The Many Uses of "**static**"

Reminder -> object and instance are two words for the same thing.

As per the text book, static is
a datatype,
a reserved word,
used with methods and variables

Not always easy to understand the different uses

From the glossary in the book:

static type - the static type of a variable is the tpe declared in the source code in the variable declaration section.

Truly enlightening.....

static variable class variable - Exactly on copy exists of a class variable at all times independent of the number of instances (objects) created.

A bit more descriptive but still...

The **static keyword** in java is used for memory management. Use the java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class instead of the instance of the class.

The static can be:

1. variable (also known as class variable)
2. method (also known as class method)
3. nested class

Note: There's also the static block and nested class, but that's for another discussion.

Java static variable

Declare any variable as static:

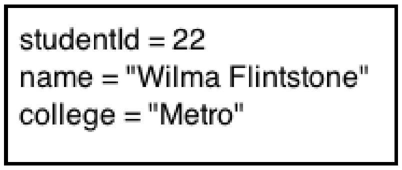
- The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.
- The static variable gets memory only once in class area at the time of class loading.

Example:

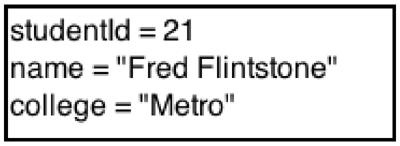
```
    class Student {
         int studentId
         String name;
         String college="Metro";
    }

Student fred = new Student(21, "Fred Flintstone", "Metro");
Student fred = new Student(22, "Wilma Flintstone", "Metro");
```

The objects would be



studentId = 22
name = "Wilma Flintstone"
college = "Metro"



studentId = 21
name = "Fred Flintstone"
college = "Metro"

Printing the object:

```
System.out.println(fred);
```
21 Fred Flintstone Metro

There are about 32,000 students at Metro. Create an object for every student -> 32,000 copies of the string "Metro". Lots of space taken up.

>>> Static property shared by ALL objects of this class. Saves space.

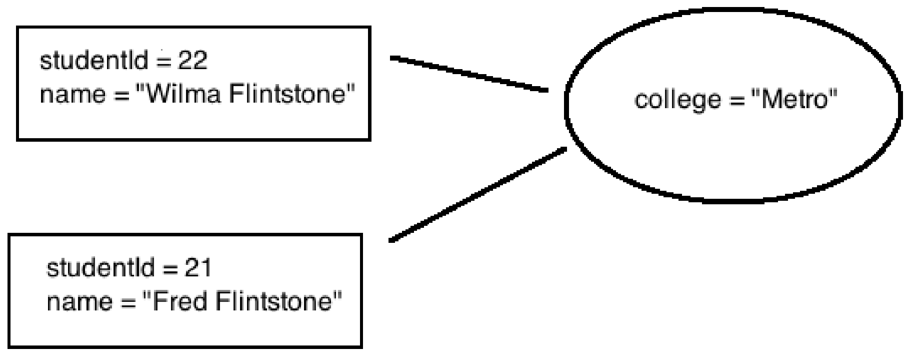
```
    class Student{
         int studentId
         String name;
         static String college="Metro";
    }

Student fred = new Student(21, "Fred Flintstone);
Student fred = new Student(22, "Wilma Flintstone);

System.out.println(fred);
```

21 Fred Flintstone Metro

And the objects in memory would be something like this, each referring to the same static variable:



Java static method

Use static keyword with any method, it is known as static method.

- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance (object) of a class.
- static method can access static data member and can change the value of it.

```
   class Student {
        int studentID;
        String name;
        static String college = "Metro";

        static void change(){
        college = "MSCD";
        }
   }

Student fred = new Student(21, "Fred Flintstone);
System.out.println(fred);
```

   21 Fred Flintstone Metro

```
fred.change();
System.out.println(fred);
```

   21 Fred Flintstone MSCD

Restrictions for static method

There are two restrictions for the static method.

- The static method can not use non static data member or call non-static method directly.
- **this** and **super** cannot be used in static context.

Why the Java main method is static

This is neccesary because **main**() is called by the JVM before any objects are made. Since it is **static** it can be directly invoked via the class.

Java static block

Initialize the static data member.
Executed before main method at the time of classloading.

The ever present "this"

At times, it can be an existential discussion.

From the Oracle documentation:
Within an instance **method** or a **constructor**, `this` is a reference to the *current object* — the object whose method or constructor is being called. You can refer to any member of the current object from within an instance method or a constructor by using `this`.

```
public class addIt {
    public int x = 0;
    public int y = 0;

    //constructor
    public addIt(int a, int b) {
        x = a;
        y = b;
    }
}
```

Using the this keyword, no need to use different parameter names (stupid example, I know):

```
public class addIt {
    public int x = 0;
    public int y = 0;

    //constructor
    public addIt(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

The ***this*** keyword can be used to refer to any member of the current object inside an instance method or a constructor.

More examples:

http://javabeginnerstutorial.com/core-java-tutorial/this-keyword-in-java/