EXPERIMENT 8

// Introduction to Simple Method Overloading and Type Casting in Java.

Prerequisite Knowledge -

Method Overloading - When two or more methods (functions) in a program have the same name but different definitions, the phenomenon is known as method overloading. Due to this overloading, same named functions can be used differently in a program.

- Method overloading depends upon the number and types of parameters that are defined for different methods with the same name.
- Whenever there is a call to an overloaded function, the compiler looks for a match between the number and type of arguments passed to the function with the formal parameters mentioned in different function definitions. The definition where this match is found is executed for the called function.

Type Casting - When a value of one data type is casted i.e., converted into some other data type, the phenomenon is known as type casting.
There are two types of type casting:

- **Widening or Implicit Type Casting -**
  When a value of a smaller data type is assigned to an entity of a larger data type, the value is automatically enhanced as per the format of the larger data type. This phenomenon is called widening type casting and it happens implicitly in the system.
- **Narrowing or Explicit Type Casting -**
  When a value of a larger data type is assigned to an entity of a smaller data type, the value may be truncated as per the format of the smaller data type. This is known as narrowing type casting and is done explicitly by the programmer.

  Syntax:

  ```
  small_datatype_var = (small_datatype) large_datatype_var;
  ```

  Example:

  ```
  double b=10.009;
  int a = (int) b;
  ```

PROGRAM 8

/* To implement a class 'OverloadDemo' that has two data members 'num1' and 'num2', an overloaded member function 'GetValue()' that assigns different types and number of values to num1 and num2, and a member function 'ShowValue()' that displays the values of num1 and num2 on the screen. */

Source Code -

```
class OverloadDemo
{
   float num1,num2;

   void GetValue(float n1,float n2)
   {
      num1=n1; num2=n2;
   }

   void GetValue(int n1,int n2)
   {
      num1=n1; num2=n2;
   }

   void GetValue(double n)
   {
      num1=(float)n;
      num2=(float)n;
   }

   void ShowValue()
   {
      System.out.println("Numbers are = " + num1 + " and " + num2);
   }
}

class MainClass
{
        public static void main(String args[])
        {
```

```
                OverloadDemo od1 = new OverloadDemo();

                od1.GetValue(1.5f,2.5f);
                od1.ShowValue();

                od1.GetValue(3,4);
                od1.ShowValue();

                od1.GetValue(5.6000009);
                od1.ShowValue();
        }
}
```

Output -

```
Numbers are = 1.5 and 2.5
Numbers are = 3.0 and 4.0
Numbers are = 5.600001 and 5.600001
```