Unit-1:

Why Java?

→ Java is a programming language & a platform. (→ can handle error)

→ It is a high level, robust, object-oriented & Secure programming language.

→ Java has a runtime environment (JRE) and API, it is called a platform.

History of Java

1.) Java project initiated in 1991 by James Gosling, Mike sheriden & Patrick Naughton known as the Green Team at Sun microsystem.

2.) Initially designed for small embedded systems like Set-up boxes.

3.) Originally called "Green talk" with file extension ".gt".

4.) Renamed to Oak (Oak symbolizes strength) (National tree of several countries).

5.) Renamed to Java in 1995 due to trademark issues with Oak technologies.

6.) Named after Java Island in Indonesia, where the first coffee was produced.

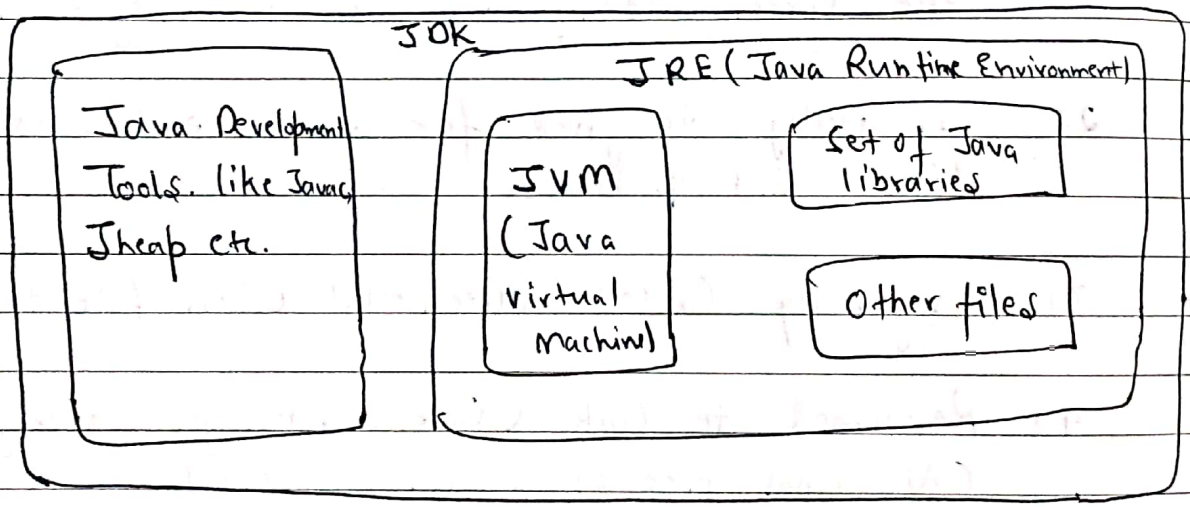
7.) Recognized by Time magazine as one of the Ten best Products of 1995.

Features of Java:-

→ Simple.
→ object Oriented.
→ Robust.
→ Platform Independent.
→ Secured.
→ Multithreaded → ek time pe alag-alag programme ek sath run kar sakte hai

JDK Vs JVM Vs JRE

(Java Development Kit)



1.) JDK

→ It's a Software development kit required to developed Java applications.

→ Includes the JRE, an interpreter, a compiler & other tools needed for Java development.

→ JDK is superset of JRE.

→ agar ap user ho you need → JRE
but if you are developer you need → JDK.

2.) JRE

→ It's a part of the JDK. but can be downloaded seperately.

→ Provides the libraries, the JVM and other components to run application.

→ Does not have tools & utilities for developers like Compilers.

3.) JVM

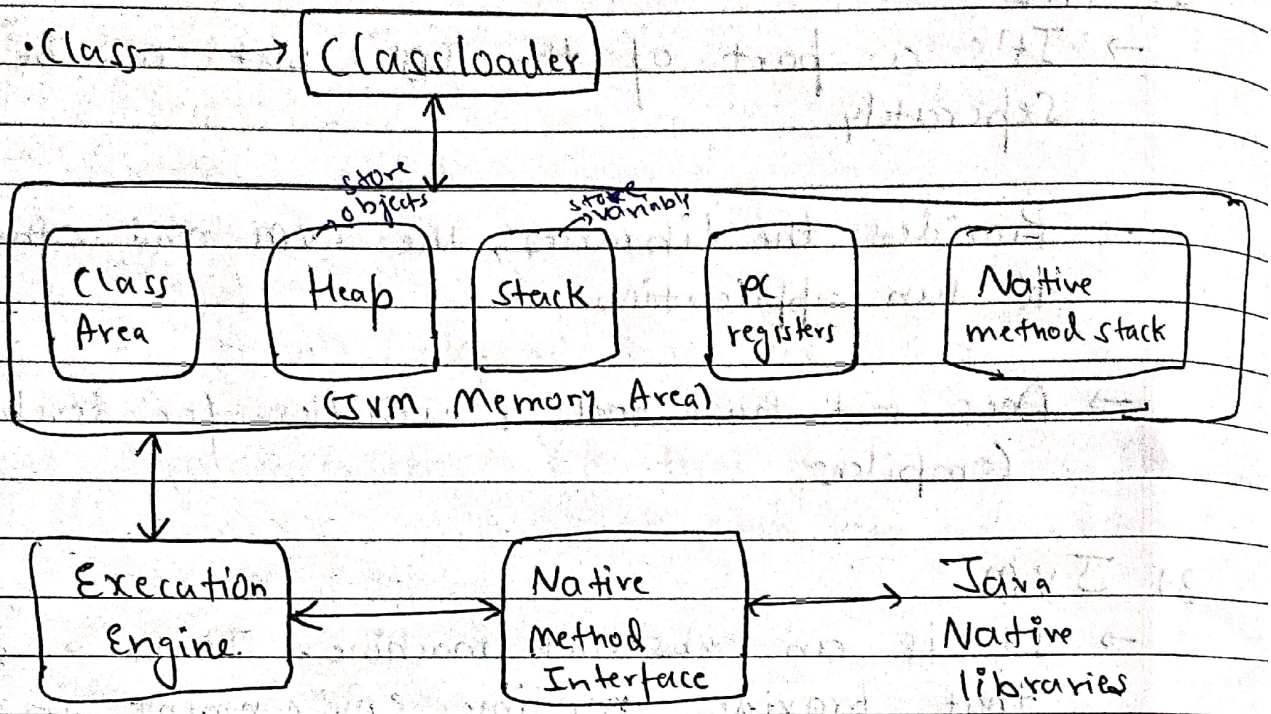
→ It is an abstract machine. It is a specification that provides runtime environment in which Java bytecode can be executed.

→ It is platform dependent. → (jaha JRE hoga vahi ye chalega)

→ The JVM performs following operation:
- Loads Code
- Verifies Code
- Executes Code
- Provide runtime environment.

* JVM Architecture:-

Let's understand the internal architecture of JVM.
It Contains: 1.) Classloader.
2.) Memory Area.
3.) Execution Engine.
4.) Native Method Interface.



→ Classloader → (Loads class file)
- → Bootstrap (rt.jar)
- → Extencion (ext)
- → System Application (Class path)

→ Class Area → Stores pre-class structure.

→ PC registers → Stores address & Currently executing instruction.
(Program Counter)

→ Execution Engine → Interpreter, compiler, all the functions of O.S.

Java Source file structure:-

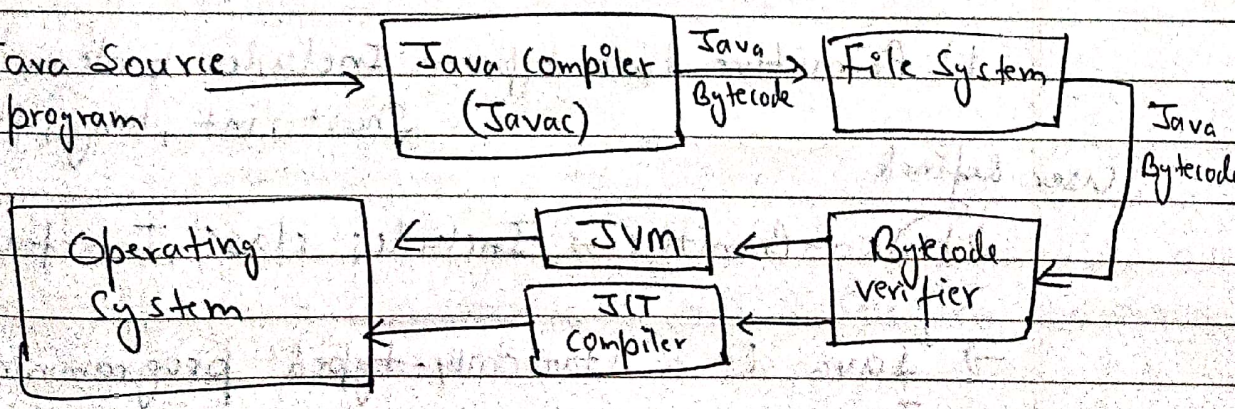
→ It is used to describe that the Java source code file must follow a structure.

→ The maximum no. of classes that may be declared as public in a Java program is one.

→ If a public class exists, the program's name and the name of the public class must match for there to be no compile time error.

Compilation & fundamentals:-

Write code in a text editor or IDE, Save as ".java", Compile with Javac, Generates ".class" bytecode files, Execute bytecode with Java command.

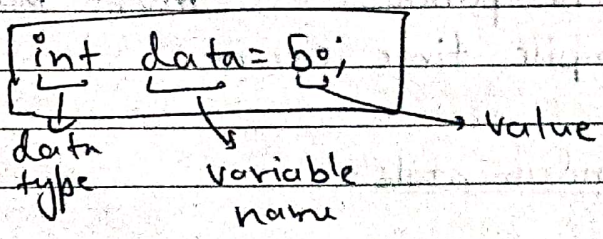


Fundamentals → class, object, Variables, Methods, Control flow statements, Packages.

→ For Writing a Java program we have to follow some rules. The set of these rules is called Syntax.

* Java Variables

A Variable is a container which holds value while the Java program is executed. A variable is assigned with a data type.



* Data Types in Java:-

→ Data types specify the different sizes & values that can be stored in variable.

1.) Primitive data types → Include boolean, char, byte, short, int, long, float & double

User defined

2.) Non-Primitive → Includes class, Interfaces & Arrays

→ Java is a statically-typed programming language. It means, all variables must be declared before it's use & it's datatype.

→ Java uses unicode System not ~~ASCII~~ Ascii code System. The ~~40000~~ 140000 is the lowest range of unicode System

Data types	Default Value	Default Size
boolean	False	1 byte
char	'\0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
Float	0.0F	4 byte
double	0.0D	8 byte

* Java Comments:-
→ Comments can be used to explain Java Code.
→ It can be also used to prevent execution when testing alternative code.

Single line → //- - - -

Multi line → /*- - -
- - - - */

* Java operators:- Operators are used to perform operation on variables & values.

→ Java divides operators into the following groups:-

1.) Arithmetic operators.
2.) Assignment "
3.) Comparison "
4.) logical "
5.) Bitwise "

1) Arithmetic operators → used to perform common mathematical operations.

operator	Name	Description	Example
+	Addition	Adds together two value	x+y
−	Subtraction	Subtracts one value from another	x−y
*	Multiplication	Multiplies two value	x*y
/	Division	Divides values	x/y
%	Modulus	Return the division remainder	x%y
++	Increment	increase value by 1.	x++, ++x
−−	Decrement	decrease value by 1.	x−−, −−x

2) Assignment operators → Assignment operators are used to assign values to variables.

Ex:			
	=	x=5	x=5
	+=	x+=5	x=x+5
	−=	x−=5	x=x−5
	*=	x*=5	x=x*5
	/=	x/=5	x=x/5
	%=	x%=5	x=x%5
	&=	x&=5	x=x&3
	|=	x|=3	x=x|3

3.) Comparison operators → used to compare two values (variables)
→ return value is 'True' or 'False'.

operator	Name	example
==	Equal to	x==y
!=	not equal to	x!=y
>	Greater than	x>y
<	less than	x<y
>=	Greater than or equal to	x>=y
<=	less than or equal to	x<=y

4.) logical operators → You can also test for true or false values with logical operators.

operator	Name	description	Example
&&	logical and	Returns true if both statements are true	x<5 && x<10
||	logical Or	Returns true if one of the statement is true	x<5 || x<4
!	logical not	Reverse the result, return False, if statement is true	!(x<5 && x<10)

Sol Bitwise operators → used to performing the manipulation of individual bits of a number.

→ They can be used with any integer type (char, short, int, etc).

operation	Result	Same as	Result
5&1	1	0101&0001	0001
5|1	5	0101|0001	0101
~5	10	~0101	1010
5<<1	10	0101<<1	1010
5^1	4	0101^0001	0100
5>>1	2	0101>>1	0010

Control flow in Java :-

→ Java Compiler executes the code from top to bottom.

→ Java provides statements that can be used to control the flow of Java code. Such statement are called control flow statements.

→ It is one of the fundamental features of Java, which provides a smooth flow of program.

* Java provides three types of control flow statement:-

1) Decision Making statements → 'if' statements and 'switch' statement

2) loops statements → for, while, do while, for-each loop.

3) Jump statements → 'Break' & 'continue' Statement.

('Break' → break the flow of program comes out of loop)

('continue' → skips to the next iteration skip that's one iteration)

Arrays in Java:-

→ Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in contiguous memory location.

→ We can store only a fixed set of elements in a Java array.

→ Advantages: 1) Code optimization → It makes the code optimize → we can retreive or sort the data efficiently.
2) Random access:- We can get any data at an index.

→ Disadvantages: 1) Size limit → We can store only the fixed size of elements in the array. It doesn't grow it's size at runtime.

String in Java:-

→ In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string.

→ How to create a string object?

- There are two ways to create String object
  1.) By string literal.
  2.) By new keyword.

OOPs (Object-Oriented programming System)

→ object means a real-world entity such as pen, chair, table, computer, watch etc.

→ object oriented programming is a methodology or paradigm to design a program using classes & objects. It simplifies software development & maintenance by providing some concepts.

(a style or approach for structuring & organizing code)

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

* Class:-

→ A class is a basic building block.
→ It can be defined as template that describes the data & behaviour associated with the class instantiation.
→ Instantiating is a class is to create an object (variable) of that class that can be used to access the member variables & methods of the class.

→ An object in Java is the physical as well as logical entity, whereas, a class in Java is a logical entity only.

→ Syntax:

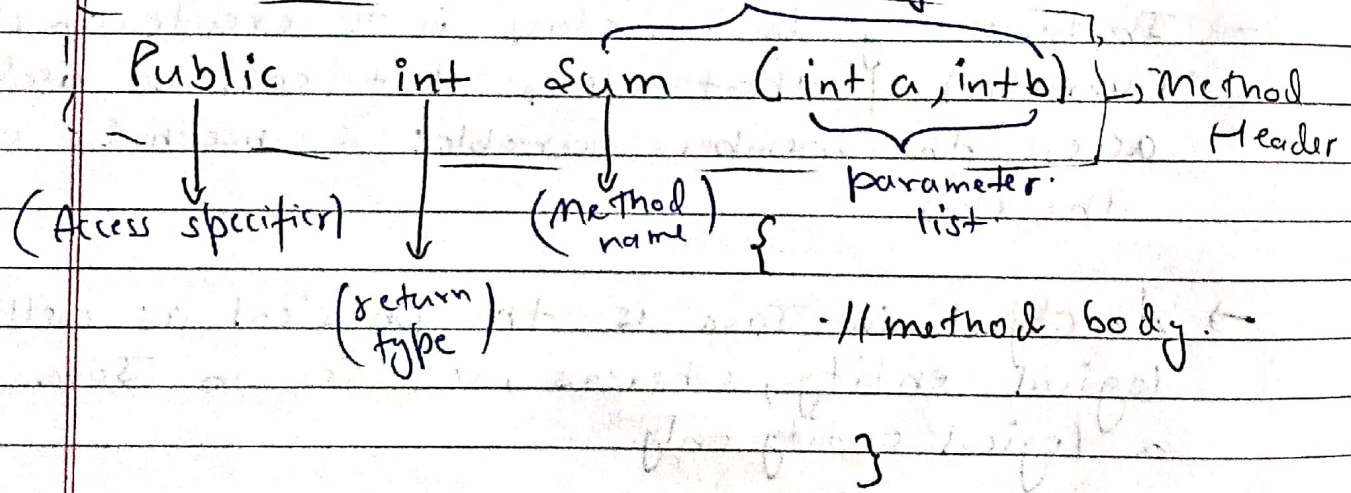
```
<access specifier> class class_name
{
.
.
}
```

What is method in Java?

→ A method is a reusable block of code designed to perform specific tasks. By calling or invoking it, we execute its functionality without needing to rewrite the code.

→ This promotes code reusability, easy modification and enhances code readability.

Method declaration



* Access specifier:

→ Access specifier or modifier is the access type of the method. It specify the visibility of the method. Java provides four access specifier

1.) Public → accessible by all classes, when we use public specifier in our application.

2.) Private → When we use a prive access specifier, the method is accessible only in the classes which is defined

3.) Protected:- When we use protected access specifier, the method is accessible within the same package or, subclasses in a different package.

4.) Default:- When we do not use any access specifier in the method declaration, Java uses default access specifier by default. It is visible only from the same package only.

Static Members:-

→ Static Members belong to the class rather than to any specific instance of the class.

→ They are shared among all instances of the class.

→ Static members can be accessed directly using the class name without needing to instantiate an object.

Final Members:- (Final variable can only be assigned while initialization (declared) or inside a constructor.)

→ Final members are constants that cannot be changed once initialized.

→ For variables, once a final variable is assigned a value, it cannot be reassigned.

→ For methods, a final method cannot be overridden by subclasses.

→ for classes, a final class cannot be subclasse

→ Final members are often used to define constants or to prevent further modificatio of certain elements in a program.

Constructor in Java:- (take values automatically)

→ A Constructor is a block of codes similar to the method. It is called when an instance of the class is created.

→ At the time of calling Constructor, memory for the object is allocated in the memory.

→ It is a special type of method which is used to initialize the object.

→ Rules for Creating Java Constructor:

1.) Constructor name must be the same as its class name.

2.) A Constructor must have no explicit return type.

3.) A Java constructor cannot be abstract, static, final & synchronized.

→ Types:- 1.) Default
2.) Parameterised.

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.

→ Why use inheritance in Java?

- For method Overriding (so runtime polymorphism can be achieved)
- For code reusability.

→ Terms used in Inheritance:-

- Class:- A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- Sub class/child class:- Subclass is a class which inherits the other class. It is also called derived class, extended or child class.
- Super class/Parent class:- Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

• **Reusability**:- As the name specifies, reusability is a mechanism which facilitates you to reuse the fields & methods of the existing class when you create a new class.

→ **The Syntax of Java Inheritance**:-

```
class subclass_name extends superclass_name
{
    //methods & fields.
}
```

→ The extend keyword indicates that you are making a new class that derives from an existing class.
→ The meaning of "extends" is to increase the functionality.

→ **Types of Inheritance**:

1) Single Inheritance
2) Multilevel "
3) Hierarchial "
4) Multiple "
5) Hybrid "

4, 5 → Practically not possible in Java.

Method Overloading:-

→ If a class has Multiple methods having same name but different in parameters, it is known as Method Overloading.

→ There are two ways to overload the method in Java
- By changing number of arguments.
- By changing the data type.

Method Overriding:-

→ If a Subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

→ Condition for method Overriding:-

1.) The method must have the same name as in the parent class.

2.) The method must have the same parameter as in the parent class.

3.) There must be an inheritance.

Encapsulation:-

→ Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.

→ We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter & getter methods to set & get the data in it.

→ The Java Bean class is the example of a fully encapsulated class.

Polymorphism:- (means many forms: "It allows one object or method to behave in multiple ways, depending on context.

→ Polymorphism in a Java is a fundamental concept in object-oriented programming (OOP) that allows objects of different classes to be treated as objects of a common superclass. This allows for flexibility & extensibility in your code.

(Think of it this way: you have group of superheroes with different ability, but they can all be referred to as "superheroes")

→ Two main types of polymorphism in Java:

1.) Compile-time → achieved through method overloading.
2.) Runtime Polymorphism → achieved through method overriding.

Abstraction:-

→ Abstraction in Java is another core concept of object-oriented programming (OOP), alongside encapsulation, inheritance, and polymorphism.

→ It involves hiding the complex implementation details of a system & showing only the essential features of the object.

→ Abstraction can be achieved in Java using two main mechanisms!

1.) Abstract class

2.) Interfaces

(Abstraction allows you to focus on what an object does rather than how it does it, promoting code reusability, maintainability, & flexibility in large software systems.)

* Abstract classes:- → (we cannot creat a object of abstract class)

→ An abstract class is a class that cannot be instantiated on its own and may contain abstract methods (methods without body) as well as concrete methods (methods with body)

→ Abstract methods serve as placeholders for methods that must be implemented by subclasses.

→ Abstract classes can have constructors & instance variables just like any other class.

* Interfaces:- (all properties are final).

→ An interface in Java is a blueprint of a class that defines a set of abstract methods.

→ Unlike Abstract classes, interfaces cannot have instance variables or constructors.

→ Classes can implement multiple interfaces, allowing them to inherit behavior from multiple sources.

Package in Java:- (by)

→ In Java, a package is a mechanism for organizing classes, interfaces, & sub-packages which helps in avoiding name conflicts & controlling access.

→ Packages are also used to group related classes & interfaces together, making the code more modular & easier to manage.

→ Package in Java can be categorized in two form, built in Package and user-defined package.

→ There are many built in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Classpath in Java:-

→ The Classpath is an environment variable used by the Java Compiler (javac) and the java runtime (Java) to locate classes that are referenced in your program. Setting the Classpath correctly is crucial for compiling & running Java programs that are organized into packages.

→ We can set Classpath in various ways:
- a) Setting CLASSPATH Temporarily.
- b) Setting CLASSPATH Permanently.

→ The variable used for this is 'CLASSPATH'.

a) Setting CLASSPATH Temporarily:-

On Windows:- ~~Set~~ set PATH = C:\Program Files\Java\JDK1.6.0\bin

b) Setting CLASSPATH Permanently:

On window:-

1.) Right-click on This PC or My Computer & Select Properties.

2.) Click on Advanced System Settings.

3.) Click on the Environment Variables button.

4.) In the System variables Section, click on New or select the existing CLASSPATH variable & click Edit.

5.) Add your paths to the CLASSPATH variable.

JAR Files in Java:

→ A JAR (Java Archive) is a package file format typically used to aggregate many Java class files & associated metadata & resources (text, images, etc) into one file to distribute application software or libraries on the Java platform.

→ In simple words, a JAR file is a file that contains a compressed version of .class files, audio files, image files, or directories. We can imagine a .jar file as a zipped file (.zip) that is created by using WinZip software. Even, Win2ip software can be used to extract the contents of a .jar. So you/we can use them for tasks such as lossless data compression, archiving, decompression, & archive un packing.

→ Create JAR File:

Syntax:

```
jar cf jar-filename input-files
```

Here, cf represents to create the file. For example, assuming our package pack is available in C:\ directory, to convert it into a jar file into the pack.jar, we can give command as:

```
C:\> jar cf pack.jar pack
```

Naming Conventions for Java Packages:-

For avoiding unwanted package names, we have some following naming conventions which we use in creating a package.

1.) Use lowercases letters:
E.g:- package com.example.myapp;

2.) Period-Delimited Structure: → same name na ho
E.g: package com.example.myapp.controllers;

3.) Base names on Company or Organization:
E.g: package com.example.myapp.services;

import and static import statement

→ Java 5.0 introduced static import, also known as the tiger version, as a new language feature. Static variables & static members can now be accessed directly without having to use class names.

→ We had normal import statements for every java class before static import was introduced we used class names to call methods & classes. Below is an example of a normal import for the list class:

```
import java.util.List;

import static java.lang.Math.*;
```

Unit-2

* Exception handling
* Input/output Basics
* Multithreading

Exception Handling:-

Exception:- It is an event that disrupts the normal flow of a program due to an error or unexpected condition.

→ The Idea Behind Exception:-

* The idea behind Exceptions in programming is to provide a robust mechanism to handle errors & other unexpected conditions that occur during the execution of a program.
* This helps in maintaining the normal flow of the application even when an error occurs.

Key concepts:-

1.) Error Detection & Management.

2.) Separation of Error handling code.

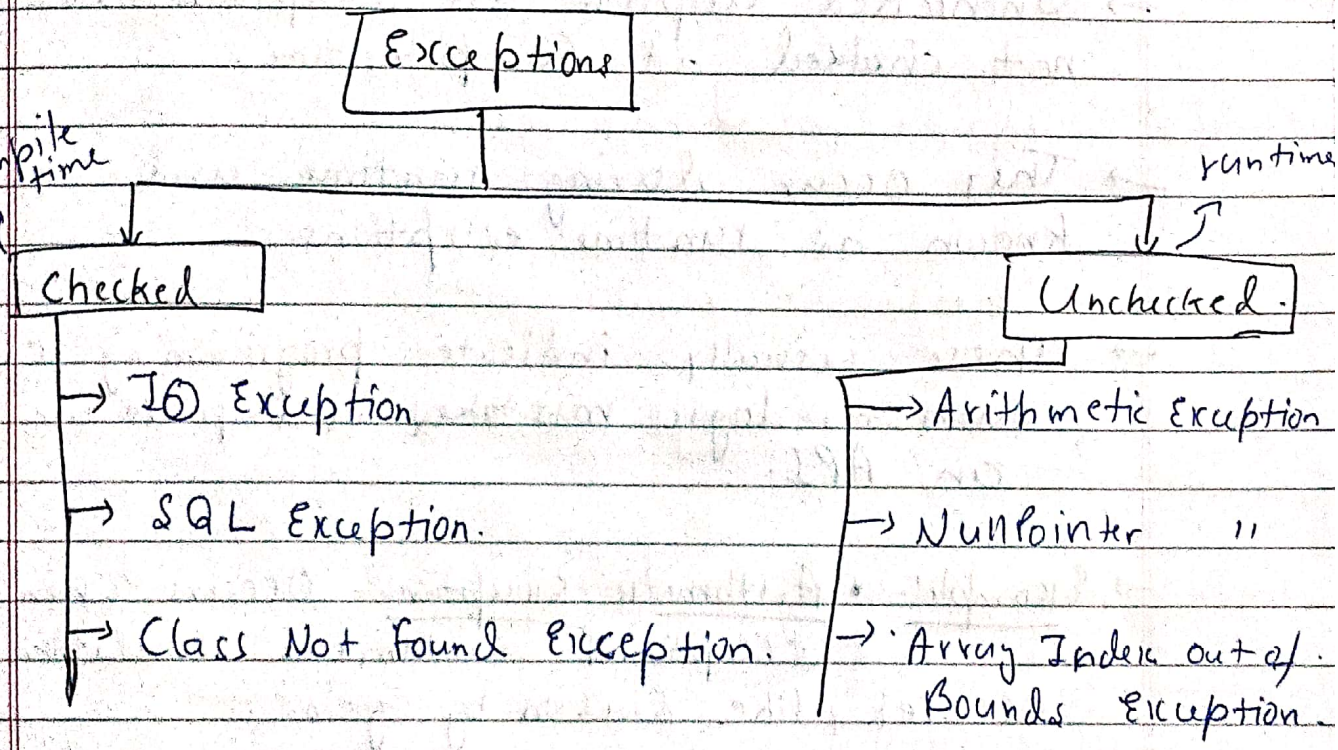
3.) Resource Management.

Exceptions & Errors:-

Exceptions	Errors
→ Generally recoverable & can be handled by the program.	→ Generally not recoverable & cannot be handled.
→ It is application-level issues.	→ It is a system level issues.
→ 'IO Exception', 'NULL Pointer Exception'	'Out of Memory Error', 'StackOverFlow Error'.
Handling mechanism → 'try', 'catch', 'finally', 'throw'	→ usually cannot be handled within the application.
→ belongs to java.lang.Exception package.	→ belongs to java.lang.Error package.

* Exceptions are manageable conditions that a program can handle, while errors represent serious issued that are typically beyond the application.

→ Types of exception



1.) Checked Exception:-

→ Checked exceptions are exceptions that are checked at compile-time.

→ If a method can throw a checked exception, it must either handle with 'try' catch or 'throws'

→ Example:-

→ IO Exception: Occurs during i/p & o/p operations, like file reading or writing.
→ SQL Exception: Related to database access errors.
→ Class Not found Exception: definition for the class with the specified name could not found.

2.1 Unchecked Exception:

→ Unchecked Exceptions are exceptions that are not checked at compile time

→ They occur during runtime and are also known as runtime exceptions.

→ These usually indicate programming errors, such as logic mistakes or improper use of an API.

→ Example:- • Arithmetic Exception: Occurs when an exceptional arithmetic condition has occured, like division by zero.

• Array Index out of Bounds Exception: Thrown to indicate that an array has been accessed with an illegal index.

Control flow in Exceptions:

→ The Control flow in exception handling in java involves a sequence of steps to manage exceptions when they occur.

→ It ensures that the program can handle errors gracefully and continue its execution or terminate in a controlled manner.

→ Control Flow steps:

1.) try block:

- Code that might throw an exception is placed inside a 'try' block.
- If no exception occurs, the 'catch' block is skipped, and control flows to the next statement after the 'try-catch-finally' block.

```
try {
    // Risky code that might throw an exception
    }
```

2.) Catch block:

- If an exception occurs in the 'try' block, control is transferred to the corresponding 'catch' block.
- The 'catch' block catches the exception and executes the code inside it.

```
Catch (ExceptionType e) {
    // Code to handle the exception.
    }
```

3.) **finally block:**

→ A 'Finally' block contains code that is always executed, regardless of whether an exception was thrown or not.

→ It is typically used for cleanup activities, such as closing files or releasing resources.

```
finally{
    // Code that will always execute
}
```

4.) **throw statement:-** (less used)

→ The 'throw' statement is used to explicitly throw an exception.

```
throw new ExceptionType("Error message");
```

5.) **throws clause:**

→ The throws clause in a method declaration specifies which exceptions might be thrown by the method.

```
public void riskyMethod() throws ExceptionType{
    // Method Code.
}
```

JVM reaction to Exceptions:-

→ When an exception occurs during the execution of a Java program, the Java virtual Machine (JVM) follows a well-defined process to handle the exception.

Step-by-step explanation of how the JVM reacts to exceptions:-

1) Exception Thrown:-

→ When an exception occurs (either explicitly using the 'throw' statement or implicitly due to an error), the JVM creates an exception object.

Exception object contains:
- type of exception.
- the state of program
- a stack trace.

2) Search for Matching Catch Block.

→ Current Method: The JVM first checks if there is a 'try-catch' block in the current method that can handle the exception.

→ Calling Method:- If no matching 'Catch' block is found in the current method, the JVM propagates the exception to the calling method, continuing this process upto Call Stack.

3.) Propagation upto the Call Stack:-

→ If no matching 'catch' block is found in any of the methods in the call stack, the exception continues to propagate until it reaches the main method.

4.) Uncaught Exception Handler:-

→ If the exception reaches the main method & still remains unhandled, the JVM's default uncaught exception handler takes over.

- Print stack Trace → exception type & sequence of method
- Terminates Program → terminates the program abnormally.

Inbuilt and User-Defined Exceptions:

1.) Inbuilt Exception:

→ Inbuilt exceptions are pre-defined in the Java API.

→ These exceptions are derived from the 'Throwable' class, and they cover a wide range of standard error conditions.

* Checked Exception.
* Unchecked Exception.
* Errors.

2.) User-Defined exceptions:

→ User-Defined Exceptions are custom exceptions created by developers to handle specific error conditions that are not covered by in-built exceptions.

→ These are created by extending the 'Exception' class or any of its subclasses.

→ Steps to create a User-Defined Exception:

(i) Define the exception class:-

- Extend the 'Exception' class for a checked exception or 'RuntimeException' for an unchecked exception.
- Provide constructors that pass custom messages to the superclass.

(ii) Throw the Exception:

- Use the 'throw' keyword to create and throw an instance of your exception class when the specific error condition occurs.

(iii) Handle the exception:-

- Use 'try-catch' block to catch & handle your user-defined exception appropriately.