Chapter 10 Important Notes : Numbers Matter

- Methods in the Math class don't use any instance variable values. And because the methods are "Static," you don't need to have an instance of Math. All you need is the Math class, so that means the methods in java defined as static cause their using is not need any data from the class object variables right? its only need is the passed arguments to it just that + the class will not store any data inside it , it just class about methods making specific functions and do not affect in the class objects that's why we define the methods static to can call the method directly by using the class name rather than making an instance from it to can use the methods.
- These methods never use instance variables, so their behavior doesn't need to know about a specific object.
- So we call the static methods with using a class name but we call the non static methods (regular methods) by using a reference variable name

- Until now we have 2 ways to prevent class to be instantiated one of them to define the class itself as an abstract class and we try it in chapter 8 and we notice that it is impossible to create an object from an abstract class and the second way we will talk about it today is that by define the constructor of the class as a private constructor and that's what happened inside the math class

- so that class having static methods doesn't mean that you can not create an object from this class cause this class having static methods on it no that is not the reason but the real reason is that the private constructor so you can make non static methods and static ones and put them together in the same class but if you make the constructor public so you can make an object from this class and call any methods by using this object or you can call directly the static methods only and that but using the class name only or if you make the constructor private so you can not make any object and the second option is the only available option for you

- if you try to use an instance variable from inside a static method you will get a compiler error and that's because the compiler do not know which object instance variable you are talking about so if you have 10 objects on the heap the static methods do not know anything about any of them so that means there is a rule that **the static methods can not use any non static methods and that's cause the non static methods they are the methods which treat with the instance**

variables states to affect the behavior of this methods so you can not call any of the non static ones inside the static methods and that's cause the same reason we talk about it

- the book about the second way about if the constructor is public and having static methods so it is legal to create an object from this class and using the dot operator you will call this static methods from inside of it easily but the book tell us it is legal but not readable

- we can too define a static variable inside the class and its use will be to making a value shared by all instances of a class that's mean one value per class instead of one value per instance , shared variables are shared all instance of the same class share a single copy of the static variables

- WHAT ABOUT : IF You want to design a class in Java in such a way that **only one instance of it can ever be created** during the lifetime of the program . Any code that wants to use the class should use **that single instance** . The class should **prevent** outside code from creating additional objects of it . How would you implement this?

```
public class Singleton {
  // Step 1: Create a private static variable to hold the single instance
  private static Singleton instance;

  // Step 2: Make the constructor private to prevent instantiation from outside
  private Singleton() {
    // Optional: initialization code
  }

  // Step 3: Provide a public static method to get the instance
  public static Singleton getInstance() {
    if (instance == null) {
      instance = new Singleton(); // Create the instance if it doesn't exist
    }
    return instance; // Return the single instance
  }

  // Example method
```

```
    public void showMessage() {
        System.out.println("Hello from Singleton!");
    }
}
```

- If there is a variable marked as final that means that once it is initialized it can never change and there is naming conventions that the constant variables names are usually are wrote in CAPITAL the final variable It is must to initialize them with a value when you define them cause if you do not give them any value so you will get a compiler error,
- FINAL you can use it with anything so If you make a final variable so that means you can not change its value , if you make a final method so that means you can not override the method , if you make a final class so that means you can not extend it that's mean this class will never be a parent for any other classes and making the class final here its goal Is for security

- **A non-static method in a class can always call a static method in the class or access a Static variable of the class.**
- **In the chapter you will have some static methods inside the math class that you can use them**
- **What about if we want to treat the primitive types as objects to use them like define an arraylist of int variables but the array list and other collections they are working with only the objects so you can not define an array list of type int that it is not legal so here come another rule WHEN YOU NEED TO TREAT A PRIMITIVE LIKE AN OBJECT WRAP IT and the wrap here means there is a class for each primitive type this classes are built in and you can use them easily so now you can define an array list of INTEGER the int primitive class wrapper**
- The wrapper object and the primitive variable both can make anything without any wrong but there is more benefits from using the wrapper class like we can use the static methods inside it like PARSEINT which converts the strings values into the integer values and you can make the opposite thing like converting the numbers into string and that by using INTEGER.TOSTRING() function