**1. Inheritance:**

This is one of the OOPS concepts. Inheritance is acquiring the properties of another class. It gives the reusable property for the entities. The keyword “extends” is used to inherit the properties. It follows “IS-A” relationship.

2. **protected:** The variables or methods with this access modifiers can be used by the sun class only. The child class can access the variable even from another package.

**Default:** If we do not specify any access modifier then is “default” is used. Using default you cannot access it outside the package.

**3. static:** If a variable is declared static then it has a single copy of memory. It is used globally and any changes made to the variable anywhere reflects it since it shared the same memory. So we need to be careful when we declare the variable as static.

**Instance:** These are the variables declared inside the class. To refer them we can also use “this” keyword. These variables can be used anywhere inside the class.

4. **Array:**

* It is a collection of similar datatypes.
* The array size must be declared during the compile time itself.
* The adding and deleting operation in arrays are complex. If we want to add or delete the elements we need to shift the adjacent elements forward or backward.
* We may get unchecked exceptions if the array size is out of bound.
* Retrieving the elements we need to traverse through the entire array.

- e.g.: int[] str = new int[10];

**ArrayList:**

* It is a class in Collection framework that implements the List interface.
* The elements in the array list are stored sequentially. (i.e internally the compiler searches for the memory area that is in sequence. If an additional element is to be added and there is no space in sequence it shift the entire array list to the sequenced memory address).
* It is not suitable for adding and deleting elements.
* This nature of array list is suitable for searching and retrieving the elements.
* E.g.: ArrayList<Integer> al=new ArrayList<>():

**LinkedList:**

* It is also a class in Collection framework that implements the List interface.
* The elements are not stored in sequence. Each element in linked list has the values and the pointer to the next memory location.
* Adding and deleting the elements becomes easy.(If you want to delete the element it deletes the element and points to the next element)
* Searching is an expensive task. (Because the address pointing may be in another memory area.)
* E.g.: LinkedList<Integer> ll=new LinkedList<>():