Guided LAB 303.2.2 - Core Java: Type Casting

Objective:

In this lab, we will demonstrate Java Type Casting and its types with the help of examples. Before you learn about Java Type Casting, make sure you know about Java Data Types.

Learning Objective:

By the end of this lab, the learner will be able to use Type casting in Java.

Introduction

There are two castings possible in Java:

1. Implicit type casting (also known as automatic type conversion).
2. Explicit type casting.

Implicit Type Casting

Automatic conversion (casting) done by the Java compiler internally is called implicit conversion or implicit type casting in Java. Implicit casting is performed to convert a lower data type into a higher data type. It is also known as automatic type promotion in Java.

Example: Implicit Type Casting

Create a Java class named `AutoTypeConversion` and write the code below in that class.

```
public class AutoTypeConversion {
   public static void main(String[] args) {
       int x = 20;
       double y = 40.5;
```

```
        long p = 30;
        float q = 10.60f;
        // int z = x + y; (1) // Error; cannot convert from double to int.
        double z = x + y;
        System.out.println("Sum of two numbers: " + z);

    // long r = p - q; // (2) // Error; cannot convert from float to long.
        float r = p - q;
        System.out.println("Subtraction of two numbers: " + r);
    }
}
```

Output:

```
Sum of two numbers: 60.5.
Subtraction of two numbers: 19.4.
```

Explanation:

1. In the above example, the result of addition is double because when *int* and *double* are added, int is promoted to the higher-ranking data type than double. Therefore, assigning the result as a double is legal.

When the result assigns to be an int, the fractional part of the value of y will be truncated, and the result will give the value of 60, not 60.5. It represents a loss of precision, and Java does not allow any loss of precision. Therefore, the compiler will generate an error in line number 1.

2. Similarly, the result of subtraction is float because when long and float are subtracted, long is promoted to the higher-ranking data type float.

Example: Automatic Type Promotion

```
public class AutoPromoteTest {
   public static void main(String[] args)
   {
        byte b = 42;
        char c = 'a';
        short s = 1024;
        int i = 50000;
        float f = 5.67f;
        double d = .1234;
// Expression:
        double result = (f * b) + (i / c) - (d * s);
//Result after all the promotions are done.
        System.out.println("result = " + result);
   }
}
```

Output:

```
result = 626.7784146484375
```

Explanation:

1. "The letter (a) converted to binary via ASCII is 01100011. The decimal representation of 0110011 (64 + 32 + 2+ 1) is 97."

2. In the first sub-expression, f * b, b is promoted to float and the result of sub-expression is float.

3. In the second sub-expression i / c, the first c is promoted to int, and the result of the subexpression will be int.

4. Then, in d * s, the value of s is promoted to double, and the data type of subexpression is double.

Finally, we will now consider these three intermediate values with data types: float, int, and double. When the addition of a float and an int is performed, the outcome is float.

Then the resultant float is minus with the last double, which is converted to double, which is the data type of the final result of the expression.

Explicit Type casting (Narrowing conversion)

The process of converting lower data types into higher data types is called **Widening** or **Narrowing conversion** in Java.

Example: Explicit Type casting

```
public class ExplicitTest {
   public static void main(String[] args) {
       double d = 100.04;
       // explicit type casting
       long l = (long)d;
       int i = (int)l;
       System.out.println("Double value "+d);
       System.out.println("Long value "+l);
       System.out.println("Int value "+i);

byte b;
int z = 257;
double dou = 323.142;
System.out.println("Conversion of int to byte.");
b = (byte) z;
System.out.println("z = " + z + " b = " + b);
System.out.println("Conversion of double to int.");

z = (int) dou;
System.out.println("dou = " + dou + " b = " + z);
System.out.println("Conversion of double to byte.");

b = (byte) dou;
System.out.println("dou = " + dou + " b = " + b);

   }
  }
```

Output:

```
Double value 100.04
Long value 100
Int value 100
Conversion of int to byte.
i = 257 b = 1
Conversion of double to int.
dou = 323.142 b = 323
Conversion of double to byte.
dou = 323.142 b = 67
```

Explanation

In this example program, when the value 257 is cast into a byte variable. The output is 1, which is the remainder of the division of 257 by 256 (the range of a byte). When “dou” is converted into an int, its fractional part is lost.

When “dou” is converted into a byte, its fractional part is also lost, and the value is reduced to module 256, which is 67 in this case.

Disadvantages of Explicit Type Casting in Java

Disadvantages of using type casting in Java:

- You can lose some information or data.
- Accuracy can be lost while using type casting.
- When a double is cast to an int, the fractional part of a double is discarded, which causes the loss of the fractional part of data.

Submission Instructions:

Include the following deliverables in your submission -

- Submit your source code using the Start Assignment button in the top-right corner of the assignment page in Canvas.