**Name – Ayan Mondal**

**Emp ID – 815954**

**Core Java Notes**

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]). The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

Java is −

* **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
* **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
* **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
* **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

**Java Data Types:**

* Primitive Data Types

A primitive data type specifies the size and type of variable values, and it has no additional methods.

There are eight primitive data types in Java:

|  |  |  |
| --- | --- | --- |
| Data Type | Size | Description |
| byte | 1 byte | Stores whole numbers from -128 to 127 |
| short | 2 bytes | Stores whole numbers from -32,768 to 32,767 |
| int | 4 bytes | Stores whole numbers from -2,147,483,648 to 2,147,483,647 |
| long | 8 bytes | Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| float | 4 bytes | Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits |
| double | 8 bytes | Stores fractional numbers. Sufficient for storing 15 decimal digits |
| boolean | 1 bit | Stores true or false values |
| char | 2 bytes | Stores a single character/letter or ASCII value |

* **Non-Primitive Data Types**

Non-primitive data types are called reference types because they refer to objects.

Examples of non-primitive types are [Strings](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp), [Classes,](https://www.w3schools.com/java/java_classes.asp)[Interface](https://www.w3schools.com/java/java_interface.asp), etc.

**Java Conditions:**

Java has the following conditional statements:

* Use **if** to specify a block of code to be executed, if a specified condition is true
* Use **else** to specify a block of code to be executed, if the same condition is false
* Use **else if** to specify a new condition to test, if the first condition is false
* Use **switch** to specify many alternative blocks of code to be executed

**Methods in Java:**

Method declarations have six components:

**1. Modifier:** It defines the **access type** of the method i.e., from where it can be accessed in your application. In Java, there 4 types of access specifiers.

* **public:** It isaccessible in all classes in your application.
* **protected:** It is accessible within the class in which it is defined and, in its subclass.
* **private:** It isaccessible only within the class in which it is defined.
* **default:** It is declared/defined without using any modifier. It is accessible within the same class and package within which its class is defined.

**2. The return type:** The data type of the value returned by the method or void if does not return a value.

**3. Method Name:** the rules for field names apply to method names as well, but the convention is a little different.

**4. Parameter list:** Comma-separated list of the input parameters is defined, preceded with their data type, within the enclosed parenthesis. If there are no parameters, you must use empty parentheses ().

**5. Exception list:** The exceptions you expect by the method can throw; you can specify these exception(s).

**6. Method body:** it is enclosed between braces. The code you need to be executed to perform your intended operations.

**Static Method**

A method that has static keyword is known as static method. In other words, a method that belongs to a class rather than an instance of a class is known as a static method. We can also create a static method by using the keyword static before the method name.

**Instance Method**

The method of the class is known as an instance method. It is a non-static method defined in the class. Before calling or invoking the instance method, it is necessary to create an object of its class. Let's see an example of an instance method.

**Method Overloading**

Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters or both. Overloading is related to compile-time (or static) polymorphism.

**Method Overriding**

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java. If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding. Method overriding is used for runtime polymorphism.

**Inheritance in Java:**

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

**Exception Handling in Java:**

The Exception Handling in Java is one of the powerful mechanisms to handle the runtime errors so that the normal flow of the application can be maintained. In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

* Checked Exception
* Unchecked Exception
* Error

**Java Exception Keywords**

Java provides five keywords that are used to handle the exception. The following table describes each.

|  |  |
| --- | --- |
| Keyword | Description |
| try | The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally. |
| catch | The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later. |
| finally | The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not. |
| throw | The "throw" keyword is used to throw an exception. |
| throws | The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature. |

**Collections in Java:**

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 (Array List, Vector, [LinkedList](https://www.javatpoint.com/java-linkedlist), [Priority Queue](https://www.javatpoint.com/java-priorityqueue), HashSet, LinkedHashSet, Tree Set).

**ArrayList:** ArrayList provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. The size of an ArrayList is increased automatically if the collection grows or shrinks if the objects are removed from the collection. Java ArrayList allows us to randomly access the list. ArrayList can not be used for primitive types, like int, char, etc.

**Vector:** A vector provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. This is identical to ArrayList in terms of implementation.

**LinkedList:** LinkedList 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 addresses.

**Priority Queue:** A PriorityQueue is used when the objects are supposed to be processed based on the priority. It is known that a queue follows the First-In-First-Out algorithm, but sometimes the elements of the queue are needed to be processed according to the priority and this class is used in these cases. The PriorityQueue is based on the priority heap.

**HashSet:** The HashSet class is an inherent implementation of the hash table data structure. The objects that we insert into the HashSet do not guarantee to be inserted in the same order. The objects are inserted based on their hashcode.

**LinkedHashSet:** A LinkedHashSet is very similar to a HashSet. The difference is that this uses a doubly linked list to store the data and retains the ordering of the elements.

**TreeSet:** The TreeSet class uses a 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.

**Java Generics:**

Java **Generic** methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods, or with a single class declaration, a set of related types, respectively.Generics also provide compile-time type safety that allows programmers to catch invalid types at compile time.Using Java Generic concept, we might write a generic method for sorting an array of objects, then invoke the generic method with Integer arrays, Double arrays, String arrays and so on, to sort the array elements.

**Generic Classes**

A generic class declaration looks like a non-generic class declaration, except that the class name is followed by a type parameter section.As with generic methods, the type parameter section of a generic class can have one or more type parameters separated by commas. These classes are known as parameterized classes or parameterized types because they accept one or more parameters.

**Concurrency in Java:**

Java is a *multi-threaded programming language* which means we can develop multi-threaded program using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

**Life Cycle of a Thread:**

Following are the stages of the life cycle −

New − A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.

Runnable − After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

Waiting − Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.

Timed Waiting − A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.

Terminated (Dead) − A runnable thread enters the terminated state when it completes its task or otherwise terminates.

**Java Lambda Expressions:**

Lambda expression is a new and important feature of Java which was included in Java SE 8. It provides a clear and concise way to represent one method interface using an expression. It is very useful in collection library. It helps to iterate, filter and extract data from collection.

The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.

**Java Lambda Expression Syntax**

(Argument-list) -> {body}

Java lambda expression is consisted of three components.

1) Argument-list: It can be empty or non-empty as well.

2) Arrow-token: It is used to link arguments-list and body of expression.

3) Body: It contains expressions and statements for lambda expression.