Learning Report – Core Java

L&T Technology Services

Document History

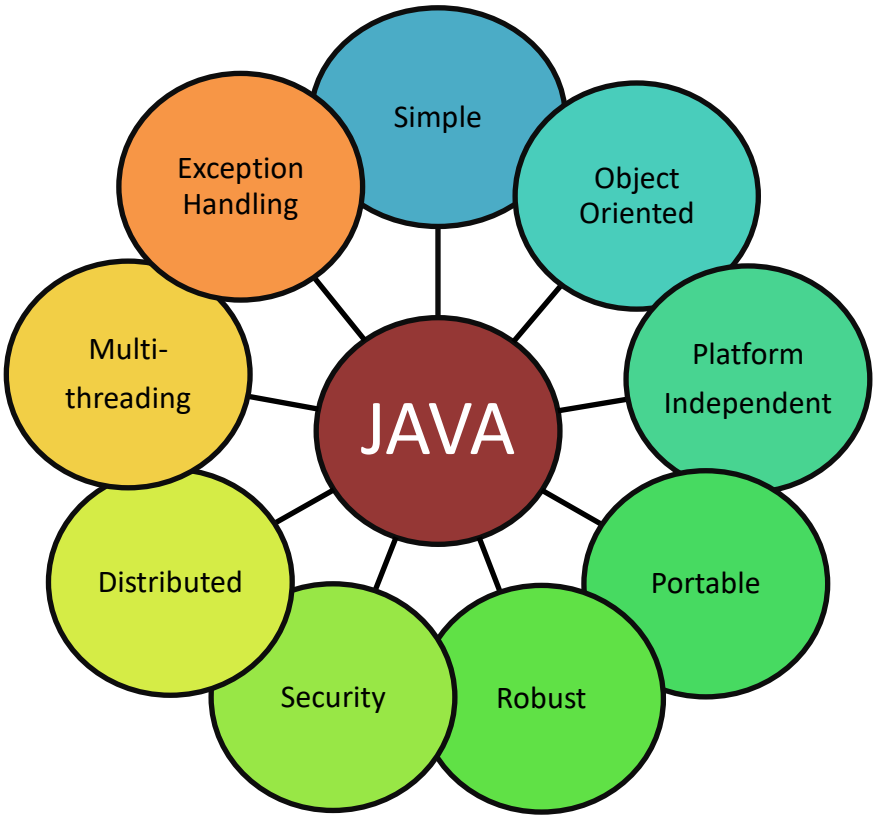
Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be approved By	Remarks/Revision Details
1.0		Sneha Anand			

Table of Contents

INTRODUCTION TO JAVA

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. It is a multi-platform, object-oriented and web-centric language. It is one of the most commonly used programming languages. Java is also used as a computing platform.

Java features



1. Simple.
2. Object Oriented
3. Platform Independent
4. Portable
5. Robust
6. Security
7. Distributed
8. Multithreading
9. Exception Handling

Java editions

1. J2SE (java 2 standard edition) or core java

It can be used to develop client-side standalone applications or applets

2. J2EE (java 2 Enterprise Edition) OR Advanced Java

It can be used to develop Server-side applications such as Java servlets, Java Server pages.

3. J2ME (Java 2 Micro Edition)

It is a development and deployment platform of portable code for embedded and mobile devices (sensors, gateways, mobile phones, printers, TV set-top boxes).

Java Virtual Machine (JVM)

JVM is basically specification that provides a runtime environment in which Java byte code can be executed i.e. it is something which is abstract and its implementation is independent to choose the algorithm

JVM performs the following tasks.

1. Load the code.
2. Verifies the code.
3. Executes the code.
4. Provide the runtime environment.

Java Runtime Environment (JRE)

JRE is the implementation of JVM i.e. the specifications which are defined in JVM are implemented and creates corresponding environment for the execution of code

Java Development Kit (JDK)

JDK a software development kit to develop applications in Java. In addition to JRE, JDK also contains number of development tools (compilers, JavaDoc, Java Debugger etc.). JDK is primarily used for code execution and has prime functionality of development.

About Java Program

- Java programs are written in text editors and program file is saved with the extension **.java**.
- To compile the java code use command javac followed by filename.
    - Syntax: **javac filename.java**
    - This will convert .java file to .class file if there is no error in the program. This .class file contains byte code.
- To execute the code use command java followed by the filename. Do not include any extension in filename.
    - Syntax: **java filename**

Structure of java program

```
Package declaration/statement;
Import statements;
Class Name_Of_Class
{
    Variable1;
    Variable2;
    .
    .
    .
    Variable N;
    Main Method Class {
    Method1 ()
    {
            Statements;
    }
    Method2 ()
    {
            Statements;
    }
    .
    .
    Method N ()
    {
            Statements;
    }
  }
}
```

- **Package Declaration :**

  A package is a namespace that organizes a set of related classes and interfaces. The package declaration is optional. It is placed just after the documentation section. In this section, the package name in which the class is placed is declared.

- **Import Statements :**

  The import statement represents the class stored in the other package. The import keyword is used to import the class. It is written before the class declaration and after the package statement.

- **Class Definition :**

  It is vital part of a Java program. A Java program may contain more than one class-definition. The class keyword to define the class. The class is a blueprint of a Java program. It contains information about user-defined methods, variables, and constants. Every Java program has at least one class that contains the main () method.

- **Class Variables and Constants :**

  In a Java program, the variables and constants are defined just after the class definition. The variables and constants store values of the parameters. It is used during the execution of the program.

- **Main method class :**

  Main method is essential for all Java programs. Because the execution of all Java programs starts from the main method. It serves as an entry point of the class, inside which we create objects and call the methods.

- **Methods :**

  The functionality of the program is declared by using the methods. The methods are the set of instructions that a user wants to perform. These instructions execute at runtime and perform the specified task.

JAVA BASIC FUNDAMENTALS

Identifier

Identifier is a name given to the variable, methods, class, interface, package etc.

Keywords

Keywords are the reserved words in the java program which cannot be used as an identifier in the program. For example if, else, while, for etc...

Data types

The data type specifies the different sizes and values that can be stored in the variable.

Primitive data types

Primitive is the most basic type and building blocks of data manipulation.
E.g.: int, float, char, double, long, byte, short, Boolean.

Non Primitive data types

Non primitive data types are formed using primitive data types to group similar type of data
E.g.: Arrays,Strings,

Variable

Variable is name to store a value in it. The variable name should start with alphabet or underscore (_) or dollar ($).

E.g.: int a;
float _q1;
char $c;

Types of variables:

Type	Description
Local variable	A variable which is declared inside the class and inside the method.
Instance variable	A variable which declared inside the class and outside the methods. For this type of variable access specifier should be mentioned.
Reference variable	Declare inside or outside the method. If you declare inside the method no access specifier allowed If you declare outside the method access specifier allowed. Created using the predefined or user defined class.
Static variable	The Variable declared inside the class and outside the method. It allows access specifier like public, private and protected. Object does not need to access this variable. Create using the primitive data type.

Literals

Literals are used for the representation of digits.

Eg: octal-> 028

Hexa-decimal -> 0x28

Binary -> 0b110

Operators

Arithmetic Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder

Relational Operator	Description
==	Equals
!=	Not Equal
<	Lesser than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Logical Operator	Expression	Description
&& (Logical AND)	expression1 **&&** expression2	true only if both expression1 and expression2 are true
|| (Logical OR)	expression1 **||** expression2	true if either expression1 or expression2 is true
! (Logical NOT)	**!**expression	true if expression is false and vice versa

CONTROL FLOW STATEMENTS

If statement

When certain statements need to be executed depending on the conditions, if statement can be used. Statement inside the "if" block is executed only when condition becomes true then.

Syntax:

```
If (condition)
    {
            Statements;
    }
```

If else statement

If else statement is similar to if statement where if condition becomes true "if" block statement can be executed. Otherwise else block statement is executed.

Syntax:

```
if(condition)
        {
            Statements;
        }
        else
        {
            Statements;
        }
```

Inline conditions

Inline statement is a single line command to write an if-else statement.

Syntax**:**

variable name = (condition)? true case : false case;

E.g.:

int a= (5>8)? 1:2;

CONTROL FLOW-> LOOPS

Loops are used to execute statement repeatedly depending upon the condition.

While loops

While loop executes statement repeatedly until the condition becomes False. The loop variable is a variable where the condition is checked on this variable. It should be updated in every iteration.

Syntax:

```
while (condition)
{
        Statements;
}
```

For Loop

For loop also execute statements repeatedly until the condition becomes False. It is used when we know how many times the loop should be executed. There are 3 sections in for loop. First is the initial condition which runs only once before execution of the code block. The second is the condition where the loop body will execute based on this. Last is the increment/decrement where the loop variable is updated after every iteration.

Syntax:

```
for (initial condition ; condition ; increment/decrement)
{
        Statements;
}
```

OBJECT ORIENTED PROGRAMMING

Java supports OOP, which is an example of a real-time problem-solving method. It binds the data and methods that operate. It increases the flexibility and maintainability of programs.

Classes

Class is a blueprint of a program. It contains variables and methods which are related to each other. It is group of variables and methods which operates on the variables defined in the group

Syntax:

```
<access_specifier> class <class_name>
        {
            member_variables;
            member_functions
        {
        }
    }
```

E.g.:

```
public class Employee
{
    private int age;
    public String name;
    protected float salary;

    int getAttendence()
    {
            int a;
            return(a);
    }
}
```

Objects

Object is an instance of a class. Classes are templates or blueprints from which objects are created. Therefore, an object is an instance (result) of a class.

Syntax:

<className> <objectName>=new <className>(arguments);

E.g.: Employee e1=new Employee ();

Access Modifier

Access Modifier modifies the visibility of the variable or method in the program, outside the class where it is defined. By default access modifier is public.

Access specifier	Description
private	Visible only to the class
public	Visible to all
protected	Visible to class and classes derived from it

Method

Method is a collection of statements. When certain set of statements is to be repeated whenever required, functions are used. Elements required to use a method is access specifier, return type, function name, parameter list and function body.

Syntax:

```
<accessSpecifier> <returnType> functionName(parameter1,parameter2)
{
    function body;
}
```

Data Encapsulation

Data encapsulation is hiding up of data into a single unit. To implement data encapsulation declare the variable as private. This variable is only accessed by the inside the class and its methods. Using public methods get and set data we can access data and set the value.

Function overloading

Defining function with same name but differ with parameters is called function overloading.

Constructor

Constructor is a member function of a class having same name as class name used to initialize members of the class. It has no return type and automatically called when the object is created.

Inheritance

Inheritance means one class inheriting the properties of another class. In technical terms derived class inherits variables and methods of the base class. It deals with IS-A and HAS-A relationship.

Eg:

```
Class Parent
{
    int a;
    int add(int x)
    {
            Return(a+x);
    }
}
Class Child extends Parent
{
    Int b;
}
```

Interface

Interface is a base class where it contains final static members, abstract methods, default methods, static methods and private methods. It is used to build loosely coupled system.

Eg:

```
interface Stu
{
    static int a;
    public void display();
}
class Student implements Stu
{
    public void display()
    {
            System.out.println(a);
    }
}
```

Abstract class

Abstract class is a base class which is not fully defined its functionality. Object of the abstract class cannot be created.

Eg:

```
Abstract class Stu
{
    float getPercentage();
    String getName();
}
class Student extends Stu
{
    float getPercentage()
    {
            return(this.Percentage);
    }
    String getName()
    {
            return(this.name);
    }
}
```

Final class

Final class is a class where any other class cannot inherit the final class. Object of final class can be created. Final class cannot be inherited.

Polymorphism

Polymorphism means “many forms”. It occurs when classes related to each other by inheritance.

There are two types polymorphism:

Compile time polymorphism: Function overloading is called as compile time polymorphism. It is function with same name but differ in parameters.

Run time polymorphism: Function overriding is called as run time polymorphism. It is only possible through inheritance. Run-time polymorphism is determined by JVM not compiler.

String

String in java is data type and a built-in class used to represent text. It is an array of characters. Java provides predefined functions to operate on strings. JVM creates a string pool to store all string literals. If any of the two strings have same value, then both points to same location. It reduces memory usage. Strings are immutable.

Eg:
String s=”JAVA”;

String Buffer

String Buffer is similar to string but mutable in nature. String is of fixed size and immutable whereas string buffer represents growable and writable objects. String buffer also provides in-built functions same as strings. In addition to that it also provides reverse, append, insert, length, capacity, etc.

Eg:
StringBuffer str=new StringBuffer(“java”);

String Builder

String Builder is same as String Buffer. It is mutable and resizable in nature. Difference between String Builder and String Buffer is String Builder is not thread safe.

Eg:
StringBuilder str=new StringBuilder(“java”);

Packages

Package is a collection of similar classes, interfaces, etc. for example we can keep classes related to input/output operation on one package and display in another package. Package helps in structuring the program.

Syntax:
Import <packagename>.<classname>;

EXCEPTION HANDLING

Exception is an error which occurs at run time. If exception are not handled program stops execution automatically.

Some exceptions are

- Dividing a number by zero
- Accessing array element beyond the size
- File not found during execution.

Types of exception

Checked exception: Exceptions which are identified at compile time is called checked exception. They are basically syntax errors.

Unchecked exception: Exceptions which occur at run time is called as unchecked exception. This can be handled by using try and catch block.

Syntax:

```
try
{
    statements;
}
catch(Exception e)
{
    <exceptionHandlingStatements>;
}
```

Eg:

```
Public static void main(String[] args)
{
    int a=10;
    b=0;
    int result;
    try{
            result=a/b;
            System.out.println(result);
    }
    catch(Exception e){
            System.out.println("Zero division error");
    }
}
```

MULTIPLE EXCEPTIONS

Multiple exceptions can be handled using multiple catch blocks.

Syntax:

```
try{
        Statements;
}
catch(ExceptionType-1 e){
        <exceptionHandlingStatements>;
}
catch(ExceptionType-2 e){
        <exceptionHandlingStatements>;
}
```

User defined exception

Java supports user defined exception where user can define exception depending upon the conditions. User defined exception class extends the Exception class.

Eg:

```
class Myexception extends Exception
{
    public String getMessage()
    {
        <exceptionHandlingStatements>;
    }
}
class Emp1
{
    public void main(String[] args)
    {
        try{
            if(b==0){
                Myexception e=new Myexception();
                Throw myexp;
            }
        }
        catch(Myexception e){
            System.out.println(e.getMessage());
        }
    }
}
```

JAVA COLLECTIONS

Java collections is set of java classes that allows to group the objects and manage them. It is a framework which helps in storing and processing the data efficiently.

ArrayList

ArrayList is an ordered sequence of elements stored in continues memory location. It is not fixed in size. It implements list interface.

Syntax:

ArrayList<datatype> arr=new ArrayList<datatype>();

Iterator

Iterator is object used to iterate over the collection objects. It has a function hasNext() which will return true if it has next element and next() will return next element.

Eg:

Iterator it=arrayList.iterator();

LinkedList

LinkedList is a linear data structure, but elements are not stored in continuous memory location. Each element in LinkedList is called a node. A node contain 2 elements. First the content of element and second element is address of the next element. It is more efficient than ArrayList in insertion and deletion of elements.

Syntax:

LinkedList<datatype> l1=new LinkedList<datatype>();

HashMap

HashMap is a type of collection used to store the key and value pairs. It uses hash map to store elements. It contains only unique keys. It not a ordered collection of elements.

Syntax:

HashMap<key,value> h1=new HashMap<key,value>();

HashSet

HashSet is a type of collection that uses hash table for storage. It implements set interface. It contains only unique elements and store elements by hashing. It is best for search operation.

Syntax:

HashSet<type> hs1=new HashSet<type>();

Activities (Github repo-link)

1. Java basics
   https://github.com/99003525/JAVA_genisis/tree/main/Basics

2. Classes and objects
   https://github.com/99003525/JAVA_genisis/tree/main/Class%20and%20objects

3. Exception handling
   https://github.com/99003525/JAVA_genisis/tree/main/Exceptions

4. Activity
   https://github.com/99003525/JAVA_genisis/tree/main/activity

5. Crud application
   https://github.com/99003525/JAVA_genisis/tree/main/CRUD%20app

6. JDBC
   https://github.com/99003525/JAVA_genisis/tree/main/JDBC