Java Classes, Objects, Constructors, Accessors, and Mutators (quite a mouthful)

The following two .java files are inside the src folder
In Replit we may just create a new file called ToolBox.java.

Source code:
https://replit.com/@andrewJJimenez/AdventurousUnnaturalMetadata#Main.java

```
public class Main {
  public static void main(String[] args) {

    //this calls the constructor
    ToolBox temp = new ToolBox("hello",
12);

    //this calls the accessor

System.out.println(temp.getClassNumber()
);

    //this calls the mutator
    temp.setClassNumber(15);

    //this calls the accessor again

System.out.println(temp.getClassNumber()
);
  }//end main
}//end class
```

```
public class ToolBox {

  //these are class instance variables
  //private because they cannot be seen
  // outside the class
  private String classText;
  private int classNumber;

  //constructor
  public ToolBox (String initString, int initFactor){
    classNumber=initFactor;
    classText=initString;
  }//end constructor ToolBox

  //accessor
  public int getClassNumber(){
    return this.classNumber;
  }//end accessor

  //mutator
  public void setClassNumber(int classNumber) {
    this.classNumber = classNumber;
  }//end mutator
}//end class
```

Things to note:

- An **instance variable** is **a variable which is declared in a class but outside of constructors, methods, or blocks**. Instance variables are created when an object is instantiated, and are accessible to all the constructors, methods, or blocks in the class. Access modifiers can be given to the instance variable

- **public** means that any other class may access this variable or method
- **private** means this variable or method may only be accessed within the class i.e, cannot be seen outside the class. Instance variables are normally declared as private.
- **void** indicates that we are not returning a value. Mutators are usually void as the are just changing the value Accessors of usually not void and must take the type they are returning. See out accessor getClassNumber is returning and integer value and hence is declared as *int* type.
- Our **constructor** creates an object of type ToolBox. ToolBox has instance variables classText and ClassNumber. A constructor must be built to **instantiate** these instance variables upon creation as we doat the beginning of Main.
- **Mutators** allow us to change the instance variables of an object

```
public class Main {
  public static void main(String[] args) {

    //this calls the constructor
    ToolBox temp = new ToolBox("hello", 12);

    System.out.println(temp.TwiceInstanceVar());
    System.out.println(temp.Twice(9));

  }
}
```

```
public class ToolBox {
  //these are class instance variables
  //private because they cannot be seen
  // outside the class
  private String classText;
  private int classNumber;

  //constructor
  public ToolBox (String initString, int initFactor){
    classNumber=initFactor;
    classText=initString;
  }//end constructor ToolBox

  //silly instance var method example
  public int TwiceInstanceVar(){
    return 2*this.classNumber;
  }//end method

  //silly general method for any int passed
  public int Twice(int x){
    return 2*x;
  }//end method
}//end class
```

More things to note:

- We may write class methods. These methods may be primitive types like int or String, however, they may also be void.
- We have two methods here. Once involves an instance variable, however the other takes in a variable and does “work” with it and has nothing to do with any of the object’s instance variables.
    - Consider a built-in sine function. It could be part of a “Math” class, however, it must take input from outside
- Regardless, these methods must be called through the object, almost exclusively by placing a dot on the end of our object.
    - Ex) temp.Twice(9)