Classes

Classes

Classes are a way to encapsulate data and the methods that operate on them. We've used lots of classes so far such as Scanner and Random.

All of our programs we've created in the labs are also their own class.

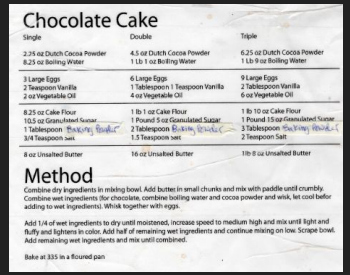
```
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

Classes

A class is a blueprint for making an object. When we used classes such as Random and Scanner, we used the new keyword in order to instantiate an object.

```
Random rand = new Random();
```

```
Scanner scan = new Scanner();
```




```
ChocolateCake cake = new ChocolateCake();
```

Classes

The names of classes have EachWordCapitalized. The name of the file you create needs to be the same name as your class.

```
// ChocolateCake.java

public class ChocolateCake {

}
```

Classes - properties

The variables contained inside a class are called: properties.

```
// ChocolateCake.java

public class ChocolateCake {
    public String cakeType;
    public boolean isFrosted;
    public double price;
}
```

Classes - properties

The functions contained inside a class are called: methods.

```
// ChocolateCake.java

public class ChocolateCake {

    public void print() {
        System.out.println("A lovely cake!");
    }

}
```

Classes

Here are the properties and method together. You'll notice everything is marked with public. This makes the properties and methods available.

```
public class ChocolateCake {
    public String cakeType;
    public boolean isFrosted;
    public double price;

    public void print() {
        System.out.println("A lovely " + cakeType + " cake!");
        if (isFrosted) {
            System.out.println("It's also frosted!");
        }
        System.out.printf("Just $%.2f\n", price);
    }
}
```

Classes

Here is how we would use our ChocolateCake class as it is so far.

```
ChocolateCake cake = new ChocolateCake();

cake.flavor = "Flourless";
cake.isFrosted = true;
cake.price = 20.00;

cake.print();
```

```
A lovely Flourless cake!
It’s also frosted!
Just $20.00
```

Classes

You can use the private keyword to make properties and methods of your class not accessible to other programmers or to prevent accidental changes.

```
public class Grader {
    private double total;

    public void addScore(double score) {
        total = total + score;
    }

    public void print() {
        System.out.printf("Total score $%.2f\n", total);
    }
}
```

Let’s Code

Don’t Forget!

Check the syllabus / schedule for reading assignments and due dates!

Class Initialization

Class Initialization - Properties

You can initialize the properties in your class. When you make a call to new these are set automatically.

```
public class Student {
    public String name = "(unknown)";
    private double total = 0.0;
}
```

```
Student s = new Student();
System.out.println(s.name);
```

```
(unknown)
```

Class Initialization - Constructor

A constructor is a special method with the same name as your class. It is run when your class is instantiated. You can have no parameters such as: `new Random()` or you may pass in any number of parameters such as `new Scanner(System.in)`

```
public class Student {
    public String name;

    public Student() {
        name = "(unknown)";
    }
}
```

```
Student s = new Student();
System.out.println(s.name);
```

```
(unknown)
```

Class Initialization - Constructor

You can overload the constructor - have multiple versions based on what parameters are passed to it.

```
public class Student {
    public String name;

    public Student() {
        name = "(unknown)";
    }

    public Student(String n) {
        name = n;
    }
}
```

```
Student s1 = new Student();
System.out.println(s1.name);

Student s2 = new Student("Harry");
System.out.println(s2.name);
```

```
(unknown)
Harry
```

References

References

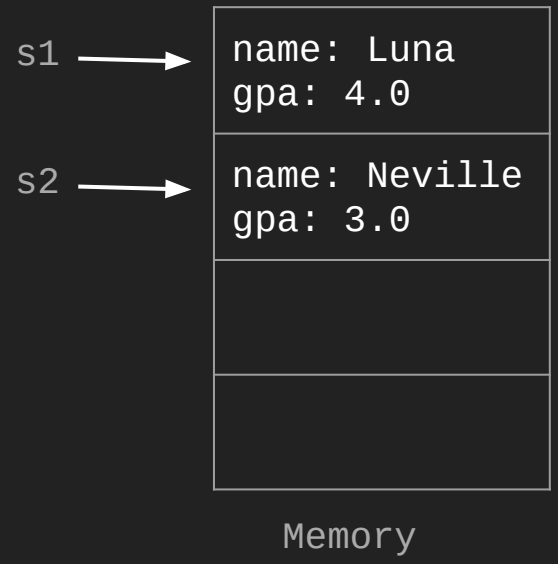
When you instantiate a class, the variable is a pointer to a location in memory.

```
Student s = new Student("Luna");
```

References

Two variables that both use new, will point to different locations in memory.

```
Student s1 = new Student("Luna");
Student s2 = new Student("Neville");
```



References

What's going to happen here?

```
Student s1 = new Student("Luna");
Student s2 = new Student("Neville");

s2 = s1;

s2.name = "Draco";

System.out.println(s1.name);
```

References

Although there will be a place in memory where Neville was, s2 will now point to the same location in memory as s1

```
Student s1 = new Student("Luna");
Student s2 = new Student("Neville");

s2 = s1;

s2.name = "Draco";

System.out.println(s1.name);
```


```
Draco
```

Class, Instantiate, Object, Reference

class	- Your definition / blueprint / recipe - (blueprint for a house)
instantiate	- The act of allocating memory to store a copy of the class "new" - (construct the house)
object	- The instance of the class (what you interact with) - (the actual house)
reference	- The variable that points to the memory with there object is. - (the address of the house)

Let’s Code

Don’t Forget!

Check the syllabus / schedule for reading assignments and due dates!