More Classes

Review

More OO Concepts

Old Car and CarTest Classes

New Improved Car Class

- Use this keyword in constructor
- Maintain class invariants
- Add a toString() method
- Add an equals() method

New Improved Book and Person Classes

Lecture notes are based on the textbook *Building Java Programs* by Reges & Stepp.

Anatomy of a Class

- A class is like a *blueprint* or *recipe* that is used to create objects of a particular type.
- A class contains
  - Instance Variables (Fields)
  - Constructor(s)
  - (Instance) Methods

Anatomy of a Class (cont.)

- Instance variables (fields)
  - Usually declared as *private*

Anatomy of a Class (cont.)

- Constructor(s)
  - A special method with the *same name as the clas*s and *NO return type*
  - Initializes the state of new objects as they are created:
    - Generally sets the values of instance variables

      NOTE: Java automatically sets values to “zero equivalent”
    - May do other tasks
  - NOTE: Java provides a *default (null) constructor* with NO arguments, if we *don’t provide* a constructor.

Anatomy of a Class (cont.)

- (Instance) Methods
  - NO static keyword!
  - May be declared as *public* or *private*
  - Mutator – An instance method that *modifies* an object's internal state.
  - Accessor – An instance method that *provides information* about the state of an object *without* modifying it.
  - Getters and Setters – Instance methods that *get* (return) / *set* the values of instance variables.
  - Predicate Method – Returns *true* or *false*

More OO Concepts

- Encapsulation – Hiding the implementation details of an object from the clients of the object.

More OO Concepts (cont.)

- Abstraction – Focusing on essential properties rather than inner details.

More OO Concepts (cont.)

- Code reuse – The practice of writing program code once and using it in many contexts.

More OO Concepts (cont.)

- Class Invariant – An assertion about an object's state that is true for the lifetime of the object.

More OO Concepts (cont.)

- Implicit Parameter – The object being referenced during an instance method call.
- this – A Java keyword that allows you to refer to the implicit parameter inside a class.

More OO Concepts (cont.)

- We often override (rewrite) these default class instance methods provided by Java
  - public String toString()
  - public boolean equals(Object o)
    - Uses instanceof keyword to determine if o is an “instance of” this class
    - If o *is* an instance of this class, *casts* o to an object of this class

```
//Our old Car class:
public class Car {

        //Instance Variables
        private String make;
        private double fuelCapacity;
        private double milesPerGallon;
        private double gallonsInTank;

        //Constructor
        public Car(String m, double fc, double mpg) {
                make = m;
                fuelCapacity = fc;
                milesPerGallon = mpg;
        }
```

```
// Accessor Methods (“getter Methods”)
public String getMake() {
        return make;
}
public double getFuelCapacity() {
        return fuelCapacity;
 }
public double getMilesPerGallon() {
        return milesPerGallon;
}
public double getGallonsInTank() {
        return gallonsInTank;
}
```

```
        // Mutator Methods
        public void fillTank() {
                gallonsInTank = fuelCapacity;
        }
        public void drive(int miles) {
                if (miles > 0) {
                        double gasNeeded = miles/milesPerGallon;
                        if (gallonsInTank > gasNeeded) {
                                gallonsInTank -= gasNeeded;
                        }
                        else {
                                gallonsInTank = 0;
                        }
                }
        }
}
```

```
//Our old CarTest class:
public class CarTest {

        public static void main(String[] args) {

                Car myCar = new Car("Honda", 15, 50);

                System.out.println(myCar.getMake() + " " +
                                        myCar.getFuelCapacity() + " gal. " +
                                        myCar.getMilesPerGallon() + " mpg. ");

                myCar.fillTank();
                System.out.println(myCar.getGallonsInTank() + " gal.");

                myCar.drive(50);
                System.out.println(myCar.getGallonsInTank() + " gal.");
```

```
        Car bobsCar = new Car("Ford", 20, 15);
        System.out.println(bobsCar.getMake() + " " +
                                bobsCar.getFuelCapacity() + " gal. " +
                                bobsCar.getMilesPerGallon() + " mpg. ");

        bobsCar.fillTank();
        System.out.println(bobsCar.getGallonsInTank() + " gal.");

        bobsCar.drive(50);
        System.out.println(bobsCar.getGallonsInTank() + " gal.");

    }
}
```

New Improved Car Class

- Use this keyword in constructor
- Maintain class invariants
- Add a toString() method
- Add an equals() method

```
public class Car {
        private String make;
        private double fuelCapacity;
        private double milesPerGallon;
        private double gallonsInTank;

        public Car (String make, double fuelCapacity,
                    double milesPerGallon) {
                if (fuelCapacity <= 0 || milesPerGallon <= 0) {
                        throw new IllegalArgumentException();
                 }
                this.make = make;
                this.fuelCapacity = fuelCapacity;
                this.milesPerGallon = milesPerGallon;
        }
```

```
public String toString() {
        String s = "";
        s += "Make: " + make;
        s += "  Fuel Capacity: " + fuelCapacity;
        s += "  MPG: " + milesPerGallon;
        s += "  Gallons in Tank: " + gallonsInTank;
        return s;
}
```

```
public boolean equals(Object o) {
        if (o instanceof Car) {
                Car other = (Car)o;
                if (make.equals(other.make) &&
                fuelCapacity == other.fuelCapacity &&
                milesPerGallon == other.milesPerGallon &&
                gallonsInTank == other.gallonsInTank) {
                        return true;
                }
                else {
                        return false;
                }
        }
        else {
                return false;
        }
}
```

```
public class CarTest {

        public static void main(String[] args) {

                Car myCar = new Car("Honda", 15, 50);

                System.out.println(myCar.toString());

                System.out.println(myCar);

                Car friendsCar = new Car("Honda", 15, 50);

                System.out.println(myCar.equals(friendsCar));

                Car bobsCar = new Car("Ford", -20, 15);

        }
}
```

CarTest output:

```
Make: Honda  Fuel Capacity: 15.0  MPG: 50.0  Gallons in Tank: 0.0
Make: Honda  Fuel Capacity: 15.0  MPG: 50.0  Gallons in Tank: 0.0
true
Exception in thread "main" java.lang.IllegalArgumentException
        at Car.<init>(Car.java:11)
        at CarTest.main(CarTest.java:13)
```

New Improved Book and Person Classes

- Use this keyword in constructor
- Maintain class invariants
- Add a toString() method
- Add an equals() method
- Test improvements using the BookTest and PersonTest classes