Assignment 4

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

```
package com.conversions;

public class WideningConversionDemo {

public static void main(String[] args) {
int intValue = 25;
double doubleValue = intValue; // Widening conversion from int to
double

System.out.println("Widening Conversion from int to double:");
System.out.println("Int value: " + intValue);
System.out.println("Double value: " + doubleValue);
}
}
```

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

```
package com.conversions;

public class NarrowingConversionDemo {

public static void main(String[] args) {
double doubleValue = 28.58;
int intValue = (int) doubleValue; // Narrowing conversion from double
to int

System.out.println("Narrowing Conversion from double to int:");
System.out.println("Double value: " + doubleValue);
System.out.println("Int value: " + intValue);
}
```

```
}
```

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

```
package com.conversions;

public class ArithmeticOperationsDemo {

public static void main(String[] args) {
int intValue = 10;
float floatValue = 8.5f;
double doubleValue = 8.25;

double result1 = intValue + floatValue; // int + float -> float
(widened to double)
double result2 = intValue + doubleValue; // int + double -> double

System.out.println("Arithmetic Operations with Different Data
Types:");
System.out.println("Int value: " + intValue);
System.out.println("Float value: " + floatValue);
System.out.println("Double value: " + doubleValue);
System.out.println("Result of int + float: " + result1);
System.out.println("Result of int + double: " + result2);
}
}
```

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

```
package com.conversions;

public class WideningConversionExtendedDemo {
```

```java
public static void main(String[] args) {
int intValue = 180;

// Widening conversion to double
double doubleValue = intValue;

// Widening conversion to float
float floatValue = intValue;

// Conversion to String
String stringValue = Integer.toString(intValue);

System.out.println("Widening Conversion from int to various types:");
System.out.println("Int value: " + intValue);
System.out.println("Double value: " + doubleValue);
System.out.println("Float value: " + floatValue);
System.out.println("String value: " + stringValue);
// System.out.println("Boolean value: " + booleanValue);
}
}
```

INTERVIEW QUESTIONS

Note: Write down this interview question on your notebook, Take a screenshort & 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?Howstatic 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?

A1. Role of the static keyword in memory management:

- Single Memory Allocation:
  Static variables & methods are allocated memory once per class rather than per instance, meaning they are shared across all instances of that class.
- Class Level Storage:
  static members are stored in method area of the java memory, which is common to all instances of the class
- Lifetime:
  Static members are initialized once & exist for the duration of the program, which can improve memory efficiency & performance for shared data.
- Access without Instance:
  static methods & variables can be accessed directly using the class name without needing to create an instance of the class.
- Shared State: Changes to static variables are reflected across all instances of class, as there is only one copy of the variable in memory.

A2. Static Methods:

- Overloading: It can be overloaded. We can have multiple static methods in the

same class with the same name but different parameter lists.

- Overriding: Static methods cannot be overridden. They belong to the class, not instances, & are resolved at compile-time. They can be hidden in subclasses if declared with the same name, but this is not overriding.
- ~~Shared~~ Static Variables:
  Shared Across instances among all instances of a class. All instances access the same memory location for that variable.
- Class level storage - changes to a static variable is reflected across all instances because there is only one copy of the variable in memory for entire class.

A3. Significance of the final keyword in java is:

- A final variable can be assigned only once & cannot be modified afterward, making it a constant.
- A final method cannot be overridden by subclasses, ensuring that the method's behaviour remains unchanged in inheritance hierarchies.
- A final class cannot be subclassed, preventing any further inheritance, which helps in securing & protecting its implementation.

A4. Widening Conversion: It automatically converting a smaller data type to a larger one, which is safe and no data is lost.

- Eg. int to long or float to double
- Widening happens implicitly.
- Narrowing Conversion: It explicitly converting a larger data types to a smaller one, which might result in data loss.
- Eg. double to int or long to byte
- Narrowing requires explicit casting.

A5. Examples of narrowing & widening conversions between primitive data types in java:

- Widening Conversions (Implicit):

1. From int to long:

```
int num = 100;
long bigNum = num;
```

2. From float to double:

```
float pi = 3.14f;
double preciePi = pi;
```

3. From char to int;
char letter = 'A';
int asciiValue = letter;
Narrowing Conversion (Explicit)
1. From double to int:
double decimal = 9.78;
int wholeNumber = (int) decimal;
2. From long to short:
long bigValue = 1000L;
short smallValue = (short) bigValue;
3. From int to byte:
int largeNum = 138;
byte smallNum = (byte) largeNum;

A6. Java handles potential loss of precision during narrowing conversions as follows:
1. Explicit Casting Required:
Java forces developers to use explicit casting for narrowing conversions to signal that they r aware of potential data loss.
2. Rounding:
When narrowing a floating-point number to an integer, the fractional part is truncated, not rounded, leading to loss of precision.
3. Overflow/Underflow:
When narrowing large values, if the value exceeds the target type's range, overflow or underflow can occur, resulting in unexpected ~~or~~ values.

A7. Implicit Conversion: Java automatically converts a smaller or less data type to a larger or more precise one without required explicit casting
- No data loss: Widening conversions are safe because the larger data type can fully accommodate the value of the smaller type, ensuring no data loss
- Happens at Compile Time: At compile time & r managed by java compiler, making it seamless & error-free in normal usage

A8. Widening is safe & does not result in data corruption, while narrowing can cause unexpected behaviours if not carefully managed.
- Widening ensures compatibility in mixed type expressions, while narrowing conversion need caution due to possible type mismatch & need for explicit casting

- Narrowing requires explicit casting, there's a risk of data loss. The smaller type might not fully store the larger type's value, leading to potential overflow.
- In widening, no data loss occurs bcz the larger type can fully accomodate the smaller type's value.