Classes

Java
Mr. Poole

Methods - Review

Methods do tasks that we want to happen multiple times.

But have we ever wondered what “class starter” does?

```
class starter {

    public static int add(int a, int b){

    }

    public static void main(String args[]) {
        // Your code goes below here

    }
}
```

Java - Object Oriented Programming (OOP)

Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time=

What is a class?

Classes are **templates** to be used multiple times!
Think of them as a generalization of a topic.

Classes allow programmers to store **MULTIPLE pieces of data**
within one variable.

All classes should be made universal, not specific to one object.

Java - Classes vs Objects

Class	Objects
Fruit	Apple Banana Mango

Java - Classes vs Objects

Class	Objects
Car	Honda Toyota BMW

Example: Class vs Object

Car is the Class!

```
public class Car{

}
```

In our main method we can create a Car object!

```
class starter{
    public static void main(String args[]){
        Car a = new Car();
    }
}
```

Example: Class vs Object

Notice!

We can make multiple Car objects using the same Car Class!

```
class starter{
    public static void main(String args[]){
        Car a = new Car();
        Car b = new Car();
        Car c = new Car();
        Car d = new Car();
        Car e = new Car();
        Car f = new Car();
    }
}
```

3 Parts of a Class

3 Parts of a Class!

1. **Global Variables**
    a. Attributes
2. **Constructors**
    a. Set up the class (when its created/born)
3. **Methods**
    a. Actions that use our values!

1. Global Variables

These are attributes that the class should have!

In this case, they are the different features of cars! This would be a list of features that EVERY car has.

1. Global Variables

Example Variables

Class	Global Variables
Car	double price String model String make int year boolean hasAirConditioning

2. Constructor

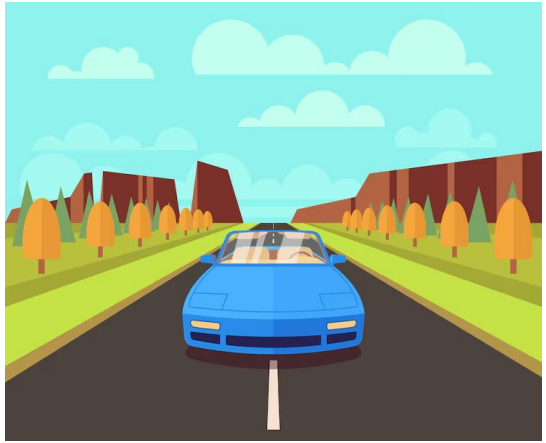
When the object is created, we should set it up for success!

Constructors run at creation and **set up ALL global variables**.

3. Methods

These are the actions of the class. They can use the data within the object to perform some action.

Methods allow us to change, update, and look at our data within the class. Similar to how you use a variable.

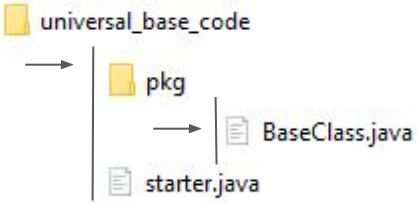


Class Structure

Classes in new files!

New file structure!

- Universal Base Code folder
  - Contains pkg and starter.java
    - pkg contains BaseClass.java



This is necessary for creating and using multiple file!

Starter.java new file structure

Two new things!

1. `import pkg.*;`
   a. This goes above everything!
   b. This command imports all custom Classes

2. `BaseClass test = new BaseClass();`
   a. This is just a basic class object

starter.java

```
import pkg.*;
import java.util.*;

class starter {
    public static void main(String args[]) {
        // Your code goes below here
        BaseClass test = new BaseClass();

    }
}
```

Example Class - BaseClass

1. `package pkg;`
   a. This references the folder pkg to tell the starter.java that this file is part of the imported package.
2. Example Constructor
   a. We'll go into Constructors more later but leave this section and we will expand later

BaseClass.java

```
package pkg;
import java.util.*;

public class BaseClass{
    public BaseClass(){

    }
}
```

NOTE:
The file name, class name, AND constructor all must be named the exact same!

Lab - Classes

No Lab

Read the slides :)

Check out the example classes