Classes Continued

Michael Wagner

Static Fields

```
public class SomeClass
{
    private static int num1 = 0;
    private int num2 = 0;

    public SomeClass()
    {
        num1++;
        num2++;
    }

    public int getNum1()
    {
        return num1;
    }

    public int getNum2()
    {
        return num2;
    }
}
```

```
public class SomeClassDemo
{
    public static void main(String[] args)
    {
        SomeClass test1 = new SomeClass();
        SomeClass test2 = new SomeClass();
        SomeClass test3 = new SomeClass();

        System.out.println(test1.getNum1());
        System.out.println(test1.getNum2());

        System.out.println(test2.getNum1());
        System.out.println(test2.getNum2());

        SomeClass test4 = new SomeClass();

        System.out.println(test3.getNum1());
        System.out.println(test3.getNum2());
    }
}
```

Example

Write a class called NetflixMovie that limits its instance count to 3.

Static Methods

```
public class Money
{
    public static double dollarsToEuros(double dollars)
    {
        return dollars * 0.7665;
    }

    public static double eurosToDollars(double euros)
    {
        return euros / 0.7665;
    }
}
```

```
double result1 = Money.dollarsToEuros(473.54);

double result2 = Money.eurosToDollars(1003.24);
```

Shadowing

```
public class BankAccount
{
    private float balance = 0;

    public float getBalance()
    {
        return balance;
    }

    public void deposit(float balance)
    {
        this.balance = balance;
    }

    public void applyServiceFee()
    {
        balance -= 5;
    }
}
```

Passing Objects

```
public class Demo
{
    public static void processAccount(BankAccount b)
    {
        if (b.getBalance() < 1000)
        {
            b.applyServiceFee();
        }
    }

    public static void main(String[] args)
    {
        BankAccount my_account = new BankAccount();
        my_account.deposit(566.32f);
        processAccount(my_account);
        System.out.print("Current: " + my_account.getBalance());
    }
}
```

toString

toString is implicitly called when concatenating and when calling println.

```
public class BankAccount
{
    private float balance = 0;
    private String name = "";
    private boolean good_standing = true;

    public String toString()
    {
        String result = name + " has " + balance + " dollars.";
        if (!good_standing)
        {
            result += "(Not in good standing.)";
        }
        return result;
    }

    /* .. more methods here ... */
}
```

Example

Create a class called **Stock** that encapsulates

- Ticker name
- Price
- Getters
- A constructor to set name and price
- A toString method

equals

```
public class BankAccount
{
    /* .. more fields .. */

    public String ssn = "";

    public boolean equals(Object other)
    {
        if (!(other instanceof BankAccount))
        {
            return false;
        }

        BankAccount ba = (BankAccount)other;
        return ssn.equals(ba.ssn);
    }

    /* .. more methods here ... */
}
```

Example

Create a class called **Grade**, that represents the usual A-F scale. Aggregate **Grade** with a **Student** class.

- Students have
  - First name, last name, SSN
  - Grade

Include a **toString** method

Include an **equals** method on Student