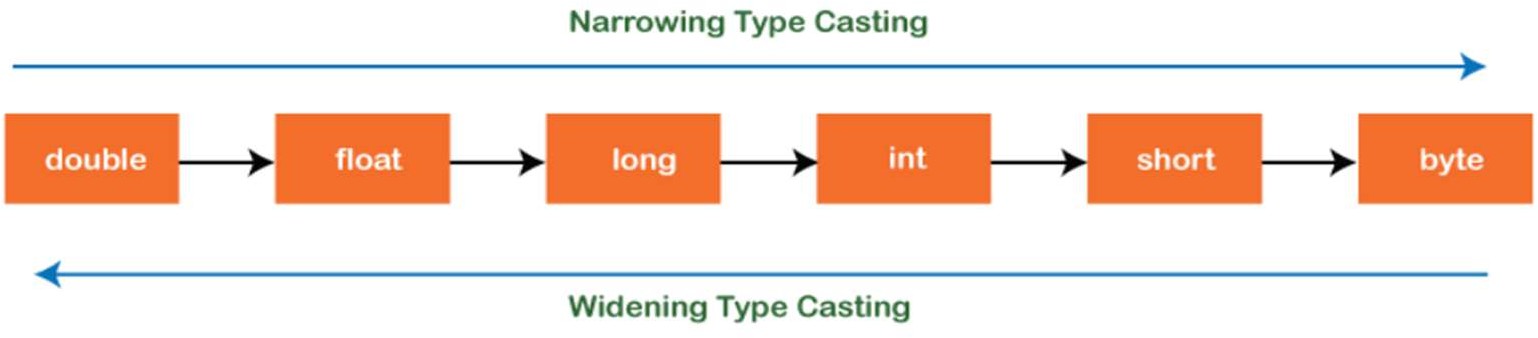
# ACTIVITY-2



**1 (A) Study and Present type casting in java.**

Type casting in Java refers to converting a value of one data type into another in both ways manually and automatically. The automatic conversion is done by the compiler and manual conversion performed by the programmer. This is done to ensure compatibility when working with different types of data or when passing data between different parts of a program.



## Type casting in Java

There are two types of type casting:

* Widening Type Casting
* Narrowing Type Casting

## Widening Type Casting

Converting a lower data type into a higher one is called widening type casting. It is also known as implicit conversion or casting down. It is done automatically. It is safe because there is no chance to lose data. It takes place when:

* + Both data types must be compatible with each other.
  + The target type must be larger than the source type.

## byte -> short -> char -> int -> long -> float -> double

For example, the conversion between numeric data type to char or Boolean is not done automatically. Also, the char and Boolean data types are not compatible with each other.

WideningTypeCastingExample.java

public class Widening {

TypeCasting Example

public static void main(String[] args)

{

int x = 7;

//automatically converts the integer type into long type

long y = x;

//automatically converts the long type into float type

float z = y:

System.out.println("Before conversion, int value "+x);

System.out.println("After conversion, long value +y);

System.out.println("After conversion, float value "+z):

}

}

Output :



Before conversion, the value is: 7

After conversion, the long value is: 7

After conversion, the float value is: 7.0

In the above example, we have taken a variable x and converted it into a long type. After that, the long type is converted into the float type.

## Narrowing Type Casting

Converting a higher data type into a lower one is called narrowing type casting. It is also known as explicit conversion or casting up. It is done manually by the programmer. If we do not perform casting then the compiler reports a compile-time error.

## double -> float -> long -> int -> char -> short -> byte

For example, we have performed the narrowing type casting two times. First, we have converted the double type into long data type after that long data type is converted into int type.

Narrowing TypeCasting Example.java

public class Narrowing TypeCasting Example

public static void main(String args[])

(

double d 166.66;

//converting double data type into long data type

long (long)d;

//converting long data type into int data type

int i = (int)l;

System.out.println("Before conversion: "+d);

//fractional part lost

System.out.println("After conversion into long type: "+1);

//fractional part lost

System.out.println("After conversion into int type: "+i);

}

Output

Before conversion: 166.66

After conversion into long type: 166

After conversion into int type: 166

# 1 (B) Study and present what are command line arguments in Java.



The java command-line argument is an argument i.e. passed at the time of running the java program. The arguments passed from the console can be received in the java program and it can be used as an input.

So, it provides a convenient way to check the behaviour of the program for the different values. You can pass N (1,2,3 and so on) numbers of arguments from the command prompt.

## Simple example of command-line argument in java:

In this example, we are receiving only one argument and printing it. To run this java program, you must pass at least one argument from the command prompt.

class CommandLineExample(

public static void main(String args[]){

System.out.println("Your first argument is: "+args[0]);

}

}

compile by > javac CommandLineExample.java

run by > java CommandLineExample sonoo

Output: Your first argument is: sonoo

## Example of command-line argument that prints all the values:

In this example, we are printing all the arguments passed from the command-line. For this purpose, we have traversed the array using for loop.

class Al

public static void main(String args[]){

for(int i=0;i<args.length;i++)

System.out.println(args[i]);

1

compile by javac A.java

run by java A sonoo jaiswal 13 abc

Output:

sonoo

Jaiswal

1

3

abc

# 1 (C) Study and present java keywords and their usage.



Java keywords are special reserved words that have pre-defined meaning and functionality in the Java programming language. Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name.

## Categories of Keywords:

1. **Control Keywords**
   * Control flow of your program

## Examples: if, else, switch, case, default, break, continue, return

1. **if, else:**

**Usage:** Used for conditional execution of code blocks.

## switch, case, default:

**Usage:** Provides a way to select one of many code blocks to be executed.

## break:

**Usage:** Terminates the loop or switch statement and transfers control to the statement immediately following the loop or switch.

## continue:

**Usage:** Skips the current iteration of loop and proceeds to the next iteration.

## return:

**Usage:** Terminates the execution of a method and returns a value (if specified).

## Modifier Keywords

* + Modify behavior of classes, methods, variables, etc.

## Examples: public, private, protected, static, final, abstract, synchronized

1. **Access Modifiers:**
   * **public:** Accessible from anywhere.
   * **private:** Accessible only within the same class.
   * **protected:** Accessible within the same package and subclasses.

## Non-Access Modifiers:

* + **final:** Indicates that a variable's value cannot be changed or that a method cannot be overridden or a class cannot be extended.
  + **abstract:** Used in abstract classes and methods. An abstract method has no implementation and must be overridden by subclasses.
  + **synchronized:** Used in multi-threaded programming to ensure that only one thread can access a synchronized method or block at a time.



## Primitive Data Types

* + Define primitive data types

## Examples: byte, short, int, long, float, double, boolean, char

1. **Integer Types:**
   * **byte:** 8-bit signed integer. Range: -128 to 127.
   * **short:** 16-bit signed integer. Range: -32,768 to 32,767.
   * **int:** 32-bit signed integer. Range: -2^31 to 2^31 - 1.
   * **long:** 64-bit signed integer. Range: -2^63 to 2^63 - 1.

## Floating-Point Types:

* + **float:** 32-bit floating-point number. Range: approximately ±3.40282347E+38F.
  + **double:** 64-bit floating-point number. Range: approximately

±1.79769313486231570E+308.

## Character Type:

* + **char:** 16-bit Unicode character. Represents a single character.

## Boolean Type:

* + **boolean:** Represents true or false.

## Reserved Keywords

* + Reserved for future use
  + Examples: **goto, const**

There are only 2 reserved keywords currently **- goto and const**

1. **goto -** Possibly meant for implementing goto statements in future
   * Not used currently and Java does not have goto statements
2. **const -** Possible usage to define constant/immutable variables
   * Currently not used
   * Alternatives like final keyword exist