

Assignment No- 4

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

```
public class WideningConversion {  
    public static void main(String[] args) {  
        int num = 42;  
  
        double convertedNum = num;  
  
        System.out.println("Original int value: " + num);  
        System.out.println("Widened to double: " + convertedNum);  
    }  
}
```

Main.java	Output
<pre>1- public class WideningConversion { 2- public static void main(String[] args) { 3- int num = 42; 4- 5- double convertedNum = num; 6- 7- System.out.println("Original int value: " + num); 8- System.out.println("Widened to double: " + convertedNum); 9- } 10 } 11</pre>	<pre>java -cp /tmp/JhbsQFIz6x/WideningConversion Original int value: 42 Widened to double: 42.0 === Code Execution Successful ===</pre>

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

```
public class NarrowingConversion {  
    public static void main(String[] args) {  
        double num = 42.75;  
        int convertedNum = (int) num;  
        System.out.println("Original double value: " + num);  
        System.out.println("Narrowed to int: " + convertedNum);  
    }  
}
```

Main.java	Run	Output
<pre>1 public class NarrowingConversion { 2 public static void main(String[] args) { 3 double num = 42.75; 4 int convertedNum = (int) num; 5 System.out.println("Original double value: " + num); 6 System.out.println("Narrowed to int: " + convertedNum); 7 } 8 } 9 } 10</pre>	<div>Run</div>	<pre>java -cp /tmp/11Q9XwwQkU/NarrowingConversion Original double value: 42.75 Narrowed to int: 42 === Code Execution Successful ===</pre> <div>Clear</div>

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

```
public class ArithmeticOperations{  
    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 = doubleNum + floatNum;  
  
        System.out.println("int + double = " + result1);  
        System.out.println("int + float = " + result2);  
        System.out.println("double + float = " + result3);  
    }  
}
```

Main.java	Output
<pre>1- public class ArithmeticOperations { 2- public static void main(String[] args) { 3- int intNum = 10; 4- double doubleNum = 5.5; 5- float floatNum = 2.5f; 6- 7- double result1 = intNum + doubleNum; 8- float result2 = intNum + floatNum; 9- double result3 = doubleNum + floatNum; 10- System.out.println("int + double = " + result1); 11- System.out.println("int + float = " + result2); 12- System.out.println("double + float = " + result3); 13- } 14- } 15-</pre>	<pre>java -cp /tmp/BFMIkwZT7/ArithmeticOperations int + double = 15.5 int + float = 12.5 double + float = 8.0 === Code Execution Successful ===</pre>


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

```
public class WideningConversionDemo {

    public static void main(String[] args) {

        int num = 42;
        double doubleNum = num;
        float floatNum = num;
        String stringNum = Integer.toString(num);
        boolean booleanNum = (num != 0);

        System.out.println("Original int value: " + num);
        System.out.println("Widened to double: " + doubleNum);
        System.out.println("Widened to float: " + floatNum);
        System.out.println("Converted to String: " + stringNum);
        System.out.println("Logically converted to boolean: " +
booleanNum);
    }
}
```

Main.java	Run	Output
<pre>1- /** 2- * WideningConversionDemo 3- */ 4- public class WideningConversionDemo { 5- 6- public static void main(String[] args) { 7- int num = 42; 8- double doubleNum = num; 9- float floatNum = num; 10- String stringNum = Integer.toString(num); 11- boolean booleanNum = (num != 0); 12- 13- System.out.println("Original int value: " + num); 14- System.out.println("Widened to double: " + doubleNum); 15- System.out.println("Widened to float: " + floatNum); 16- System.out.println("Converted to String: " + stringNum); 17- System.out.println("Logically converted to boolean: " + booleanNum); 18- } 19- }</pre>		<pre>java -cp /tmp/zPx08lAX6V/WideningConversionDemo Original int value: 42 Widened to double: 42.0 Widened to float: 42.0 Converted to String: 42 Logically converted to boolean: true === Code Execution Successful ===</pre>

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.
2. Can static methods be overloaded and overridden in Java?
How static variables shared across multiple instances of a class?
3. What is the significance of the final keyword in Java?
4. What are narrowing and widening conversions in Java?
5. Provide examples of narrowing and widening conversions between primitive data types.
6. How does Java handle potential loss of precision during narrowing conversions?
7. Explain the concept of automatic widening conversion in Java.
8. What are the implications of narrowing and widening conversions on type compatibility and data loss?

Assignment No. 4.

Q.1] What does the static keyword in java mean?
Explain the difference between static & non-static methods?

→ In Java, the static keyword is used to indicate that a member (field or method) belongs to the class itself, rather than instances of the class. This means that the static member is shared among all instances of the class. Static members can be accessed without creating an instance of the class.

Difference between static & non-static method.

Aspect	Static Method.	Non-Static Method.
Belongs to	Class	Instance of the class.
Accessing.	Can be called using the class name e.g, <code>ClassName.method()</code>	Can only be called on an object instance. e.g, <code>object.method()</code>
Memory Allocation	Memory is allocated once, shared across all objects.	Memory is allocated every time a new object is created.
Access to Instance Variable	Cannot directly access instance variable.	Can access both static and instance variable.
Access to Static Variable	Can access static variable directly.	Can access static variable directly.
Method Overriding.	Cannot be overridden (but can be hidden in subclass).	Can be overridden by subclasses.
Use Case.	Used for utility or helper method.	Used when behaviour is dependent on object state.
Example	<code>public static void show().</code>	<code>public void show().</code>

Q1]. What is the role of the static keyword in the context of memory management?

→ The static keyword allows a field or method to belong to the class rather than instances of the class. Static fields are stored in the method area of JVM memory, and only one copy is shared among all instances. This saves memory because it avoids creating multiple copies of the variable for each object. Static methods, also shared, can be accessed without creating an instance of the class, further optimizing memory usage.

Q2] Can static methods be overloaded and overridden in Java?
How static variables shared across multiple instances of a class?

- a) • **Overloading:** Yes, static method can be overloaded.
Multiple static methods can have same name but different parameter lists within the same class.
- **Overriding:** Static method cannot be overridden in the same way instance methods are. Instead, they are hidden in the subclass if a static method with the same signature is declared.

b) Static Variables are shared across all instances of a class. Each instance access the same memory location for a static variable. Since memory for static variable is allocated only once when the class is loaded, all instances read and write to the same variable, maintaining consistency across the class.

Q3]. What is the significance of final keyword in Java?

→ The final keyword in Java is used to define constants, prevent method overriding, & prevent inheritance. It can be applied to:-

- Variables: Makes the variable a constant that can be assigned only once.
- Methods: Prevent a method from being overridden by subclass.
- Classes: Prevents a class from being subclassed.

Q4]. What are narrowing and widening conversions in Java?

-
- Widening Conversion: When a smaller data type is automatically converted to a larger one. eg. int to long.
 - Narrowing Conversion: When a larger data type is explicitly converted to a smaller one. eg. long to int.

Q5]. Provide examples of narrowing and widening conversions between primitive data types.

→ • Widening:

```
int i = 100;  
long l = i; // Automatic widening from int to long.
```

• Narrowing:

```
double d = 123.45;  
int i = (int) d; // Explicit narrowing from double to int.
```

Q6]. How does java handle potential loss of precision during narrowing?

→ Java requires explicit casting for narrowing conversions to alert the programmer of potential precision loss.

```
double d = 123.45;  
int i = (int) d; // The fractional part (.45) is lost.
```

In this case, the conversion discards the decimal portion, & only the whole number part is retained.

Q7]

Explain the concept of automatic widening conversion in Java?

→ Java automatically handles widening conversions, where a smaller type is promoted to a larger type without requiring explicit casting.

```
int i = 42;  
double d = i; // int is automatically widened to double.
```

Q8]

What are the implications of narrowing and widening conversions on type compatibility & data loss?

→ • Widening: Safe, no data loss, automatic conversion by the compiler.

• Narrowing: Requires explicit casting because it can lead to data-loss. Narrowing conversions can cause issue with type compatibility & precision, while widening conversions are generally safe and automatically handled by Java.