* **Strings Class( java.lang package):-**
* We can create strings Object in two ways:-

1) **By String Literal –>** String s = “Welcome”;

It is created in the string pool and JVM maintains this to store all the strings in memory if there is already created same string so it will not create another, it simply referenced to another variable not first once.

2) **By new keyword -->** String s = new String(“Welcome”);

Because of using a new keyword, this string Object was created in heap memory, even though the “Welcome” string is already present in the string pool area.

* Strings are Immutable in java.

**Immutable -->** Once an object is created, its state can’t be changed after creation just like a String is a class and when we create a string object we can not change or update it.

If we want to make our String object **Mutable,** so we have to use **StringBuffer and StringBuilder** classes.

* **StringBuffer →** It is synchronized, two threads can’t call simultaneously. It is less efficient for use.
* **StringBuilder →** It is not synchronized but is more efficient for use.
* **String Compare in java →** 3 type

1. equals() -> It compare content(values) of strings.
2. == operator -> it compare reference not value.
3. compareTo() -> It compare values laxographically

* **Arrays:-**
* In java, **the array** is a special type of **object** whose class is non-existence, whenever JVM encounters [ ] it will create an array object in memory.
* At the time of creating an array object, all the variables inside the array are initialized with their default value.
* **Scanner Class(java.util package):-**
* The nextLine() method terminated when it encounters “\n”.
* **Method Overriding:-**
* We can not override the static method.
* Method must have the same name and parameter as parent class as in child class.
* **Inheritance in java(IS-A Relationship):-**

“Taking all the parent class properties to its child class.”

* Advantage (Use) of inheritance →

1. Code reusability

(2) To get Runtime polymorphism(Method overriding)

* Types of Inheritance -

1. Single - One class extends its parent class
2. Multilevel - One class extends its parent class and this parent class also extends its parent class.
3. Hierarchical - When to class extends one parent class.
4. Multiple - One class extends two parent classes **(Only possible via Interface and it is not supported at the class level).**
5. Hybride - Diamond shape inheritance **(Only possible via Interface ).**

* **Super Keyword:-**
* Super keyword use for calling instance variable and method of an immediate parent class.
* super() is used for calling the immediate parent class constructor.
* **Upcasting:-**

“If the reference variable of parent class refers to its child class object.”

* **Downcasting:-**

“If the reference variable of child class object refers to parent class object”.

* **instanceof Operator:-**

“This is used to test whether the object is an instance of its class, child class, or of an Interface.

* **Final:-**
* **Final variable -** We can change the value of this variable but, it must be initialized before we use it.
* **Final method -** If we make any final method we can not override those methods inside the child class.
* **Final Class-** If we make any final class we can not extend it. The final class does not have any child classes.
* **Access Modifier in Java:-**

“It Specify the scope of varible,method, class and constructor.”

* **Private (Class member Only)**  → It’s access level is only within the class, for accessing from outside of class we have some corresponding methods for it.(getter Setter).
* **Protected** **(Class member Only)** → It’s acess level is with in the pacakge only, from accessing outside from the package we have created child class of it’s class (inheritence).
* **Public** → It’s access level is every where class, methods, varibles and constructor also can be public.
* **Deafult** → It’s access level is also with in the package only, we can not access from outside the pacakge. AKA packege protected modifier.
* **Abstraction in java:-**

**“** It is a process to hiding the implementation details and showing only functionality to user. **”**

* Ways to achive Abstraction in Java →
* Abstract Class (0 to 100 % abstraction.)
* Interface ( 100 % abstraction)

1. **Abstract Class** → The class that is decleard with abstract keyword is known as abstract class.

* It can have abstract and non abstract methods.
* We can not create it’s Object (initialized).
* It can have constructor and static nonStatic and main method.
* It can have final methods.
* **Abstract Method -** These methods are only have method decleration not method body. If any class has a abstract method than this class must be abstract class but it’s vice-versa is not true.

**Ex →** abstract returnType funName(parameter);