Type Conversion

Data Type Compatibility

A variable will not be able to store a value if the value is more than the capacity of the datatype of the variable.
Whenever any operation is performed containing different data types, the data type of result is the largest data type in the expression.

For example, the result of (intVar + floatVar) will be a float value, the result of (intVar + longVar + doubleVar) will be a double value.

When you assign the value of one data type to another or when you perform an operation on two operands, their data types must be compatible with each other. If the data types are not compatible, then the data type of an operand needs to be converted.

This conversion is of two types:

1. Implicit
2. Explicit

Implicit Type Conversion

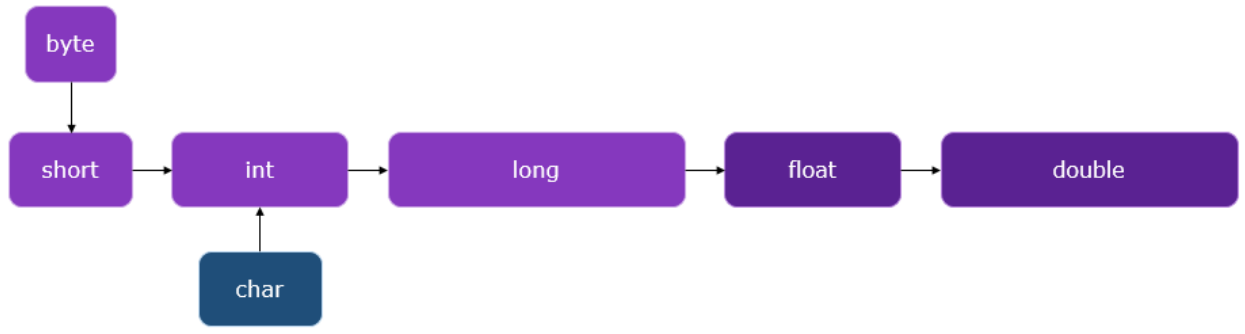
I**mplicit Type Conversion** is also known as **Widening** conversion. It happens in the below scenarios:

1. When a value of a data type with smaller range is assigned to a variable of a compatible data type with larger range.
2. When two variables of different data types are involved in an expression, the value of smaller range datatype is converted to a value of larger range datatype and then the operation is performed.

Eg.
int discountPercentage = 10;
double newDiscountPercentage = discountPercentage;

Here, when we assign the value of discountPercentage to newDiscountPercentage, automatically discountPercentage is converted to double type. It is done automatically by the compiler; hence it is called as Implicit type conversion.

In Java, implicit type conversion happens as depicted in the diagram below. For example, if a short variable is assigned to int, long, float or double variable, it will get converted implicitly.



Explicit Type Conversion

Explicit Conversion is used when you want to assign a value of larger range data type to a smaller range data type. This conversion is not done by the compiler implicitly as there can be loss of data in some cases. Hence, programmers have to be cautious about such conversions. This is also known as **Narrowing** conversion.

```
double totalPrice = 200;
int newPrice = totalPrice;
```

This will lead to an error because a value of larger range data type, in this case double, cannot be directly assigned to a smaller range data type, in this case int. Here, you need to explicitly typecast double to int as shown below.

```
int newPrice = (int)totalPrice;
```

Eg.

```
public static void main(String[] args) {
   int discountPercentage = 10;
   double totalPrice = 200;
   double priceAfterDiscount = totalPrice * (1 - (discountPercentage / 100));
   System.out.println("Customer has paid a bill of amount: "+ priceAfterDiscount);
}
```

}

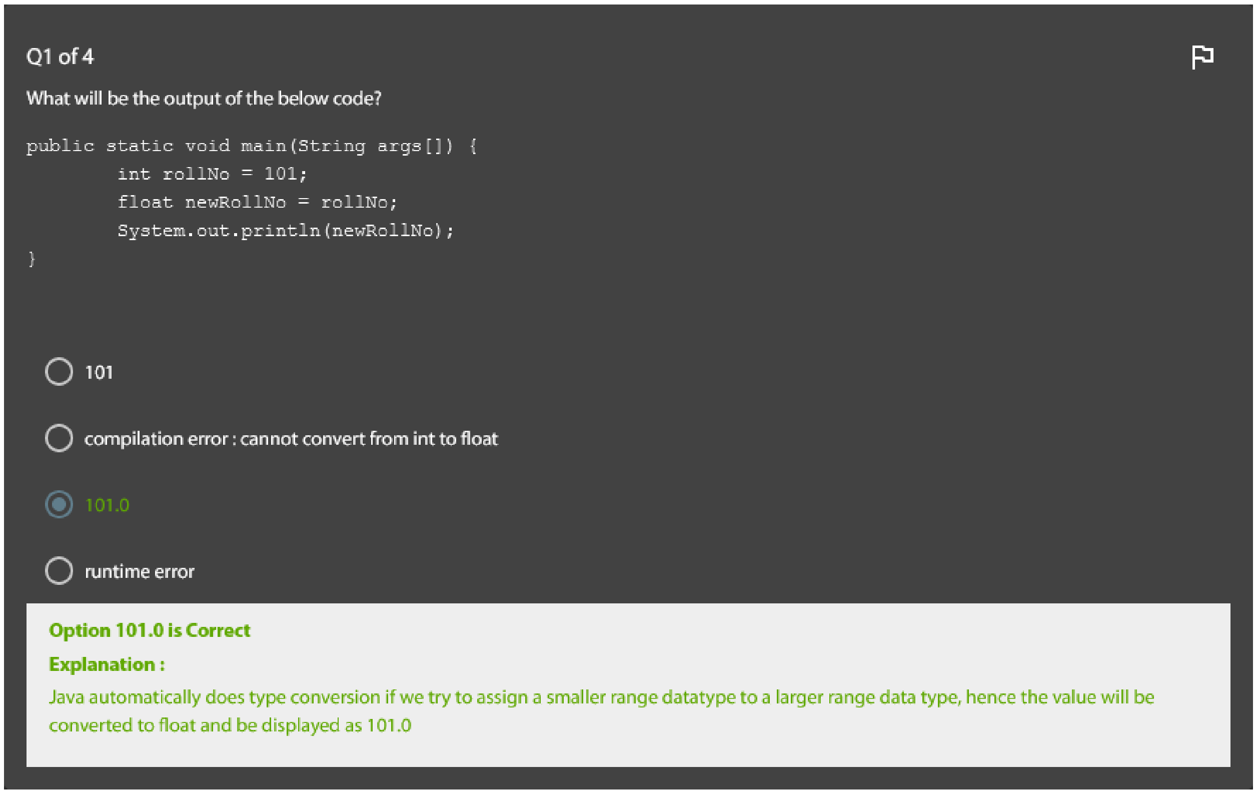
Here, the result of (discountPercentage / 100) will be integer and the fractional part will be lost. This happens because both the operands, discountPercentage and 100, are integers. To get the result with fractional part, one of the operands must be converted to float or double as below.

```
double priceAfterDiscount = totalPrice * (1 - ((double)discountPercentage / 100));
```

Tryout Code

```
class Tester {

   public static void main(String srgs[]) {
         int variableOne = 10;
         float variableTwo = variableOne;
         System.out.println(variableOne);
         System.out.println(variableTwo);
         // Here, a variable of data type having smaller range is automatically
         // accommodated in a variable of data type having bigger range. This is
         // known as widening.

         // When we try to accommodate a variable of data type having bigger
         // range into a data type having smaller range, it leads to an error.
         float variableThree = 12.5f;
         //int variableFour = variableThree; // comment this line or fix the error
                                                                    // to execute the
code successfully
         System.out.println(variableThree);
         //System.out.println(variableFour); // comment this line or fix the error
                                                                    // to execute the
code successfully

         // Since our assignment is leading to loss of data, we need to
         // explicitly specify that the data needs to be converted to a data type
         // having smaller range.
         // This is known as explicit typecasting.
         int variableFive = (int) variableThree;
         System.out.println(variableFive);
   }

}
```

Quiz



Q1 of 4

What will be the output of the below code?

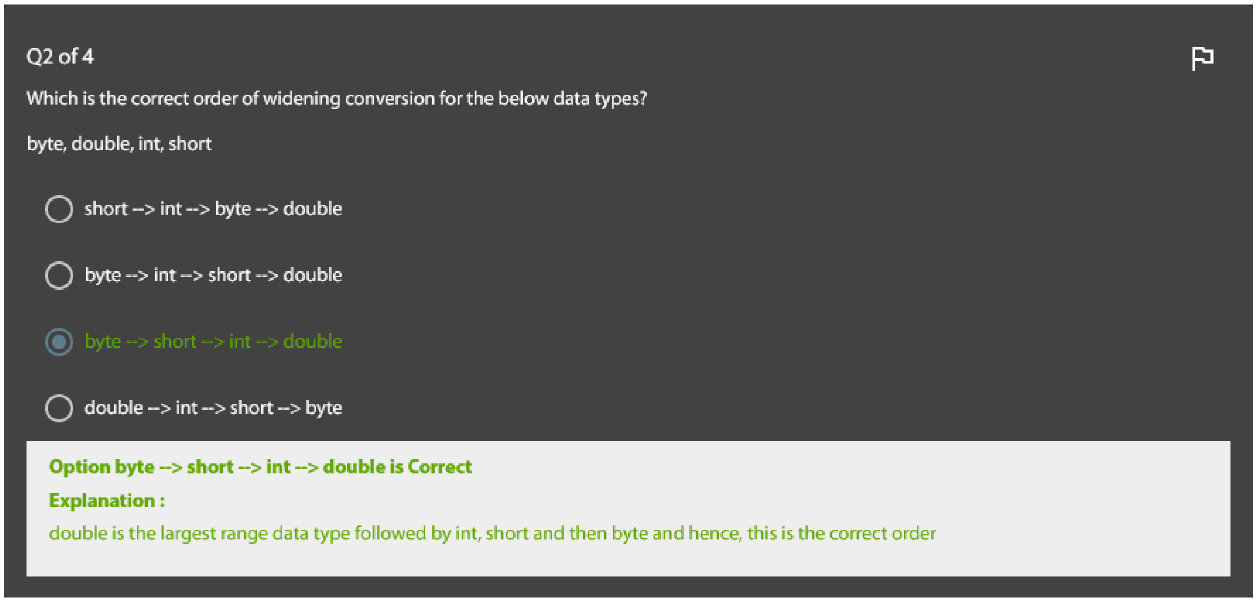
```
public static void main(String args[]) {
        int rollNo = 101;
        float newRollNo = rollNo;
        System.out.println(newRollNo);
}
```

- 101
- compilation error : cannot convert from int to float
- [✓ 101.0]
- runtime error

Option 101.0 is Correct

Explanation :

Java automatically does type conversion if we try to assign a smaller range datatype to a larger range data type, hence the value will be converted to float and be displayed as 101.0



Q2 of 4

Which is the correct order of widening conversion for the below data types?

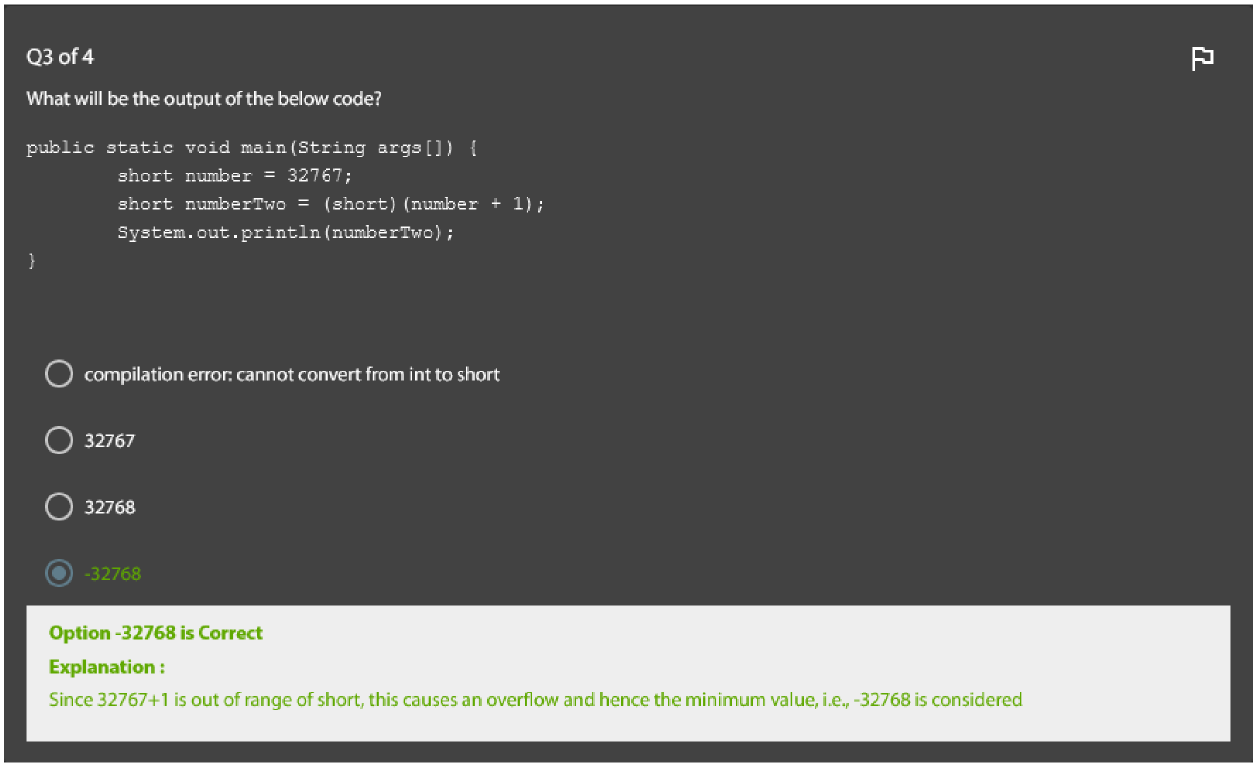
byte, double, int, short

- short --> int --> byte --> double
- byte --> int --> short --> double
- [✓ byte --> short --> int --> double]
- double --> int --> short --> byte

Option byte --> short --> int --> double is Correct

Explanation :

double is the largest range data type followed by int, short and then byte and hence, this is the correct order



Q3 of 4

What will be the output of the below code?

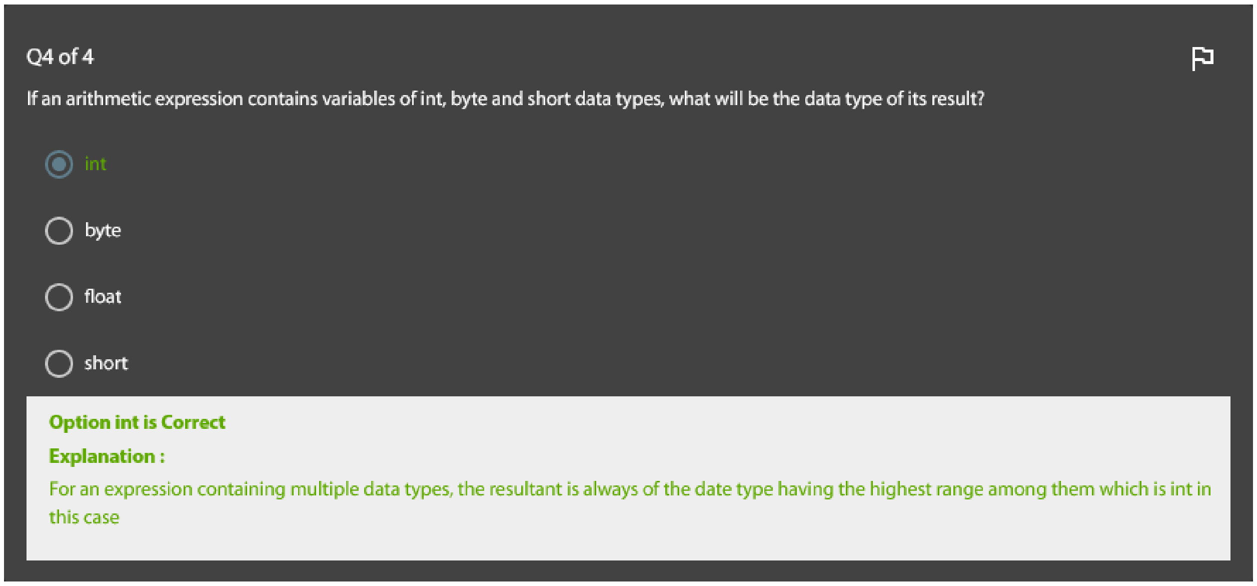
```
public static void main(String args[]) {
        short number = 32767;
        short numberTwo = (short)(number + 1);
        System.out.println(numberTwo);
}
```

- compilation error: cannot convert from int to short
- 32767
- 32768
- [✓ -32768]

Option -32768 is Correct

Explanation :

Since 32767+1 is out of range of short, this causes an overflow and hence the minimum value, i.e., -32768 is considered



Q4 of 4

If an arithmetic expression contains variables of int, byte and short data types, what will be the data type of its result?

- [✓ int]
- byte
- float
- short

Option int is Correct

Explanation :

For an expression containing multiple data types, the resultant is always of the date type having the highest range among them which is int in this case

Assignment 1

Implement a program to find the area of a circle by using the formula given below and display the calculated area.

Area = pi*radius*radius
The value of pi is 3.14.

```
class Tester {
   public static void main(String[] args) {
         // Implement your code here
         float PI = 3.14f;
         int radius = 10;
         float area = PI * radius * radius;
         System.out.println(area);
   }
}
```

Assignment 2

Implement a program to convert temperature from Fahrenheit to Celsius degree by using the formula given below and display the converted value.

C = ((F-32)/9)*5 where, C represents temperature in Celsius and F represents temperature in Fahrenheit.

```
class Tester {
   public static void main(String[] args) {
         // Implement your code here
         int F = 50;
         float C = (((float) F - 32) / 9) * 5;
         System.out.println(C);
   }
}
```