Classes and Objects

Creating class methods

- Class methods are associated with the class, rather than individual object.
- They have the keyword `static` in their heading.
- For example, a class method for the `Fraction` class that multiply two objects of type `Fraction` and return the product in a `Fraction`

```
public static Fraction times (Fraction f1, Fraction f2)
{
     Fraction result = new Fraction();
     result.num = f1.num * f2.num;
     result.den = f1.den * f2.den;
     return result;
}
```

- From another class method within the same class, a class method should be called using the form `<method identifier>(<parameter list>)`, e.g.,

```
f = times(g,h);
```

- From anywhere else, a class should be called using the form <class identifier>.<method identifier>(<parameter list>), e.g.,

```
f = Fraction.times(g,h);
```

Class Fields

- Class fields are associated with the class, rather than individual objects
- There is only a single copy of a class field but there is a copy of each instance field for every instance of an object
- If one object changes the value of a class field, all other objects of the same class will see the change.
- A class field is created by using the modifier `static` in its declaration.
- For example, assume all cars use the same grade of gasoline, the price will be that same for all cars. Therefore in the `Car` class, the `gasPrice` should be a class field.

```
Class Car {
     private double distance;
     private static double gasPrice;
}
```

- Class fields are often used for constants associated with a class, e.g.,

```
Class Elevator {
     public static final int CAPACITY = 1800;
     // max load in kg
}
```

- Within the same class, class fields are referred by `<field identifier>`
- From a different class, class fields (that are not private) are referred by `<class identifier>.<field identifier>`