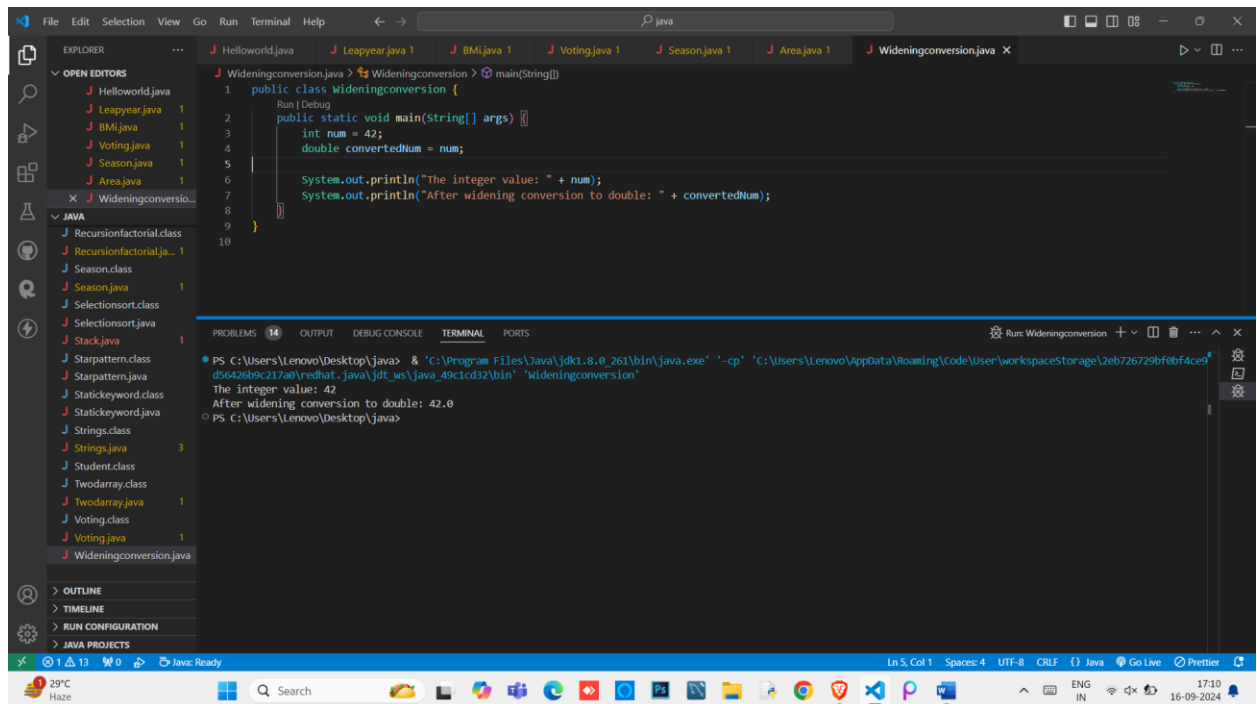


## CDAC Mumbai PG-DAC August 24

### Assignment No- 4

- 1) Write a program that demonstrates widening conversion from int to double and prints the result.



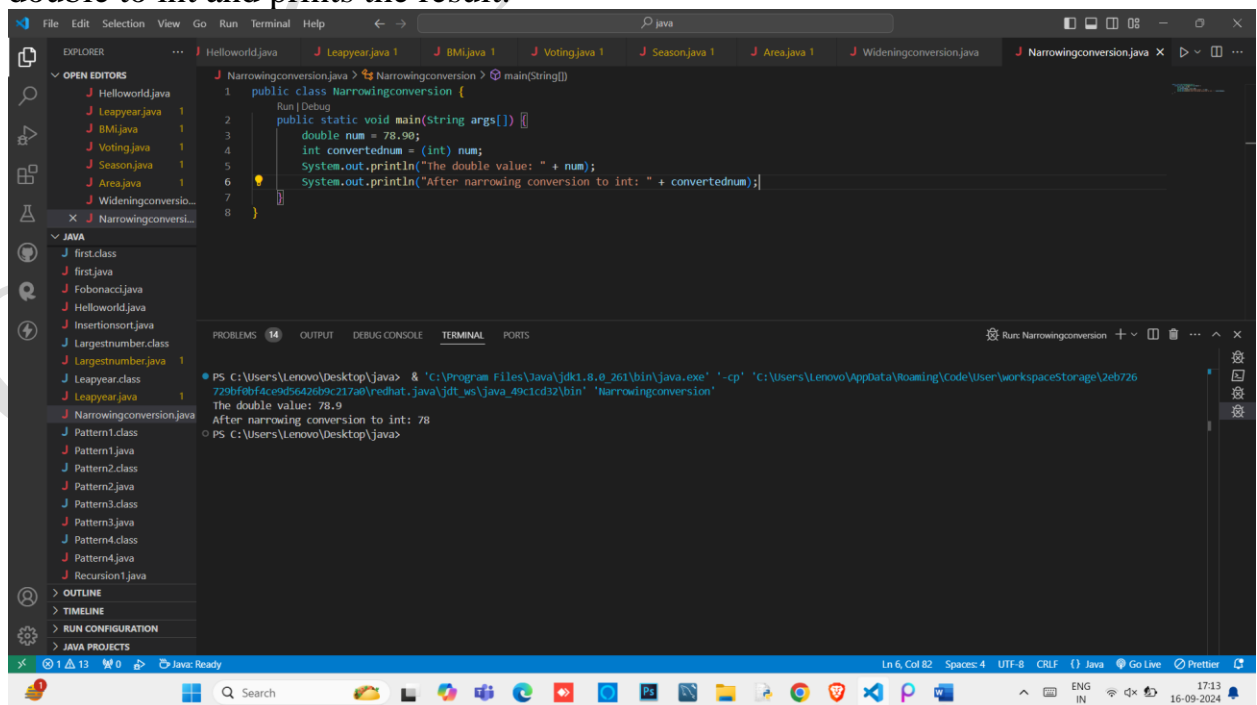
The screenshot shows an IDE with the following components:

- EXPLORER:** A list of Java files including `Helloworld.java`, `Leapyear.java`, `BMI.java`, `Voting.java`, `Season.java`, `Area.java`, and `Wideningconversion.java`. The `Wideningconversion.java` file is selected.
- EDITOR:** The code for `Wideningconversion.java` is displayed:

```
1 public class Wideningconversion {
2     public static void main(String[] args) {
3         int num = 42;
4         double convertedNum = num;
5
6         System.out.println("The integer value: " + num);
7         System.out.println("After widening conversion to double: " + convertedNum);
8     }
9 }
10
```
- TERMINAL:** The output of the program is shown:

```
PS C:\Users\Lenovo\Desktop\java> & "C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" "-cp" "C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\2eb726729bf0b4ce9d5426b9c217a0\redhat.java\jdk1.8.0_261\bin" "Wideningconversion"
The integer value: 42
After widening conversion to double: 42.0
PS C:\Users\Lenovo\Desktop\java>
```

- 2) Create a program that demonstrates narrowing conversion from double to int and prints the result.



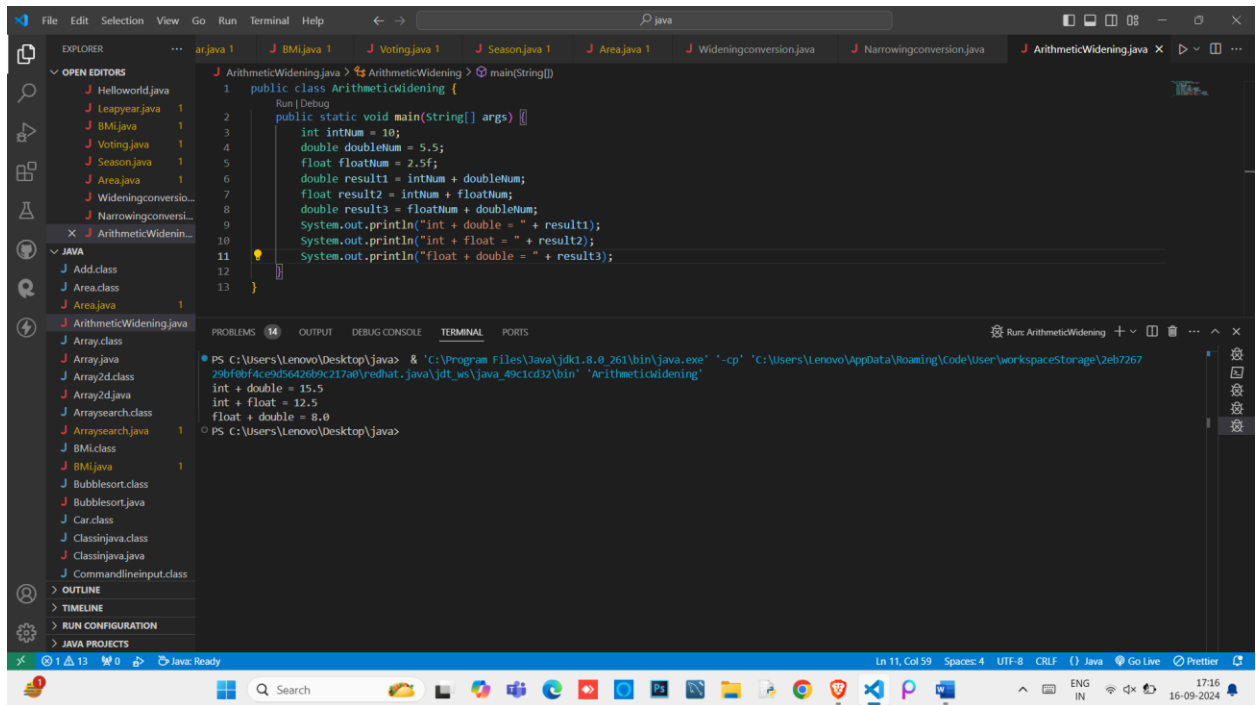
The screenshot shows an IDE with the following components:

- EXPLORER:** A list of Java files including `Helloworld.java`, `Leapyear.java`, `BMI.java`, `Voting.java`, `Season.java`, `Area.java`, `Wideningconversion.java`, and `Narrowingconversion.java`. The `Narrowingconversion.java` file is selected.
- EDITOR:** The code for `Narrowingconversion.java` is displayed:

```
1 public class Narrowingconversion {
2     public static void main(String[] args) {
3         double num = 78.90;
4         int convertednum = (int) num;
5         System.out.println("The double value: " + num);
6         System.out.println("After narrowing conversion to int: " + convertednum);
7     }
8 }
```
- TERMINAL:** The output of the program is shown:

```
PS C:\Users\Lenovo\Desktop\java> & "C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" "-cp" "C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\2eb726729bf0b4ce9d5426b9c217a0\redhat.java\jdk1.8.0_261\bin" "Narrowingconversion"
The double value: 78.9
After narrowing conversion to int: 78
PS C:\Users\Lenovo\Desktop\java>
```

- 3) Write a program that performs arithmetic operations involving different data types (int, double, float) and observes how Java handles widening conversions automatically



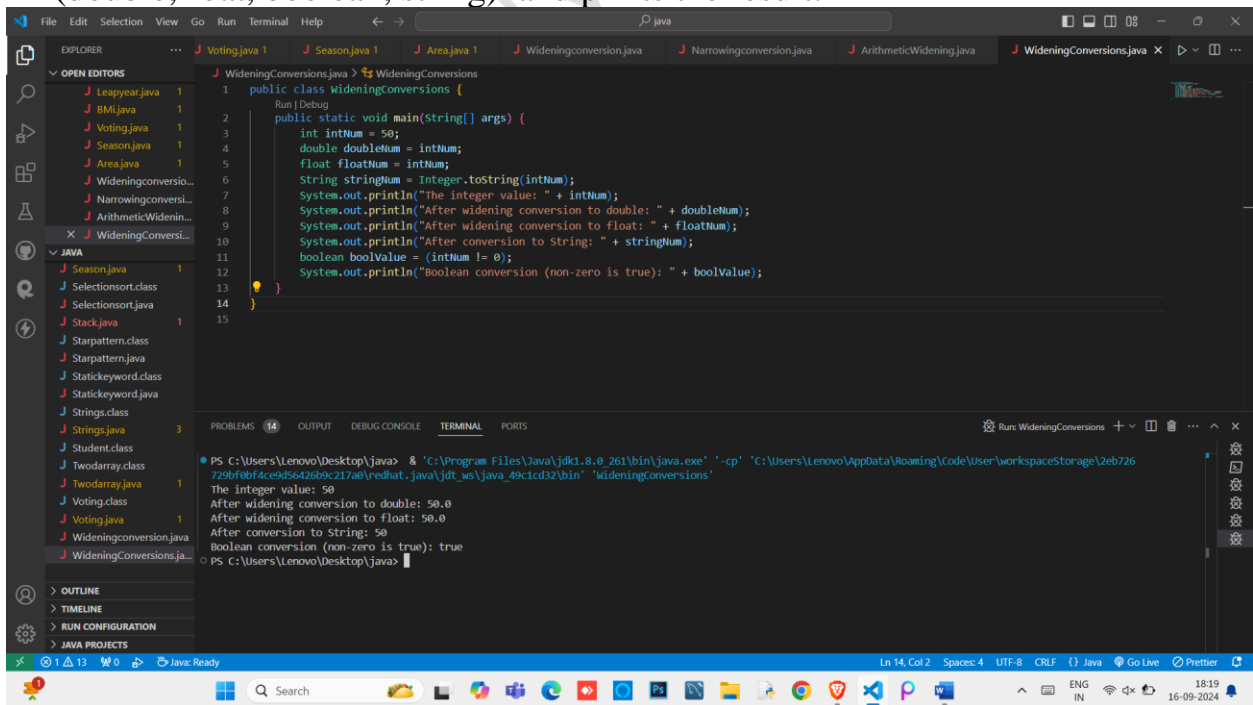
The screenshot shows an IDE with a Java file named `ArithmeticWidening.java`. The code defines a public class `ArithmeticWidening` with a `main` method. Inside the `main` method, several variables are declared and assigned values: `int intNum = 10;`, `double doubleNum = 5.5;`, and `float floatNum = 2.5f;`. Then, arithmetic operations are performed: `double result1 = intNum + doubleNum;`, `float result2 = intNum + floatNum;`, and `double result3 = floatNum + doubleNum;`. Finally, the results are printed using `System.out.println`. The terminal output shows the execution of the program, displaying the results of these operations: `int + double = 15.5`, `int + float = 12.5`, and `float + double = 8.0`.

```
public class ArithmeticWidening {  
    public static void main(String[] args) {  
        int intNum = 10;  
        double doubleNum = 5.5;  
        float floatNum = 2.5f;  
        double result1 = intNum + doubleNum;  
        float result2 = intNum + floatNum;  
        double result3 = floatNum + doubleNum;  
        System.out.println("int + double = " + result1);  
        System.out.println("int + float = " + result2);  
        System.out.println("float + double = " + result3);  
    }  
}
```

Terminal Output:

```
PS C:\Users\Lenovo\Desktop\java> & 'C:\Program Files\Java\jdk1.8.0_261\bin\java.exe' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\2eb726729b9bf9bface956426b9c217a0\redhat.java\jdk_ws\java_49c1cd32\bin' 'ArithmeticWidening'  
int + double = 15.5  
int + float = 12.5  
float + double = 8.0  
PS C:\Users\Lenovo\Desktop\java>
```

- 4) Write a Program that demonstrates widening conversion from int to (double, float, boolean, string) and prints the result.



The screenshot shows an IDE with a Java file named `WideningConversions.java`. The code defines a public class `WideningConversions` with a `main` method. Inside the `main` method, an integer variable `intNum` is declared and assigned the value 50. Then, widening conversions are demonstrated: `double doubleNum = intNum;`, `float floatNum = intNum;`, `String stringNum = Integer.toString(intNum);`, and `boolean boolValue = (intNum != 0);`. Finally, the results are printed using `System.out.println`. The terminal output shows the execution of the program, displaying the results of these conversions: `The integer value: 50`, `After widening conversion to double: 50.0`, `After widening conversion to float: 50.0`, `After conversion to String: 50`, and `Boolean conversion (non-zero is true): true`.

```
public class WideningConversions {  
    public static void main(String[] args) {  
        int intNum = 50;  
        double doubleNum = intNum;  
        float floatNum = intNum;  
        String stringNum = Integer.toString(intNum);  
        System.out.println("The integer value: " + intNum);  
        System.out.println("After widening conversion to double: " + doubleNum);  
        System.out.println("After widening conversion to float: " + floatNum);  
        System.out.println("After conversion to String: " + stringNum);  
        boolean boolValue = (intNum != 0);  
        System.out.println("Boolean conversion (non-zero is true): " + boolValue);  
    }  
}
```

Terminal Output:

```
PS C:\Users\Lenovo\Desktop\java> & 'C:\Program Files\Java\jdk1.8.0_261\bin\java.exe' '-cp' 'C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\2eb726729b9bf9bface956426b9c217a0\redhat.java\jdk_ws\java_49c1cd32\bin' 'WideningConversions'  
The integer value: 50  
After widening conversion to double: 50.0  
After widening conversion to float: 50.0  
After conversion to String: 50  
Boolean conversion (non-zero is true): true  
PS C:\Users\Lenovo\Desktop\java>
```

---

## INTERVIEW QUESTIONS

**Note: Write down this interview question on your notebook ,Take a screenshot & Paste that SS in the word document & upload on your Github.**  
**What does the static keyword mean in Java? Explain the difference between static and non-static methods.**

### **1. What is the role of the static keyword in the context of memory management?**

The static keyword in Java is used to indicate that a member (method or variable) belongs to the class itself rather than to instances of the class. Static variables are stored in a common memory space (method area), which is shared across all instances of the class. This makes them efficient in terms of memory management, as they are initialized only once, regardless of how many objects are created.

### **2. Can static methods be overloaded and overridden in Java? How are static variables shared across multiple instances of a class?**

- **Overloading Static Methods:** Yes, static methods can be overloaded in Java. Overloading means having multiple methods with the same name but different parameter lists within the same class.
- **Overriding Static Methods:** Static methods cannot be overridden because they belong to the class rather than the instance. If a subclass defines a static method with the same signature as a static method in the parent class, it hides the parent class's method, but this is not true method overriding.
- **Static Variables:** Static variables are shared across all instances of a class. They are initialized once, and all objects of that class share the same copy. Changing the value of a static variable in one object reflects across all other objects of that class.

### **3. What is the significance of the final keyword in Java?**

The final keyword in Java can be used in three contexts:

- **Final Variables:** Once a variable is declared as final, it cannot be modified after initialization. This makes the variable a constant.
- **Final Methods:** A method marked as final cannot be overridden by subclasses.

- **Final Classes:** A class marked as final cannot be subclassed. This ensures that the class's behavior remains unchanged in terms of inheritance.

#### **4. What are narrowing and widening conversions in Java?**

- **Widening Conversion:** A widening conversion happens when a smaller data type is automatically converted to a larger data type. For example, converting an int to a double. It occurs implicitly and does not lose information.
- **Narrowing Conversion:** A narrowing conversion happens when a larger data type is explicitly converted to a smaller data type. For example, converting a double to an int. This conversion must be done manually and may lead to a loss of precision.

#### **5. Provide examples of narrowing and widening conversions between primitive data types.**

- **Widening Conversion Example:**

```
int intValue = 10;  
double doubleValue = intValue
```

- **Narrowing Conversion Example:**

```
double doubleValue = 10.99;  
int intValue = (int) doubleValue;
```

#### **6. How does Java handle potential loss of precision during narrowing conversions?**

Java requires explicit casting for narrowing conversions to indicate that the developer is aware of potential data loss. For example, when converting from double to int, the decimal part is truncated, and the precision is lost. Java issues no warnings at runtime but ensures that the programmer explicitly casts the conversion.

Example:

```
double doubleValue = 10.99;  
int intValue = (int) doubleValue
```

#### **7. Explain the concept of automatic widening conversion in Java.**

Automatic widening conversion occurs when Java converts a smaller primitive type to a larger primitive type to prevent data loss. This happens automatically, without explicit casting, because the larger data type can

accommodate the smaller data type without losing precision.

Example:

```
int intValue = 42;
```

```
double doubleValue = intValue;
```

**8. What are the implications of narrowing and widening conversions on type compatibility and data loss?**

- **Widening Conversion:** It is type-safe, and there is no risk of data loss since the larger data type can hold all possible values of the smaller type. It occurs automatically, so there's no need for explicit casting.
- 1. **Narrowing Conversion:** It may result in a loss of precision or truncation of values. Because it is not type-safe, explicit casting is required, and developers must be cautious of potential data loss, particularly when converting between types like double to int or long to int.