One end --> a full class

instance variables

constructors

methods

FULLY SPECIFIED

Other end --> an interface

NO instance variables

NO constructor definitions

NO methods implemented,

JUST REQUIRED METHOD SIGNATURES

Is there some middle ground?

1) Specify some but not all instances

2) Specify how to build a portion of the object but not the whole thing.

3) Specify SOME methods, **but just give method signatures that need to be implemented for others.**

Yes, Java allows us this middle ground via Abstract Classes

Rules for an abstract class:

It's a partial implementation of a class fitting the description above. **You can not instantiate an object from an Abstract Class.**

If you do this, the whole purpose it to write more than one class that inherits from that Abstract Class, but house all of the common items in that Abstract Class

My mammal example from day #1 of inheritance is kind of inaccurate. You can't actually make a real Mammal object, that's just a Mammal. Many creatures inherit like characteristic from Mammal, but there's no such thing as "just a Mammal". We have Cats, Humans, Dogs, Lions, etc. **each of these extends Mammal.**

Mammal can house the common functionality, so that we don't have to recode it in all of the classes that inherit from it.

Then we looked at Pacman

Piece is abstract must implement

public abstract char getCode(); // code for the 2d grid

Instance variables in the game:

```
// How big the board is. It goes -numX to numX inclusive in x
// it goes from -numY to numY in y, inclusive.
private int numX;
private int numY;

// Player for pacman
private Pacman player;

// Manages all of the pieces on the board.
private int numFoodObj;
private Piece[] food;

// Keeps track of how long we have played and how many dots
we've eaten.
private int timeStep;
private int dotsLeft;
```

How I put you in the grid.
p.loc.getX() is in between -numX and +numX
but valid array indexes are in between 0 and 2*numX
Need to add numX to make the location a valid array index

```
grid[p.loc.getX()+numX][p.loc.getY()+numY]=p.getCode();
```