Object Oriented Overview

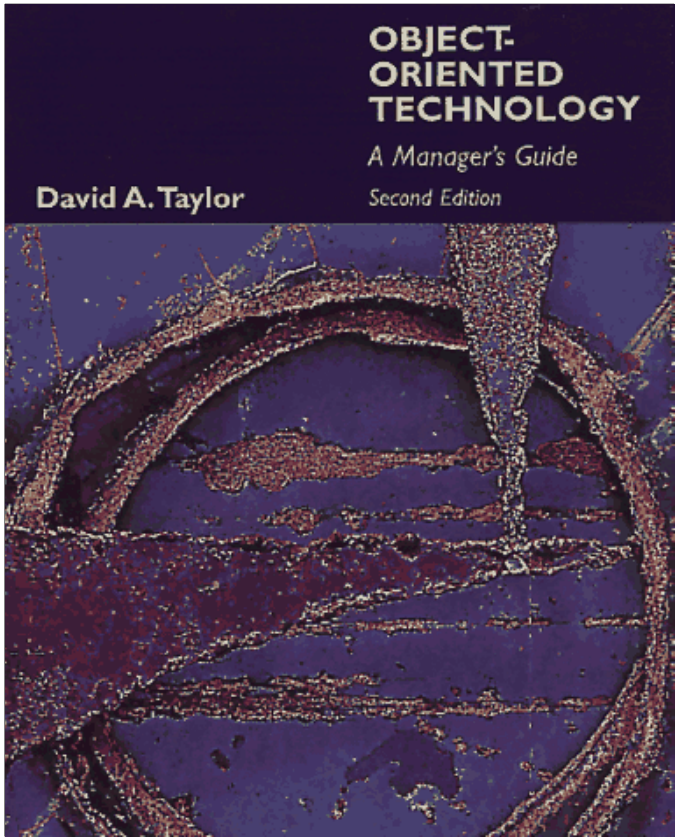
Presented by

Robert Estey

“Anyone can program but can they program Object Oriented?”

Eric Richardson

Object Technology
David A Taylor, Ph.D.



Paperback - 176 pages
2nd edition (September 1997)
Addison-Wesley Pub Co
ISBN: 0201309947

Background

- The Software Crisis
  - Corporations are drowning in data
  - Most software is delivered late/over budget

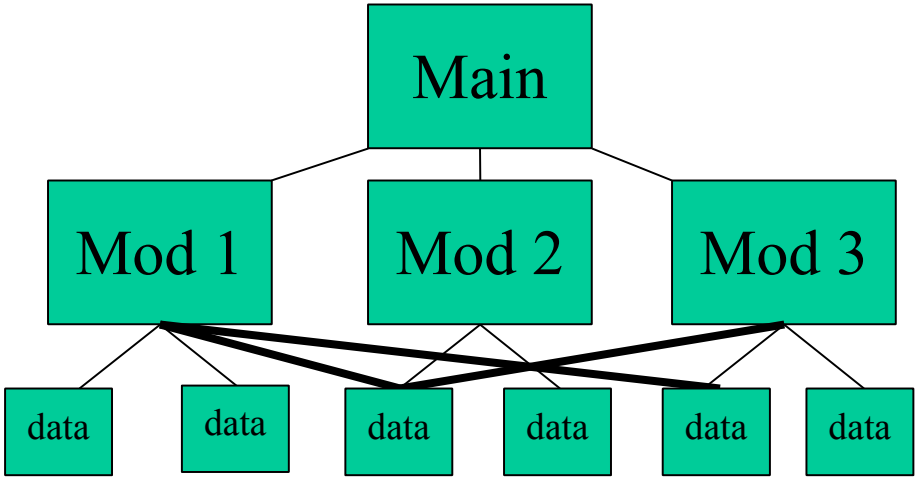
- Building Programs
  - Program
  - Modular Programming, e.g. subroutines

More Background

- Structured Programming
  - Function decomposition
- Computer Aided Software Engineering
  - Manage the process of functional decomposition
- 4th Generation Languages
  - Simple programs and well understood problems

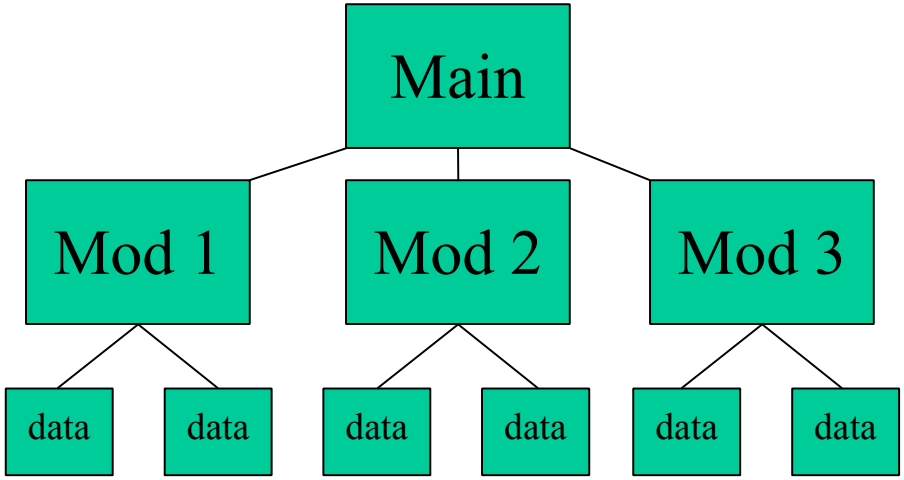
Data Within Programs

Sharing data is a violation of modular programming, which requires that modules be as independent as possible.



Information Hiding

Modularize data along with the procedures.



Models

- Hierarchical Model
- Network Model
- Relational Model
- Object Model

Object Model

- What is the Object Model?
- Simula - 1960's - First Object Oriented Language
- Focus on data in applications

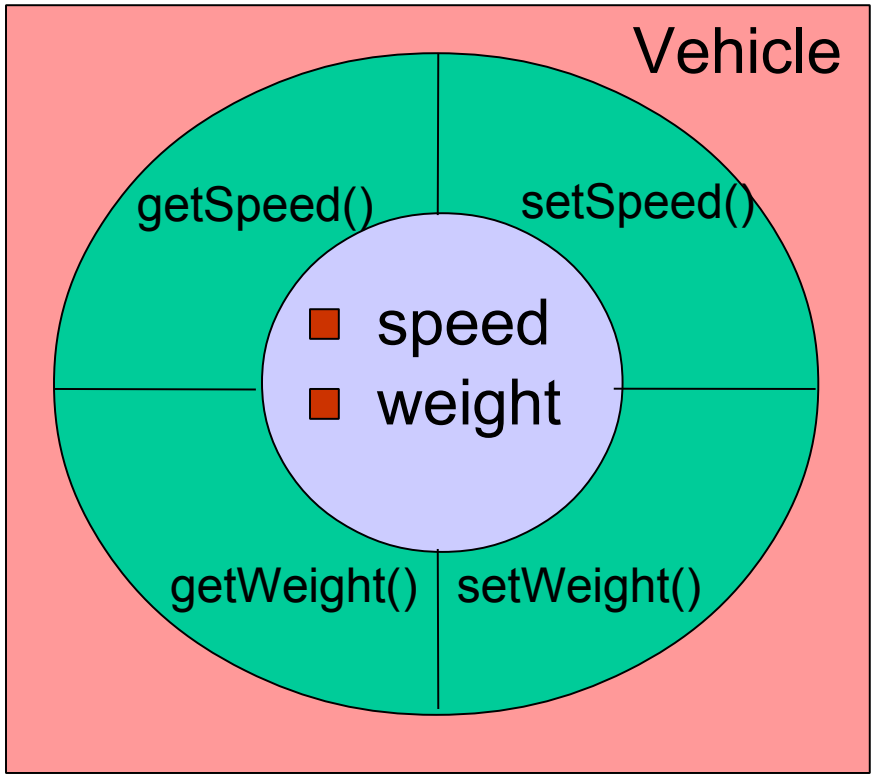
Definitions

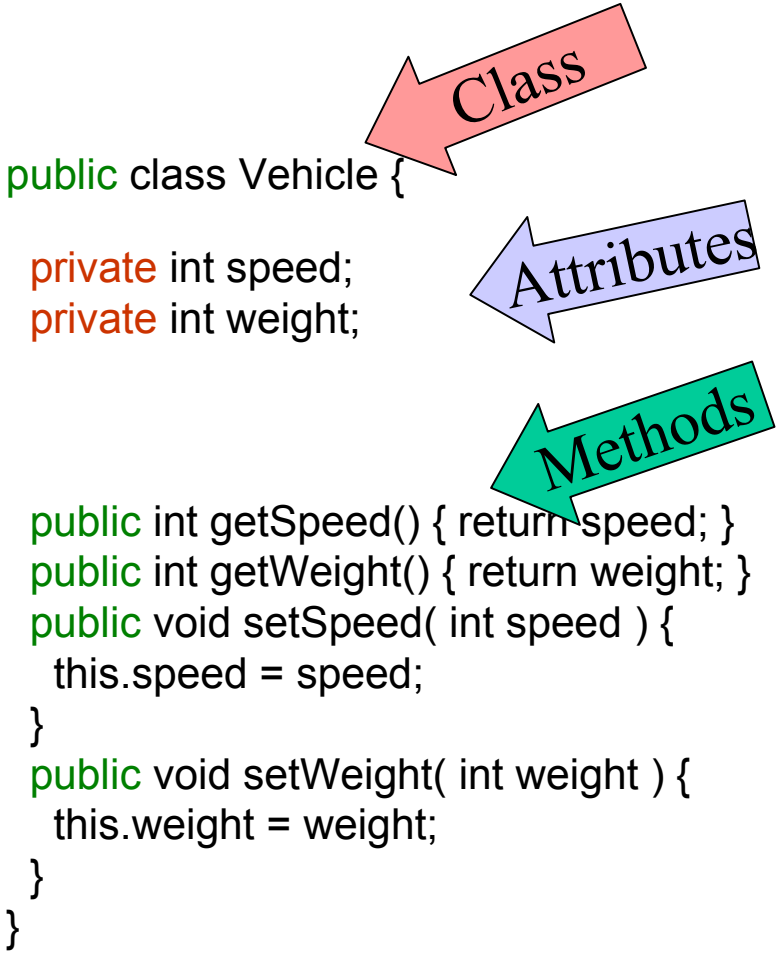
- Class
  - collection of attributes and methods
  - A TYPE, e.g. Lamp, Vehicle
- Object
  - an instance of a class, kitchenLamp, bedroomLamp
- Method (Operation) – functions
  - control attribute state
- Attribute
  - state, variable

Coding Conventions

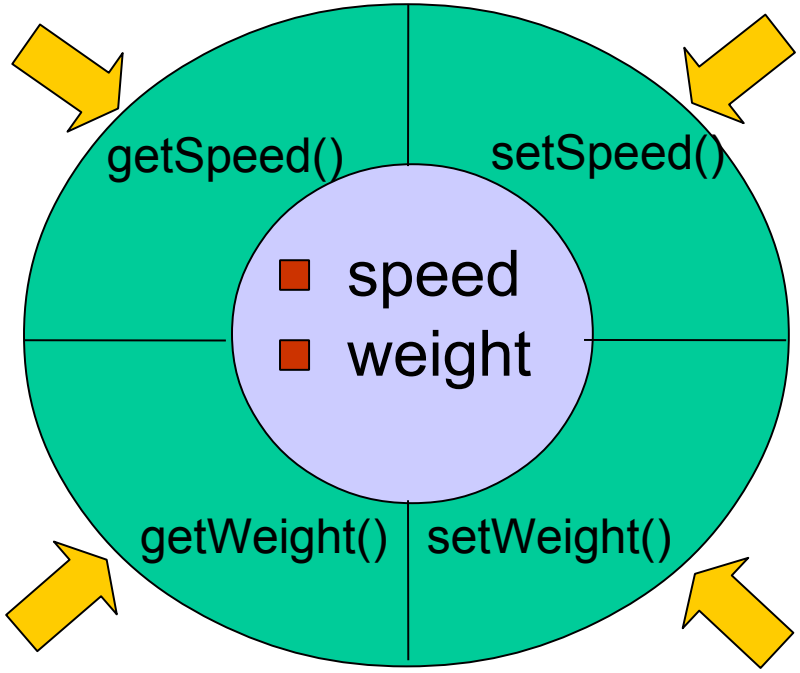
- Classes, Interfaces, Constructors(), Adapters
  - Start with uppercase letter
  - Constructors() match Class name and have ()'s
  - Constructors() have 0 to many inputs
- objects, methods(), attributes
  - Start with lowercase letter
  - methods() do not match Class name and have ()'s
  - methods() have 0 to many inputs
  - methods() have 0 to 1 outputs

Class Anatomy



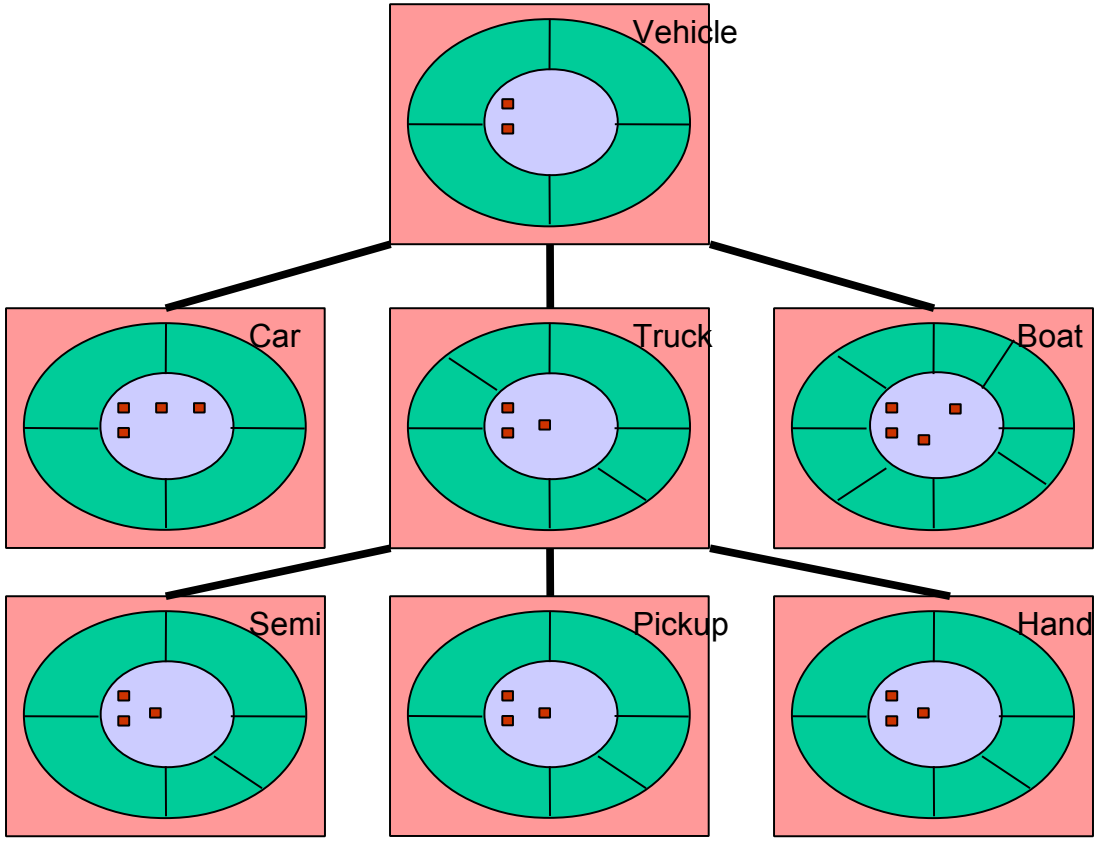


Encapsulation




```
public class Vehicle {

  private int speed;
  private int weight;

  public int getSpeed() { return speed; }
  public int getWeight() { return weight; }
  public void setSpeed( int speed ) {
    this.speed = speed;
  }
  public void setWeight( int weight ) {
    this.weight = weight;
  }
}
```

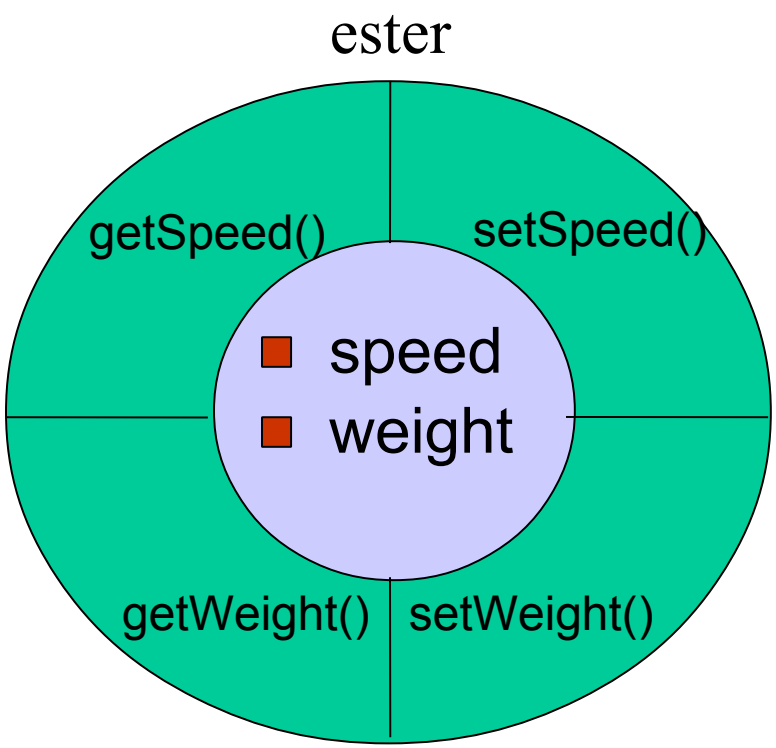
Super and Sub Classes
(Class Hierarchy)

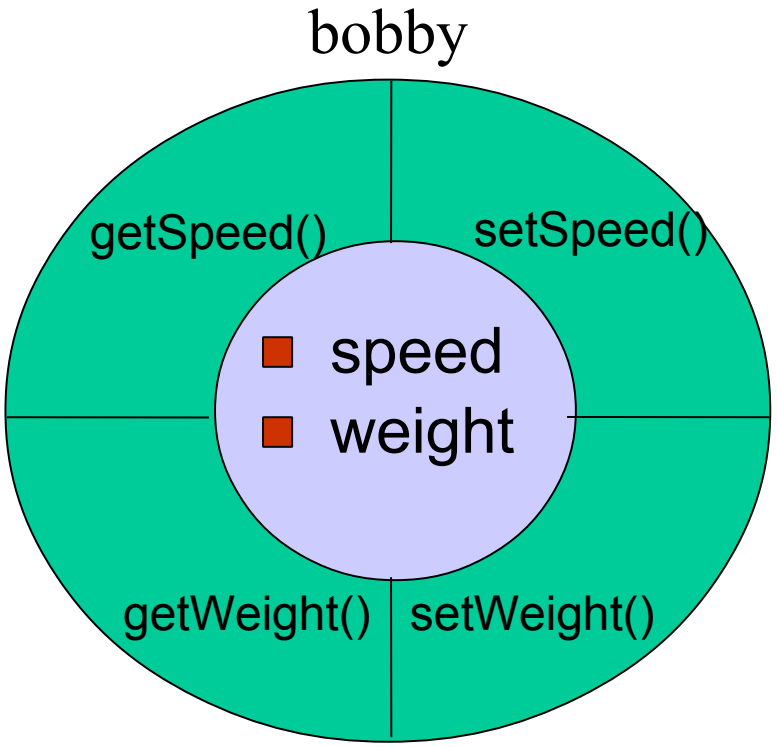


Vehicle is a superclass of Car, Truck and Boat.

Car, Truck and Boat are subclasses of Vehicle.

Objects





Vehicle ester, bobby;

Creating two objects

Constructors

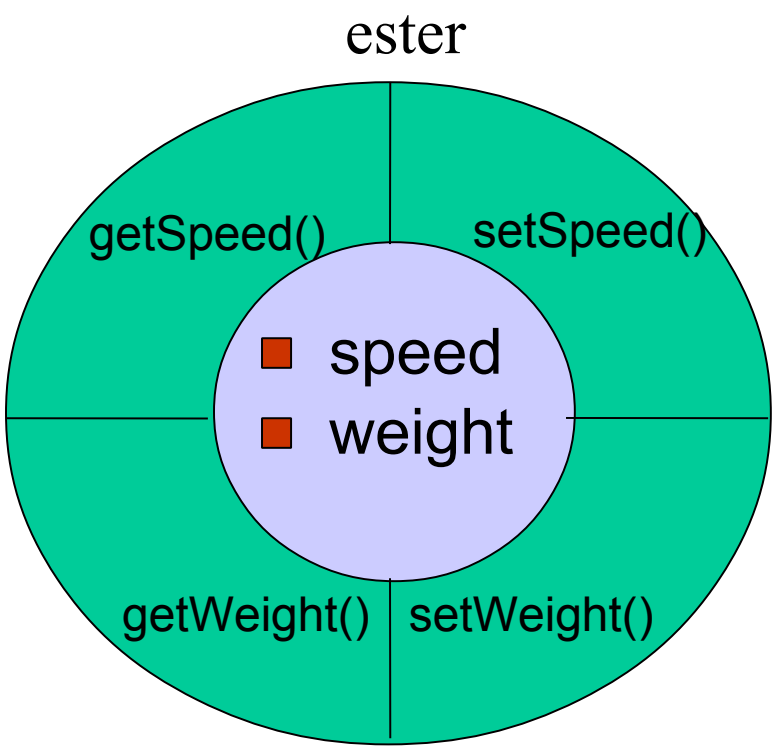
No Argument Constructor

One Argument Constructor

Multiple Constructors

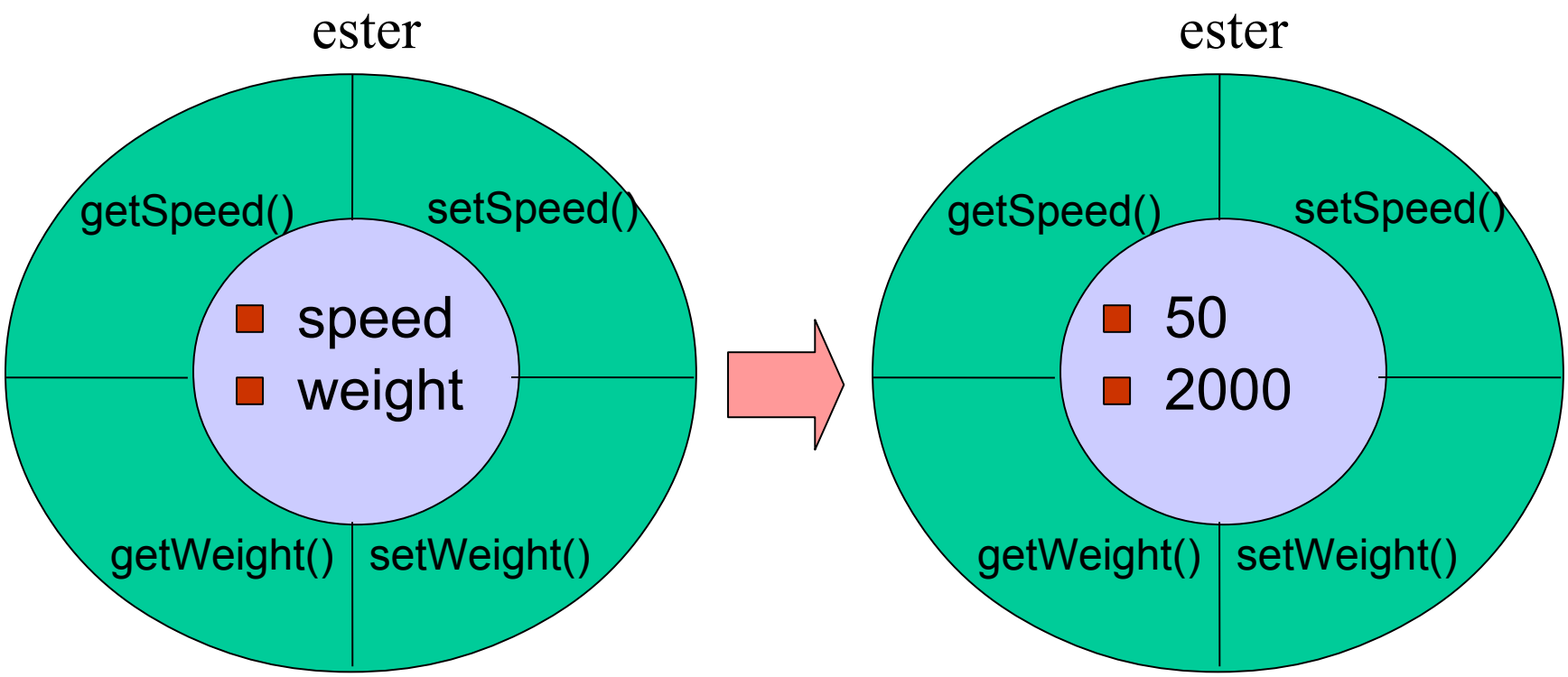
```
public class Vehicle {
  public Vehicle() {
    this.weight = 2000;
  }

  public Vehicle( int speed ) {
    this();
    this.speed = speed;
  }

  private int speed;
  private int weight;

  public int getSpeed() { return speed; }
  public int getWeight() { return weight; }
  public void setSpeed( int speed ) {
    this.speed = speed;
  }
  public void setWeight( int weight ) {
    this.weight = weight;
  }
}
```

Initializing Objects (Constructors)




```
// Builds symbol table no memory allocated
Vehicle ester;
//Creates an object with no argument constructor
ester = new Vehicle();
//Creates an object with one argument constructor
ester.setSpeed( 50 );
//Three steps in one
Vehicle ester = new Vehicle( 50 );
```

Initializing Objects (Constructors)



Robert Estey
07 April 1999

Truck Class
(Subclass Vehicle)

```
public class Vehicle {
  public Vehicle() {
    this.weight = 2000;
  }

  public Vehicle( int speed ) {
    this();
    this.speed = speed;
  }

  private int speed;
  private int weight;

  public int getSpeed() { return speed; }
  public int getWeight() { return weight; }
  public void setSpeed( int speed ) {
    this.speed = speed;
  }
  public void setWeight( int weight ) {
    this.weight = weight;
  }
}
```

```
public class Truck extends Vehicle {
  public Truck() {}

  private boolean fwd = false;

  public boolean isFourWheel() { return fwd; }

  public void setFourWheel( boolean  fwd ) {
    this.fwd = fwd;
  }
}
```

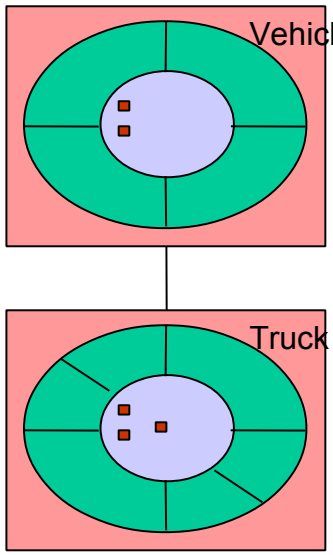
Example Class

```
public class Example {
  public static void main( String[] args ) {

    Vehicle geo = new Vehicle();
    Truck ram = new Truck();
    Truck dakota = new Truck();

    geo.setSpeed( 50 );
    ram.setSpeed( 100 );
    ram.setFourWheel( true );
    dakota.setSpeed( 70 );

    ...
}
```

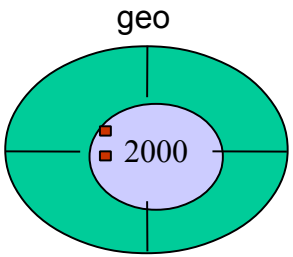
Output

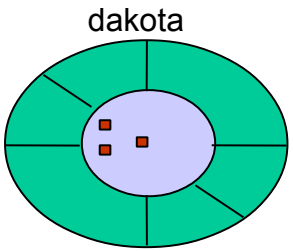
```
geo Speed: 50
ram Speed:     100
ram 4WD:       true
dakota Speed: 70
```

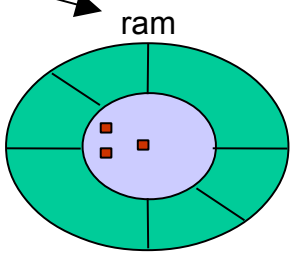
Example Class

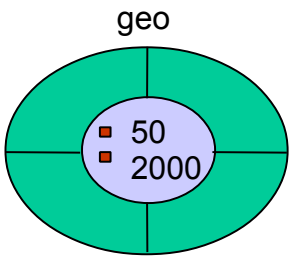


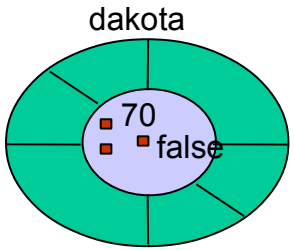

```
Vehicle geo = new Vehicle();
Truck ram = new Truck();
Truck dakota = new Truck();

geo.setSpeed( 50 );
ram.setSpeed( 100 );
ram.setFourWheel( true );
dakota.setSpeed( 70 );
```

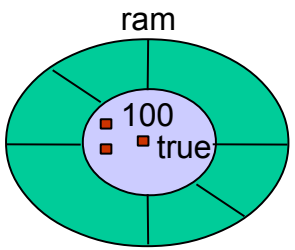




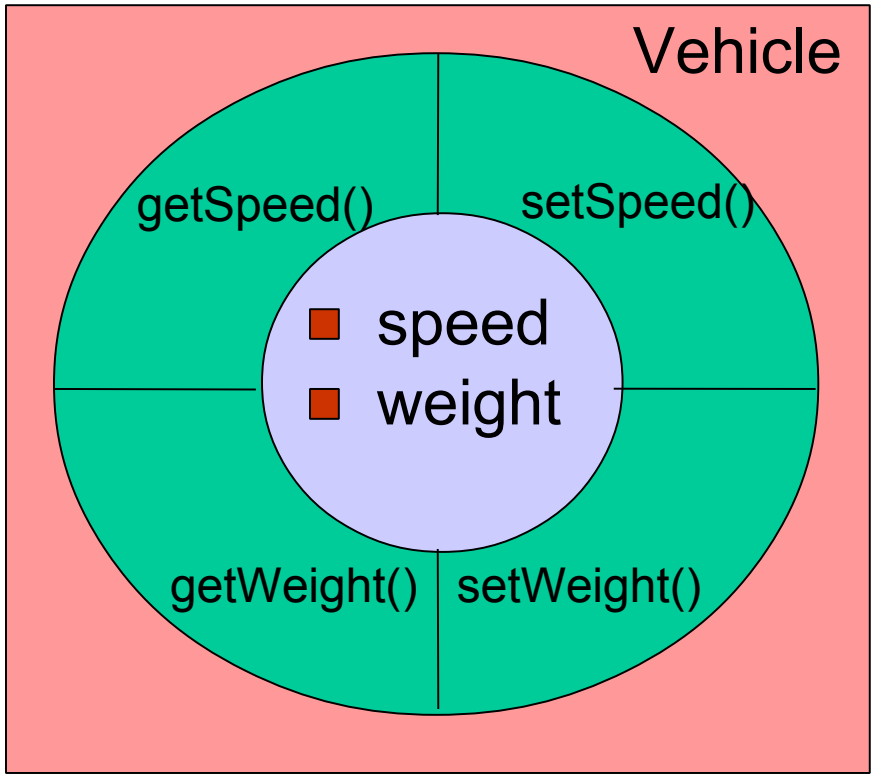


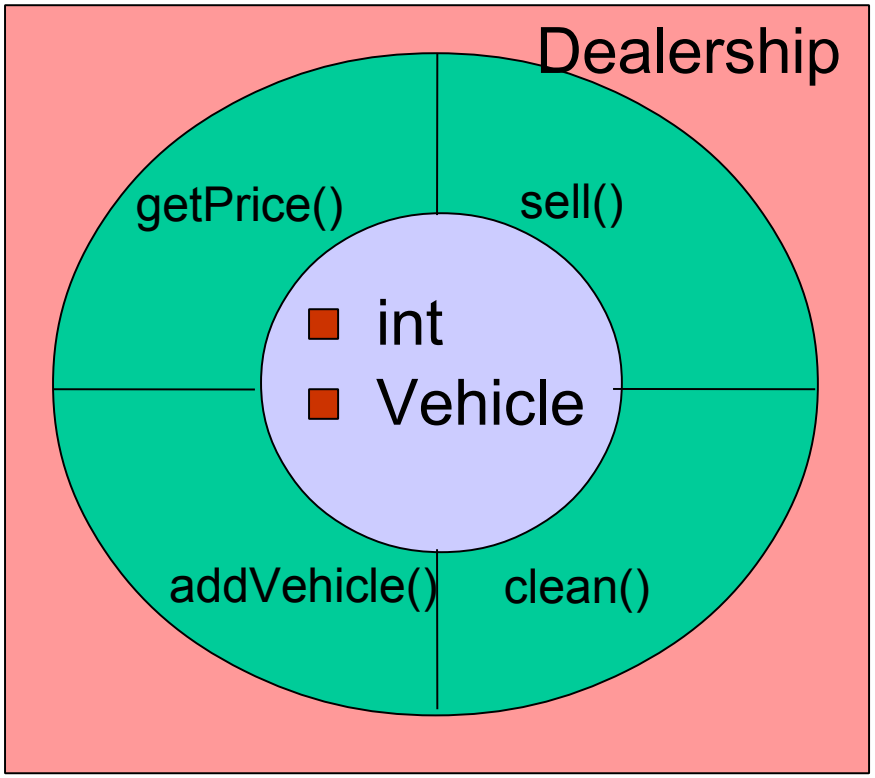




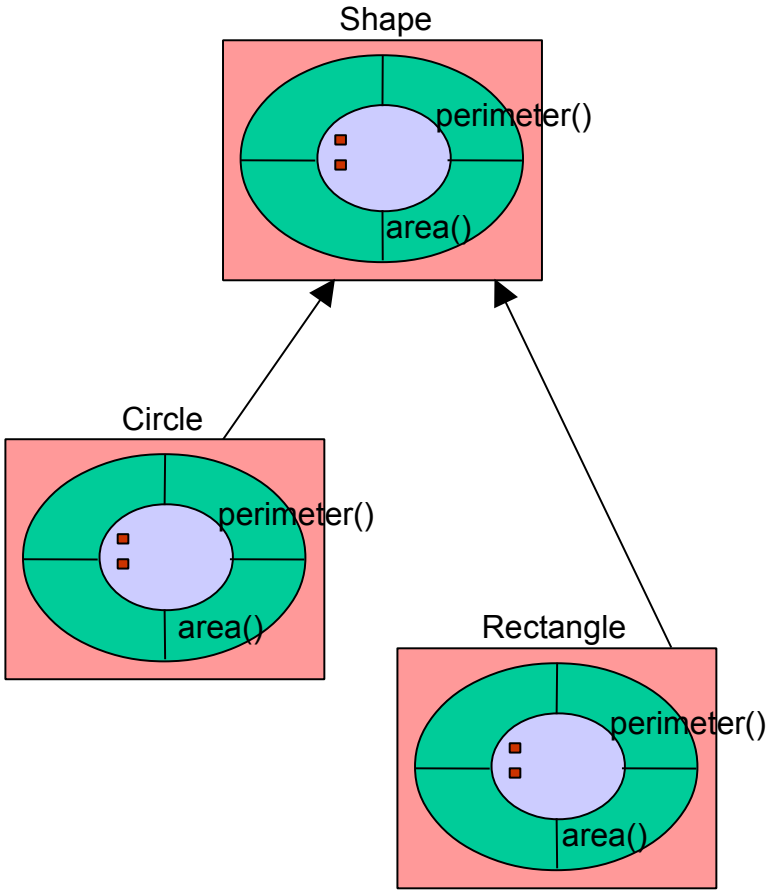


Abstract Types





Abstract Classes




```
public abstract class Shape {
  public double area() { return 0.0; }
  public abstract double perimeter();
}

public class Line extends Shape {
  private double length;
  public Line( double length ); { this.length = length; }
  public double perimeter() { return length; }
}

class Circle extends Shape {
  protected double r;
  protected static final double PI = 3.14
  public Circle( double r ) { this.r = r; }
  public double area() { return PI * r * r; }
  public double perimeter() { return 2 * PI * r; }
  public double circumference() { return perimeter(); }
  public double getRadius() { return r; }
}

class Rectangle extends Shape {
  protected double w, h;
  public Rectangle( double w, double h  ) { this.w = w; this.h = h; }
  public double area() { return w * h; }
  public double perimeter() { return 2 * ( w + h ); }
  public double getWidth() { return w; }
  public double getHeight(0 { return h; }
}
```

Polymorphism

Robert Estey
07 April 1999

Overriding / Overloading

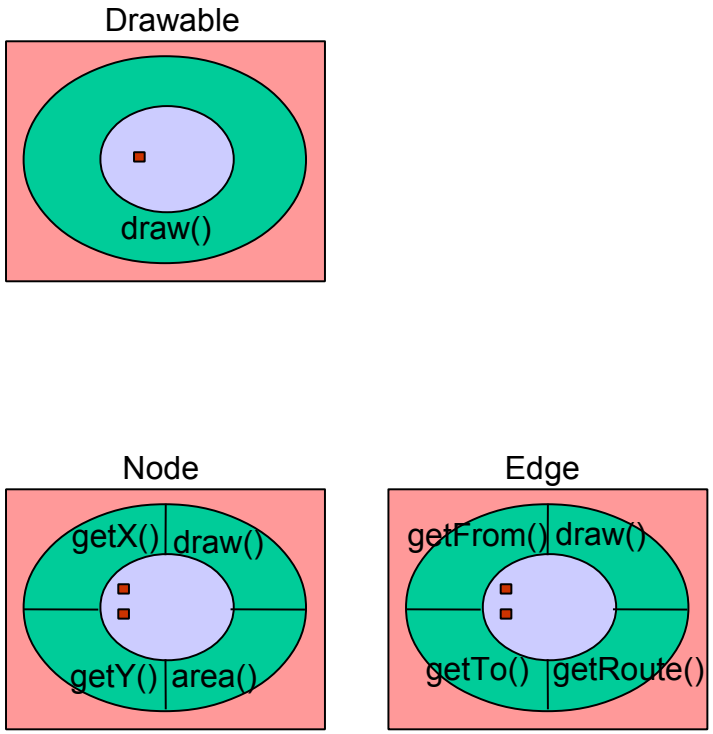
Overriding

```
pubic class SuperClass {
  public method1();
}

public class SubClass extends SuperClass {
  public method1() {
    do something else
  }
}
```

Overloading

```
pubic class SuperClass {
  public method1();
  public method1( int number );
  public method1( int number, float number );
}
```

Interface




```
public interface Drawable {
  void draw( Graphics g );
}

public class Node implements Drawable {
  public void draw( Graphics g ) {
    drawArc( g );
  }
}

public class Edge implements Drawable {
  public void draw( Graphics g ) {
    g.setColor( getColor() );
    g.drawString( Integer.toString( route.getPriority(), x, y );
  }
}
```

OO Overview Exam

```
1 public class Company {

2   private int numberOfEmployees;
3   private int stockPrice;

4  public setStockPrice( int price );
5    this.stockPrice = price;

6  public setNumberOfEmployees( int numberOfEmployees ) {
7    this.numberOfEmployees = numberOfEmployees;
8  }
9 }

*1 Company company = new Company();
*2 company.setStockPrice( 78 );
*3 company.setNumberOfEmployees( 140000 );
```

Robert Estey
07 April 1999

OO Overview Exam

Class

Attribute

Attribute

Method

Method

Object

```
1 public class Company {

2   private numberOfEmployees;
3   private stockPrice;

4  public setStockPrice( int price );
5    this.stockPrice = price;

6  public setNumberOfEmployees( int numberOfEmployees ) {
7    this.numberOfEmployees = numberOfEmployees;
8  }
9 }

*1 Company company = new Company();
*2 company.setStockPrice( 78 );
*3 company.setNumberOfEmployees( 140000 );
```

Robert Estey
07 April 1999