* Two types of methods:

1. Instance Methods

* Instance methods are used to access the instance variables but can also access class variables.

1. Class Methods

* Class methods can access class variables but cannot access the instance variables unless and until they use an object for that purpose.
* Methods created using ‘static’ keyword is called class methods.
* Static variables are known as Class variables.
* About Parameters:
* Formal Parameters: Parameters that are used in Method definition that is called as formal parameters.
* Actual Parameters: Parameters that are used in Method calling that is called as Actual parameters.
* In java all the variables are passed by reference.
* Constructors:
* Implicit return type of constructor is the class type itself.
* When no explicit constructor is provided at that time java complier creates a default constructor and <init> method for the default constructor.
* <init> is a special method, meant for the JVM and not for the programmer.
* Garbage Collector:
* Java Runtime Environment deletes objects when it determines that they are no longer in use.
* An object is eligible for garbage collection when no references exist on that object.
* Two basic approaches used by garbage collection

1. Reference counting
2. Tracing [ Mark and sweep ]

* Garbage collector can be run synchronously or asynchronously. Synchronously when the system runs out of the memory. And Asynchronously when system is idle.
* We can call garbage collector run time by calling System.gc( ) or Rutime.gc( ).
* Advantage:
* Programmers no need to worry about deallocate memory.
* Disadvantage:
* Need more processing time because need to track which objects are in use and which are not in use.
* Static keyword:
* Sometimes we need multiple objects, shared variables and shared methods. Static keyword will be used to achieve this.
* Java doesn’t allow global variables. The closest thing we can get to global variable in java is to make instance variable in the class static.
* Static Block:
* Static block is called even before JVM executes main method.
* It used to initialize static variables.
* See the example [StaticBlockDemo.java](Java%20Fundamentals/StaticBlockDemo.java)
* this keyword:
* The exact purpose of this is to remove ambiguity between local and instance variables.
* If same names are provided for both the parameters then the instance variables will be hidden by the local variables.
* Constructor chaining:
* It means a constructor can be called from another constructor.
* See the example ConstructorChaining.java
* Nested Classes:

1. Non static inner class

* Non static inner class is a member of the outer class.
* Instance of non-static inner class is created within the instance of outer class.
* All the variables and function of outer class is accessible from inner class.
* Non static inner classes have instance scope.
* See Example InnerClassTest.java

1. Static Nested Class:

* A static nested class is a static member of a class just like normal static member.
* They have access to all the static methods of the outer class.
* Static nested class can’t directly refer to instance variable and instance method of outer class.
* Static nested class have class scope.
* To create a static nested class object, outer class’s object doesn’t needed.
* Why do we need nested classes?
* Nested classes are a security mechanism in java. Because we know class can’t be private. But we class is the member of another class that it can be declared as private.
* Inheritance:
* The test for inheritance is that there exists an ‘is-a’ kind of relationship. Example: manager is a kind of employee.
* The test for aggregation is that there exists an ‘is-a-part-of’ relationship among classes.
* Types of inheritance:

1. Single Inheritance
2. Multilevel Inheritance
3. Multiple inheritance [ Not supported in java ]
4. Hierarchical inheritance
5. Hybrid Inheritance

* Referencing Subclass object with subclass vs superclass reference.

<https://www.geeksforgeeks.org/referencing-subclass-objects-subclass-vs-superclass-reference/>

* Super keyword:
* Super keyword refers to the parent class.
* It is used for following purpose:

1. For calling the methods of the superclass.
2. For accessing the member variable of the superclass.
3. For invoking the constructors of the superclass.

* Final keyword:
* It is used for following purpose:

1. To declare constant.
2. To disallow method overriding
3. To disallow inheritance

* String class:
* Strings are immutable objects in java.
* There are two ways in java to declare a string.

1. String x = “Hello world”;
2. String y = new String(“Hello world”);

* In both the way String class object is created. In 1st example String object is created implicitly and memory is allocated from memory pool which is created from string literals.
* In 2nd example memory is allocated from out of the memory pool.
* For example 1, Before creating objects for String literal. JVM checks the memory pool for existence of String literal in the pool and if JVM found, a reference to the existing string object is passed otherwise new string object is created.
* String Manipulation:
* String s1=”Hello”;

String s2=s1+”World”;

* Strings are immutable so how s2 gets executed? So, s2 is converted into the following statement:

String s2= new StringBuffer( ).append(s1).append(“World”).toString( );

* New StringBuffer object is created which is mutable set of characters.
* Java Collections:

Collection interface

Iterable interface

List Interface

1. ArrayList:

* ArrayList implement list interface.
* ArrayList can’t be used for primitive types. We need wrapper classes for that.
* ArrayList can be seen as vector in C++.
* See the example ArrayListExample.java