1) **Static:**
    1) Used for memory management
    2) Used with:
        1) Variables:
            1) Can be used to refer common properties of all objects
                1) Eg: company name for employees is going to be same, so we can declare it as static so that memory is assigned only once
            2) Gets memory only once in the class area-> it gets at the time of class loading
            3) They are shared among all the instances of the class
                1) Because we can directly access them using class-name we don't have to be dependent on the objects of the class that are created
            4) Imp Points:
                1) Belong to a class
                2) Can be accessed directly using class name and don't need any object reference
                3) Can be declared only at class level
                4) Static variables can be accessed without object initialisation
        2) Methods:
            1) They also belong to class instead of objects
            2) They can be called without creating the object of the class in which they reside
            3) They are resolved at compile-time:
                1) Method-overriding -> Runtime Polymorphism because of this we cannot @Override static methods
            4) Abstract methods cannot be static
            5) Static methods cannot use **this** or **super** keyword
        3) Blocks:
            1) Inside a static block I cannot declare variable as static it can be either normal or final
            2) Can be used to initialise static variables
            3) A class can have multiple static blocks, which will execute in the same sequence in which they have been written
        4) Classes(Inner classes):
            1) Refer code

2) **Final:**
   1) Can be used with:
      1) Variables: cannot change the value once assigned
      2) Methods: we cannot override
      3) Classes: we cannot inherit

3) **Strings:**
   1) String is a non-primitive data-type in Java
   2) String is immutable, so it is easy to share across different functions
   3) Whenever String manipulation is done like concatenation, substring etc., it generates a new string and discards the string for garbage collection
   4) We cannot inherit String class -> because String class is final
   5) Represented in UTF-16 format
   6) Instantiate String by:
      1) String s1 = “abc”;
      2) String s2 = new String(“abc”);
   7) String class overrides equals(), toString() and hashCode() methods
   8) Declared in Object class
   9) All the String values are stored in String Pool
   10) Since String is immutable, so whenever we perform any manipulation, the older String is discarded, it leads to a lot of discarded string in the heap memory. To solve this issue:
       1) **StringBuilder and StringBuffer classes:**

	String	**StringBuffer**	**StringBuilder**
Storage	String Pool	Heap	Heap
Modifiable	No	Yes	Yes
Thread Safe	Yes	Yes	No
Synchronised	Yes	Yes	No
Performance	Fast	Slow	Fast