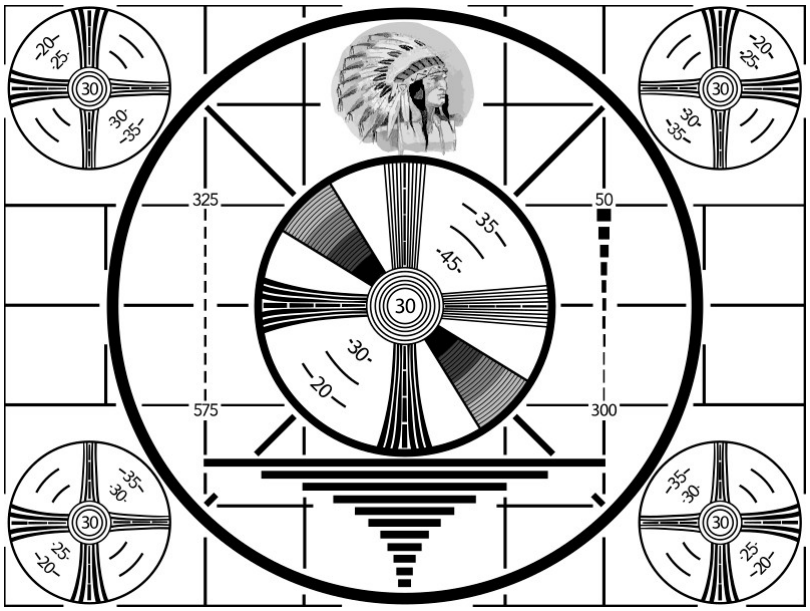
EECS 293 starting soon ...



Lectures are available on canvas

Class Design

<u>Inheritance</u>

- Avoid deep inheritance trees
- Avoid combinatorial explosion in number of subclasses

```
class Cat {
        public void scratch() { ... }
        public void drink() {... }
        public void hunt() { ... }
        ...
}

class ScratchlessCat extends Cat {
        public void scratch() {}    ?
}

public LactoseIntolerantCat extends Cat {
        public void drink() { }
}

public MicelessCat extends Cat {
        public void hunt() {}
}

class ScratchlessLactoseIntolerantCat extends Cat {
        public void scracth() { }
        public void drink() {}
}
```

... all possible combinations ...

Alternative:
boolean variables to denote whether a cat can scratch, ...

```
public void scratch() {
        if (canScratch) { ... }
}
```

Constructors and Builders

- Constructors are not supposed to throw exceptions, but builders can! Ensuring failure atomicity
- With inheritance: factory methods (a builder can return an object of the given type or of <u>any subtype</u>)

Interface

- implementing interface is generally fine because interface deal with outward facing part of classes, not internals therefore not breaking information hiding, encapsulation
- provide something that looks like a default implementation of methods through Skeletal Implementations