Designing Classes

Christopher Simpkins
chris.simpkins@gatech.edu

Consider a concrete `Point` data type:

```
public class Point {
  public double x, y;
}
```

and an abstract `Point` data type:

```
public interface Point {
  double getX();
  double getY();
  void setCartesian(double x, double y);
  double getR();
  double getTheta();
  void setPolar(double r, double theta);
}
```

- The concrete `Point` exposes its implementation, the abstract `Point` hides it.
- Abstract `Point` expresses that it take two elements to define a point, concrete `Point` allows x and y to be set independently.
- The abstract `Point` is truly an absraction - its interface expresses the essence of pointness and hides its implementation. Data abstraction isn't just about making instance variables private and providing getters and setters.