Basic Java II

- Break and continue can be used with labels as well
- Java Enhanced for loop: Java Enhanced for loop provides a simpler way to iterate through the elements of a collection or array. It is used when we need to sequentially iterate through elements without knowing the index of the currently processing element.
  Syntax:

```
for(T element: collection obj / array) {
   // statements
}
```

- *Bit Manipulation:* To count 1's in binary format for an integer, use:
  - Instead of converting the number to a binary string, you can use bitwise operations to count the 1's.
  - For example, repeatedly check the least significant bit using N & 1, then right-shift N by 1.
  - Alternatively, use Brian Kernighan's algorithm
- *Memory Allocation:* JVM does it in two mays -
  - Stack:
    - used for static memory allocation and thread execution;
    - Methods, local variables, and reference variables are all stored in the Stack portion of memory.
    - When a method completes its execution, the resulting stack frame is cleared, the flow returns to the calling method, and space for the next method becomes available.
    - Since each thread runs in its own stack, this memory is thread-safe.
    - As compared to heap memory, access to this memory is fast.
  - Heap:
    - In heap memory, new objects are often formed, and references to these objects are stored in stack memory.
    - Garbage Collection, a discrete function, keeps flushing the memory used by previous objects that have no reference.
    - This memory model is divided into generations, which are as follows:
      - Young Generation – All new objects are allocated to and aged in this memory. When this is complete, a minor garbage collection occurs which determines the transfers.
      - Old or Tenured Generation – This is the memory where long-lasting items are kept. When objects are stored in the Young Generation, an age threshold is set, and when that threshold is met, the object is transferred to the Old Generation.
      - Permanent Generation – This is a collection of JVM metadata for runtime classes and application methods.
    - Access to this memory is slower than access to stack memory.

        - Unlike the stack, this memory is not immediately deallocated. Garbage Collector is used to free up unused objects in order to maintain memory efficiency.
        - A heap, unlike a stack, is not thread-safe and must be protected by synchronizing the code properly.
    - Only a reference is created in Java when we only declare a variable of a class type (memory is not allocated for the object). We must use new() to assign memory to an object. As a result, the heap memory is always assigned to the object.
    - Ways to make an object eligible for the garbage collector:
        - Nullifying the reference variable
        - By assigning a reference variable to another.
        - By Aonymous object (not assigning the new object to a variable)
    - There are two ways for requesting a JVM to run a garbage collector.
        - Using System.gc() method.
        - Using Runtime.getRuntime().gc() method.
    - When an object is no longer referenced, it may be reclaimed by the garbage collector. If an object declares a finalizer, the finalizer is executed before the object is reclaimed to give the object a last chance to clean up resources that would not otherwise be released. When a class is no longer needed, it may be unloaded.
    - Garbage Collection cannot be controlled by a program. A program can only request to run the garbage collector.
- *Exception Handling:*
    - In Java, the throw keyword is used to explicitly throw an exception. Using the throw keyword, we can throw either checked or unchecked exceptions. The throw is mostly used to throw user-defined exceptions.
    - Java allows us to create our own exception, which is essentially a derived class of Exception.
    - Throwable is the parent class of all the exception classes.
    - Exceptions in java arise at run time.
    - Following is the hierarchy of the Java Exception Classes:
        - In Java, the throwable class is the parent class of all exception classes.
        - There are two child classes in the throwable class, i.e., Error and Exception.
        - The Exception class represents exceptions that the programmer can handle using the try-catch block.
        - RuntimeException is a class that is a subclass of the Exception class and comprises the majority of the unchecked exceptions. Unchecked Exceptions include ArithmeticException, NullPointerException, NumberFormatException, etc.
    - In Java, there are two types of exceptions:
        - Checked Exception: If the compiler does not handle the exception, we will get a compile-time error in the case of checked exceptions. These

exceptions can be handled using try-catch blocks or by declaring the exception with the throws keyword.

- Unchecked Exception: The compiler does not check Unchecked Exceptions at compile time. However, they are checked at runtime. Unchecked exceptions include ArithmeticException, NullPointerException, ArrayIndexOutOfBoundException, and others.