**Java Industry Assignment**

Module 1: Core Java

1.Introduction to Java

* History of Java
* Java is a programming and platform-based language.
* Java is a high level, robust, object-oriented programming language.
* Java was developed by sun Microsystem (which is now subsider by oracle) in the year 1995.
* James Gosling is known as a father of java.
* Before Java its name was oak.
* Since oak was already registered company, so James’s gosling and his team changed the name from oak to java.
* Features of Java
* Simple:
  + Java is very easy to learn and its syntax is simple, clean any easy to understand.
  + Java has removed many complicated and rarely used feature for e.g. explicit, pointers, operator overloading etc.
* Object-oriented:
  + Java is an object-oriented programming language.
  + Everything in java is an object.
  + Object-oriented means we organize our software as a combination of different type of objects that incorporate both data and behaviours.
* Platform-independent:
  + Java is a platform independent because it is different from other languages like c ,c++ etc.
  + Java code can be executed on multiple platforms for e.g. Windows, Linux, mac/os, etc.
  + Java code is compiled by compiler and converted into bytecode. This bytecode is a platform-independent because it can be run on multiple platforms. i.e. write once and run anywhere.
* Secured:
* Java is best known for its security.
* Java is secured because no explicit pointer, Java programs run inside a virtual machine sandbox.
* Robust:
  + It uses strong memory management.
  + Java provides automatic garbage collection which runs on the java virtual machine to get rid of objects which are not being used by Java Application Anymore.
* Portable:
* Java is portable because it facilities you to carry the java bytecode to any platform.
* It doesn’t require any implementation.
* High-performance:
  + Java is faster than other traditional interpreted programming language because java bytecode is “close” to native code.
* Understanding JVM, JRE, and JDK
  + JDK:
    - JDK stands for Java Development Kit, it internally contains JRE & JVM.
    - JDK provides all the tools to work with java language.
* JRE:
* JRE stands for Java Runtime Environment, it provides on environment to execute the java program.
* It internally contains JVM which is responsible to execute Java Program.
* JVM:
  + JVM stands for Java Virtual Machine, it is the software in the form of interpreter written in ‘c’ language through which we can execute java program.
  + It is specification that provides runtime environment in which java bytecode can be executed.
* Java Program Structure (Packages, Classes, Methods)
  + Packages:
    - A package average number of classes, methods, interfaces and sub-packages of same type into a particular group.
    - It is used for reusability, security and fast searching.
* Classes:
  + Class is a collection of objects and it doesn’t take any space in memory, class is also called blue print of entity.
  + Group of data members and structure.
  + Two types of classes:

1)predefined class (Scanner, System, String, Console)

2)User defined class (Demo, Test etc.)

* Methods:
  + Method is a group block of code which take input from the user processed it & give output.
  + Method runs only when it called.

2. Data Types, Variables, and Operators

* Primitive Data Types in Java (int, float, char, etc.)
  + Int:
    - The int data type in java is a primitive data type that represents a 32-bit signed two’s complement integer.
    - It has a range of values from -2,147,483,648 to 2,147,483,647.
    - The int data type is one of the most commonly used data types in Java and is typically used to store whole numbers without decimal points. Its default value is 0.
* Float:
  + The float data type in java is a primitive data type that represents single-precision 32-bit IEE 754 floating-point numbers.
  + It can represent a wide range of decimal values, but it is not suitable for precise value such as currency.
  + The float data type is useful for applications where a higher range of values needed, and precision is not critical.
* Boolean:
  + In Java, the Boolean data type represents a single bit of information with two possible states: true or false.
  + It is used to store the result of logical expressions or conditions.
* Byte:
  + The Byte data type in java is primitive data type that represents an 8-bit signed two’s complement integer.
  + It has a range of values from -128 to 127.
  + Its default value is 0.
  + The byte data type is commonly used when working with raw binary data or when memory conversation is a concern, as it occupies less memory than larger integer types like int or long.
* Short
  + The short data type in java is a primitive data type that represent a16-bit signed two’s complement integer.
  + Short is used when memory conversation is concern, but nore precision than byte is required.
  + Its default value is 0.
* Long:
  + The Long data type is primitive data type that represents a 64-bit signed two’s complement integer.
  + It has a wider range of values than int, ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
  + Its default value is 0.0f.
  + The long data type is used when int is not large enough to hold the desired value, or when a larger range of integer value is needed.
* Double:
  + The Double data type in java is represents double-precision 64-bit IEEE 754 floating-point numbers.
  + Its default value is 0.0.
  + It provides wider range of values and greater precision compared to the float data type, making it suitable for applications where accurate representation of decimal value is required.
* Char:
  + The char data type in java is a primitive data type that represents a single 16-bit Unicode character.
  + It can store any character from the Unicode character set, that allows Java to support internationalization and representation of characters from various languages and writing systems.
* Variable Declaration and Initialization
* Variables is the name of memory location.
* In other word we can say it is user defined name which is given by user.
* Variables can store any types of values.
* Three types of variables:
  1. **Local Variable:** A Variable which is declared inside the body of the method or method parameter called local variable.
  2. **Instance Variable:** A Variable which is declared inside the class but outside of all the methods called instance variable.
  3. **Static Variable:** A Variable which is declared with the help of static keyword called static variable.
* Memory allocation for static variables happens only once when the class is loaded in the memory.
* Operators: Arithmetic, Relational, Logical, Assignment, Unary, and Bitwise
* Operator is a symbol that is used to perform operations according to user’s requirement.
* **Arithmetic Operator:** The ability to do basic mathematical calculation on numeric data types makes arithmetic operators one of Java’s most often used operators.
  + - 1. Additional Operator (+)
      2. Subtraction Operator (-)
      3. Multiplication Operator (\*)
      4. Division Operator (/)
      5. Modules Operator (%)
      6. Increment Operator (++)
      7. Decrement Operator (--)
      8. Compounds Assignment: Compound Assignment operators enable you to execute a computation and assign the result to a variable in a single statement by combining an arithmetic operator with an assignment operator.
* **Relational Operator:** Relational Operators are bunch of binary operators used to check for Relations between two operands, including equality, greater than, less than etc. They return a Boolean result after the comparison and are extensively used in looping statement as well as conditions if-else statements and so on.
  1. ‘Equals to’ Operator (==)
  2. ‘Not equal to’ Operator (! =)
  3. ‘Greater than’ Operator (>)
  4. ‘Less than’ Operator (<)
  5. ‘Greater than or equal to’ (>=)
  6. ‘Less than or equal to’ (<=)
* **Logical Operator:** Logical Operators are used to perform logical “AND”, “OR” and “NOT” operations, i.e. the function similar to AND gate and OR gate in digital electronics. They are used to combine two or more conditions/constraints or to complement the evaluation of the original condition under particular consideration. One thing to keep in mind is, while using AND operator, the second condition is not evaluated if the first one is false. Whereas while using OR operator, the second condition is not evaluated if the first one is true, i.e. the AND and OR operators have a short-circuiting effect. Used extensively to test for several conditions for making a decision.
  1. **AND Operator (&&) –** if (a && b ) [if true execute else don’t]
  2. **OR Operator (||) –** if (a || b) [if one of them is true to execute else don’t]
  3. **NOT Operator (!) –!** (a<b) [returns false if a is smaller than b]
* **Ternary Operators**: Ternary operator is the only conditional operator that takes three operands.  It’s a one-liner replacement for the if-then-else statement and is used a lot in Java programming. We can use the ternary operator in place of if-else conditions or even switch conditions using nested ternary operators. Although it follows the same algorithm as of if-else statement, the conditional operator takes less space and helps to write the if-else statements in the shortest way possible.

**Syntax:** (? :)

* **Bitwise Operators:** Bitwise operators are used to performing the manipulation of individual bits of a number which can be useful for optimizing performance in certain cases. They can be used with any integral type (char, short, int, etc.). They are used when performing update and query operations of the Binary indexed trees.
  1. Bitwise OR (|)
  2. Bitwise AND (&)
  3. Bitwise XOR (^)
  4. Bitwise Complement (~)
* **Unary Operators:** In Java Unary Operators requires only one operand. Unary Operators are used to perform various operations.
  1. Incrementing/decrementing a value by one.
  2. Negating an expression.
  3. Inverting the value of a Boolean
* Type Conversion and Typecasting
* **Type Conversion:** In type conversion, a data type is automatically converted into another data type by a compiler at the compiler time. In type conversion, the destination data type cannot be smaller than the source data type, that’s why it is also called widening conversion. One more important thing in that it can only be applied to compatible data types.
* **Type Casting:** In type casting, converting one datatype to another datatype is called typecasting. two types of type casting:
  1. Implicit: It is automatically performed by the compiler.
  2. Explicit: By default, the compiler doesn’t allow the explicit typecasting.

3.Control Flow Statement

* **In-else Statement:** It is used to execute two statements either if statement or else statement for a single condition.

**Syntax:** if(condition) {

} else {

}

* **Switch case statement:** Switch is a multiple-choice decision-making selection statement. It is used when we want to select only one case out of multiple cases.

**Syntax:** Switch () {

Case 1: statement 1;

break;

Case 2: statement 2;

break;

}

* Loops (For, While, Do-While)
* **For-Loop:** for loop is the most commonly used loop.
  + It is used to when we want to perform initialization, condition and increment/decrement operation in single line.

**Syntax:** for (initialization, condition, increment/decrement)

{

}

* **While Loop:** While loop is a pre-test loop, it is used when we don’t know the number of iterations in advance.
  + It is also known as entry control loop.

**Syntax:** While(condition)

{

}

* **Do-While Loop:** Do while loop is a post-test loop, it is used when we want to execute loop body at least once even condition is false.
  + It is also known as exit control loop.

**Syntax:** do {

}

While(condition)

* Jump Statements
  + **Break Statement:** The break statement to used to break the current flow of the program and transfer the control to next statement outside a loop or switch statement. However, it breaks only the inner-loop in the case of nested loop.
* The break statement cannot be used independently in java program, i.e., it can only be written inside the loop or switch statement.
* **Continue Statement:** The Continue statement doesn’t break the loop, whereas, it skips the specific part of the loop and jumps to the next iteration of the loop immediately.

4. Classes and Objects

* Defining a Class and Object in Java
  + **Class:** Class is a collection of objects and it doesn’t take any space an memory, class is also called blueprint of entity.
* Group of data members and Structure.
* Two types of class:

1. Predefined Class (Scanner, System, String, Console)
2. User defined Class (Demo, Test, etc.)

* **Object:** Object is a instance of a class that executes the class once the object t is created, it takes up space like other variable in memory.
* **Syntax:**
* Class-name Object reference name = new Constructor name ()
* Constructors and Overloading
  + **Constructor:** Constructor is a special type of method whose name is same as class name.
* Main purpose of constructor is initialization the object.
* Every java class has a constructor.
* A constructor is automatically called at the time of object creation.
* A constructor never contains any return-type including void.
* Types of constructors:

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor

* **Overloading:** Whenever a class contain more than one method with same name and different types of parameters is called Method Overriding.
* Object Creation, Accessing Members of the Class
  + **Object Creation:** The object is building of a block of an oops language. In java, we cannot execute any program without creating an object.
* Java provides five ways to create an object.
  + Using **new** keyword
  + Using **clone ()** method
  + Using **newInstance ()** method of the **Class** class
  + Using **newInstance ()** method of the **Constructor** class
  + Using **Deserialization**
* Accessing Members of class
* Public
* Private
* Protected
* default
* **this Keyword:** this keyword refers to the current object inside a methos or constructor.
* Whenever the name of instance and local variable both are same then our runtime environment JVM gets confused that which one is instance variable, to avoid this problem we should use this keyword.
* It is also used when we want to call the default constructor of its own class
* It also called parameterized constructor of its own class.

5. Methods in Java

* Defining Methods:
  + The Method in java or methods of java is a collection of statements that perform some specific tasks and return the result to the caller.
  + A java method can perform some specific tasks without returning anything.
  + Java Method allow us to reuse the code without retyping the code.
* Method Parameters and Return Types
  + **Method Parameters:** Comma – separated list of the input parameter is defined, preceded by their data type, within the enclosed parentheses. If there are no parameters, you have must use empty parentheses () . It is optional in syntax.
  + **Return Types:** The data type of the value returned by the method or void if does not return a value. It is mandatory in syntax.
* Method Overloading
  + Method overloading in java is also known as Compile-time polymorphism, static polymorphism, or early binding. In method overloading compared to the parent argument, the child argument will get the highest priority.
* Static Methods and Variables
  + **Static method:** Access the static data using class name. Declared inside class with static keyword.
  + **Static variable:** A variable which is declared with the help of static keyword called static variable.
* Memory allocation for static variables happens only once when the class is loaded in memory.

6. Object-Oriented Programming (OOPs) Concepts

* Basics of OOP: Encapsulation, Inheritance, Polymorphism, Abstraction
  + **Encapsulation:** Encapsulation is a mechanism through which we can wrapping the data members and member methods of class in a single unit called encapsulation.
* Declare the class variables as a private.
* Declare the class methods as a public.
* **Inheritance:** To access property of one class to another class is called inheritance.
* In java extends keywords is used to perform inheritance.
* It provides code reusability.
* We can’t access private members of class through inheritance.
* A sub class contains all the features of super class so, we should create all the object of sub class.
* Method overriding only possible through inheritance.
* Multiple and hybrid inheritance is not supported in java. It occurs Ambiguity.
* **Polymorphism:** Same function name but having different functionalities.
* Two types of polymorphism:

1. Compile time (Overriding)
2. Runtime time (Overloading)

* **Abstraction:** Hiding Internal detail and showing essential information to user.
* To provide abstraction abstract keyword is used class and function.
* In abstract class abstract, default and static functions are allowed.
* In function is abstract then class also should be abstract.
* Abstract class provides near to 100% abstraction.
* By using child class object, we can access static property of abstract class.
* We can not instantiated object of class.
* Abstract class should be inherited purify abstract property of abstract class.
* Method Overriding and Dynamic Method Dispatch
* **Method Overriding:** Whenever we writing method in super class and sub classes in such a way the method name and parameter must be same called method overriding.
* **Dynamic Method Dispatch:** Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time.

7. Constructors and Destructors

* Constructor Types (Default, Parameterized)
  + **Default Constructor:** A constructor which does not have any parameter is called default constructor.
  + **Parameterized Constructor:** A constructor through which we can pass one or more parameters called parameterized constructor.
* Copy Constructor (Emulated in Java)
* Whenever we pass object reference to the constructor then it is called copy constructor.
* Constructor Overloading
* In java, we can overload constructor like methods. The constructor overloading can be defined as the concept of having more than one constructor with different parameter so that every constructor can perform a different task.
* Object Life Cycle and Garbage Collection
  + **Object Life Cycle:** There are seven steps comes in the life cycle of object in java.
    - Step 1: Creation of .class file on disk
    - Steo 2: Loading .class file into memory
    - Step 3: Looking for initialized static members of class
    - Step 4: Ways to initialize class in java
    - Step 5: Allocation of memory for object and reference variable
    - Step 6: Calling of the constructor of class
    - Step 7: Removing of object and reference variable from memory
  + **Garbage collection:** Garbage collection is a process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

8. Arrays and Strings

* One-Dimensional and Multidimensional Arrays
  + **One-Dimensional:** A one-dimensional array in java is a linear collection of elements of the same data type. It is declared and instantiated using the following syntax: arrayRefvar = new datatype[size];
  + **Multidimensional Arrays:** A multidimensional array in java is an array of arrays where each element can be an array itself. It is useful for storing data in row and column format.
* String Handling in Java: String Class, StringBuffer, StringBuilder
  + **String Class:** String is a sequence of characters.
  + In java string in immutable, if you want to change a value for string. Java provides a classes StringBuffer and StringBuilder.
  + String is an object that represent a sequence of the characters.
* **StringBuffer:** String buffer class is used to create mutable (modifiable) String object. The StringBuffer class in java is the same as String class except it is mutable i.e. it can be change.
* **StringBuilder:** StringBuilder class is used to create mutable (modifiable) String. The Java StringBuilder class is same as StringBuffer class except that is non-synchronized.
* Array of Objects
* An array of object is used to store multiple instances of a class within a single array. This allows us to easily manage a collection of objects when working with large datasets or collections.
* String Methods (length, charAt, substring, etc.)
  + **length:** It returns string length
  + **charAt**: It returns character value for the particular index
  + **substring:** It return substring for given begin index.

9. Inheritance and Polymorphism

* Inheritance Types and Benefits
  + **Inheritance Types:** InJava following five types of inheritance:
    - Single Inheritance
    - Multi-level Inheritance
    - Hierarchical Inheritance
    - Multiple inheritance
    - Hybrid Inheritance
  + **Benefits:** 
    - Code Reusability
    - Facilitates Polymorphism
    - Class Hierarchy
    - Code Maintainability
    - Code Modularity
    - Abstraction
    - Code Flexibility
* Method Overriding
  + Whenever we writing method in super class and sub classes in such a way the method name and parameter must be same called method overriding.
* Dynamic Binding (Run-Time Polymorphism)
  + A polymorphism which exists at the time of execution of program is called runtime polymorphism.
  + Runtime polymorphism achieve method overriding.
* Super Keyword and Method Hiding
  + **Super Keyword:** Super keyword refers to the objects of super class, it is used when we want to call the super class variable, method and constructor through sub class object.
  + **Method Hiding:** Method hiding can be defined as, "if a subclass defines a static method with the same signature as a static method in the super class, in such a case, the method in the subclass hides the one in the superclass." The mechanism is known as method hiding. It happens because static methods are resolved at compile time.

10. Interfaces and Abstract Classes

* Abstract Classes and Methods
  + To provide abstraction abstract keyword is used class and function.
  + In abstract class, abstract, default and static functions are allowed.
  + In function is abstract then class also should be abstract.
  + Abstract class provides near to 100% abstraction.
  + Abstract class should be inherited purify abstract property of abstract class.
* Interfaces: Multiple Inheritance in Java
  + Multiple inheritance is a feature of an object-oriented concept, where a class inherit properties of more than one parent class.
  + The problem occurs when there exist methods with the same signature in both the superclass and subclass. On calling method compiler cannot determine which class method to be called even on calling which class method gets the priority.
* Implementing Multiple Interfaces
  + To implement multiple inheritance in java, the following syntax is used:

**Class** MyClass **implements** interface1, interface2, interface3 {

//class body

}

11. Packages and Access Modifiers

* Java Packages: Built-in and User-Defined Packages
  + Built-in packages:
    - Java.lang
    - Java.util
    - Java.io
    - Java.applet
    - Java.awt
  + User-Defined packages:
    - Package p1
    - Package add
    - Package myPackage
* Access Modifiers: Private, Default, Protected, Public

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifier | Within class | Within Package | Outside package by subclass | Outside package |
| Private | ✓ | x | x | x |
| Default | ✓ | ✓ | x | x |
| Protected | ✓ | ✓ | ✓ | x |
| Public | ✓ | ✓ | ✓ | ✓ |

* Importing Packages and Classpath:
  + Importing packages:
* To use classes or interfaces from other packages.
* To improve code readability and reduce the need for fully qualified class names.
* **Classpath:** Classpath is a parameter that specifies the location of classes and packages required by the java compiler and JVM.

12. Exception Handling

* Types of Exceptions: Checked and Unchecked
  + **Checked:** The checked exception which are checked by compiler for smooth execution of runtime.
* Checked exception are commonly occurred exception. So, the compiler takes very much care about this exception.
* **Unchecked:** The exception which are most checked by compiler and it directly taken by JVM.
* Unchecked exception are rarely occurred exception so the compiler doesn’t take very much care about these exceptions.
* try, catch, finally, throw, throws
  + **try block:** Whenever we write a statement and if the statement is error suspecting statement or risky code then put that code inside the try block.
* try is a block that contains only risky code.
  + **catch block:** The main purpose of catch block is to handle the exception which are throws by try block.
* catch block is used to handle the exception.
  + **finally block:** finally, is a block.
* It is always executing weather the exceptions is handled by user or not.
  + **throw keyword:** throw keyword is used to throw an exception explicitly.
  + **Throws keyword:** throws keyword is used when we doesn’t want to handle the exception and try to sent the exception to the JVM.
* Custom Exception Classes:
  + Java exceptions cover almost all the general types of exceptions that may occur in the programming. However, we sometimes need to create custom exceptions.
  + Following is a few of the reasons to use custom exceptions:
    - To catch and provide specific treatment to a subset of existing Java exceptions.
    - Business logic exceptions: These are the exceptions related to business logic and workflow. It is useful for the application users or the developers to understand the exact problem.

13. Multithreading

* Introduction to Thread
  + Smallest unit of process is called thread.
  + At a time only single thread can be execute.
  + There are three priorities: min, max, nom
* Creating Threads by Extending Thread Class or Implementing Runnable Interface
  + Threads by Extending Thread class:
    - The simplest way to create a thread in java is by extending the Thread class and overriding its run () method.
    - Thread class provide constructor and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable Interface.
  + Implementing Runnable Interface:
    - Another approach to creating thread in java is by implementing the Runnable interface. The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface has only one method named run (). This approach is preferred when we want to separate the task from thread itself, promoting better encapsulation and flexibility.
* Thread Life Cycle:
  + As we know a thread is well known for independent execution. During the life cycle a thread can move from different states.

1. New State (Born)
2. Runnable State (Ready)
3. Running State (Execution)
4. Waiting State (Blocked)
5. Dead State (Exit)

* Synchronization and Inter-thread Communication:
  + **Synchronization:** Synchronization is a technique through which we can control multiple threads or among the no of threads only one thread will enter inside with synchronized area.
  + **Inter-thread Communication:** Inter-thread communication or co-operations is all about allowing synchronized threads to communicate with each other.

14. File Handling

* Introduction to File I/O in Java (java.io package)
  + Java brings various Streams with its I/O package that helps the user to perform all the input-output operations. These streams support all the types of objects, data-types, characters, files etc to fully execute the I/O operations.
* FileReader and FileWriter Classes
  + **FileReader:** Java FileReader class is used to read data from the file. It returns data in byte format like FileInputStream class.
* It is character-oriented class which is used for file handling in java.
* **FileWriter:** Java FilterWriter class is an abstract class which is used to write filtered character streams.
* The sub class of the FilterWriter should override some of its methods and it may provide additional methods and fields also.
* BufferedReader and BufferedWriter
  + **BufferedReader:** The BufferedReader class, part of Java's java.io package, provides a convenient way to read text from an input stream. It extends the abstract class Reader and offers additional functionality for buffering input, making it particularly suitable for scenarios where reading character data is required, such as parsing text files or processing network communication.
  + **BufferedWriter:** Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast. It inherits Writer class. The buffering characters are used for providing the efficient writing of single arrays, characters, and strings.
* Serialization and Deserialization
  + **Serialization:** Serialization is a mechanism of converting the state of an object into a byte stream.
  + **Deserialization:** Reverse process od serialization.

15. Collections Framework

* Introduction to Collections Framework
  + The collection in java is a framework that provides an architecture to store and manipulate the group of objects.
* Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
  + Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, priorityQueue, HashSet, LinkedHashSet, TreeSet).
* List, Set, Map, and Queue Interfaces
  + **List:** In list insertion order is maintain.
* In list duplication is allowed.
* List is slow.
* **Set:** In set insertion order is not maintaining.
* In set duplication is not allowed.
* Set is fast.
* **Map:** In map data store in key, value pair.
* Map is interface.
* Key should be unique and values can duplicate.
* HashMap and Hashtable are two classes which implement map interface.
* **Queue:** A queue is another kind of linear data structure that is used to store elements just like any other data structure but in a particular manner. In simple words, we can say that the queue is a type of data structure in the java programming language that stores elements of the same kind. The components in a queue are stored in a FIFO (First In, First Out) behaviour. There are two ends in the queue collection, i.e., front & rear. Queue has two ends that is front and rear.
* ArrayList, LinkedList, HashSet, TreeSet, HashMap, TreeMap
  + **ArrayList:** ArrayList increased list its size by 50% based on elements.
* ArrayList is newer version in collection
* It provides us with dynamic arrays in Java.
* **LinkedList:** Linked list is a part of the collection framework present in java.util package.
* This class is an implementation of the LinkedList data structure which is a linear data structure where the elements are not stored in contiguous locations and every element is a separate object with a data part and address part. The elements are linked using pointers and address and each element is known as node.
* **HashSet:** HashSet in java implements the Set interface of collection Framework. It is used to store the unique elements and it doesn’t maintain any specific order of elements.
* Can store the null value.
* Uses HashMap internally.
* **TreeSet:** TreeSet is one of the most important implementations of the SortedSet interface in Java that uses a Tree (red – black tree) for storage. The ordering of the elements is maintained by a set using their natural ordering whether or not an explicit comparator is provided. This must be consistent with equals if it is to correctly implement the Set interface.
* **HashMap:** n Java, HashMap is part of the Java Collections Framework and is found in the java.util package. It provides the basic implementation of the Map interface in Java. HashMap stores data in (key, value) pairs. Each key is associated with a value, and you can access the value by using the corresponding key.
* **TreeMap:** Java TreeMap class is a red-black tree-based implementation. It provides an efficient means of storing key-value pairs in sorted order.
* The java.util package contains the Java TreeMap class, which is a component of the Java Collections Framework. It extends the AbstractMap class and implements the NavigableMap interface. TreeMap is an effective red-black tree-based solution that sorts key-value pairs. TreeMap works well in situations where ordered key-value pairs are necessary since it preserves ascending order. It also only includes distinct components, guaranteeing that every key corresponds to a single value. TreeMap can have more than one null value, even if it cannot have a null key.
* Iterators and ListIterators
  + **Iterators:** In Java, an Iterator is one of the Java cursors. Java Iterator is an interface that is practiced in order to iterate over a collection of Java object components entirety one by one.
  + **ListIterators:** The listIterator () method of Java ArrayList returns a list iterator over the elements in this list starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

16. Java Input/Output (I/O)

* Streams in Java (Input Stream, Output Stream)
  + **InputStream:**InputStream is an abstract class of Byte Stream that describe stream input and it is used for reading and it could be a file, image, audio, video, webpage, etc. it doesn’t matter. Thus, InputStream read data from source one item at a time.
  + **OutputStream:**OutputStream is an abstract class of Byte Stream that describes stream output and it is used for writing data to a file, image, audio, etc. Thus, OutputStream writes data to the destination one at a time.
* Reading and Writing Data Using Streams
  + **Readers/Writers:** Readers and writers are specialized stream classes designed for handling character data. They provide a convenient way to read from and write to character-based data sources. Readers read character data from input streams, while writers write character data to output streams.
* Handling File I/O Operations
  + In Java, with the help of File Class, we can work with files. This File Class is inside the java.io package. The File class can be used by creating an object of the class and then specifying the name of the file.