TP04

Java Operators, if…else and switch

Remark

1. Operators:
    a. Arithmetic Operators
    b. Relational Operators
    c. Bitwise Operators
    d. Logical Operators
    e. Assignment Operators
    f. Miscellaneous Operators

1.a. The Arithmetic Operators

Assume that integer variable A = 10 and variable B = 20, then:

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	A + B will give 30
- (Subtraction)	Subtracts right-hand operand from left-hand operand.	A - B will give -10
* (Multiplication)	Multiplies values on either side of the operator.	A * B will give 200
/ (Division)	Divides left-hand operand by right-hand operand.	B / A will give 2
% (Modulus)	Divides left-hand operand by right-hand operand and returns remainder.	B % A will give 0
++ (Increment)	Increases the value of operand by 1.	B++ gives 21
-- (Decrement)	Decreases the value of operand by 1.	B-- gives 19

1.b. The Relational Operators

Assume that integer variable A = 10 and variable B = 20, then:

Operator	Description	Example
== (equal to)	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!= (not equal to)	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
> (greater than)	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
< (less than)	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>= (greater than or equal to)	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<= (less than or equal to)	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

1.c. The Bitwise Operators

Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60 and b = 13; now in binary format they will be as follows −

a = 0011 1100
b = 0000 1101

---

a&b = 0000 1100
a|b = 0011 1101
a^b = 0011 0001
~a = 1100 0011

Assume integer variable A = 60 and variable B = 13 then

Operator	Description	Example
& (bitwise and)	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
| (bitwise or)	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^ (bitwise XOR)	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001

~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.
<< (left shift)	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>> (right shift)	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>> (zero fill right shift)	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 15 which is 0000 1111

1.d. The Logical Operators

Assume Boolean variables A = true and variable B = false, then:

Operator	Description	Example
&& (logical and)	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false
|| (logical or)	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A || B) is true
! (logical not)	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true

1.e. The Assignment Operators

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C – A

*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand.	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator.	C <<= 2 is same as C = C << 2
>>=	Bitwise AND assignment operator.	C >>= 2 is same as C = C >> 2
&=	Right shift AND assignment operator.	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator.	C ^= 2 is same as C = C ^ 2
|=	bitwise inclusive OR and assignment operator.	C |= 2 is same as C = C | 2

1.f. Miscellaneous Operators

- Ternary operator

```
variable x = (expression) ? value if true : value if false
```

Example:

```
public class TernaryTest {
   public static void main(String[] args) {
       int a, b;
       a = 10;
       b = (a == 1) ? 20 : 30;
       System.out.println("Value of b is : " + b);

       b = (a == 10) ? 20 : 30;
       System.out.println("Value of b is : " + b);
   }
}
```

Output:

```
Value of b is : 30
Value of b is : 20
```

- An instanceof operator

```
( Object reference variable ) instanceof  (class/interface type)
```

Example:

```
public class InstanceofTest {
   public static void main(String[] args) {
       String name = "Dara";

       // following will return true since name is type of String
       boolean result = name instanceof String;
       System.out.println(result);
   }
}
```

2. Control structure if…else

Syntaxes:

```
if(expression) action;
```

Or

```
if(expression){actions;}
```

Or

```
if(expression) action;
else action;
```

Or

```
if(expression){
   actions;
}else{
   actions;
}
```

Or

```
if(expression){
   actions;
}else if(expression2){
   actions;
}else if …
```

3. Choices – switch … case

Syntax:

```
switch(expression) {
   case value :
      // Statements
      break; // optional

   case value :
      // Statements
      break; // optional

   // You can have any number of case statements.
   default : // Optional
      // Statements
}
```

TP04.1. Prime Number

Prime number is positive number greater than 2 and divisible only to its own and 1.
Create class PrimeNumber with:

- a constructor that takes 1 parameter
- a method isPrime() that returns true only if the number is a prime number.

Implement an application Java that let user input a number then determine if it is a prime number using PrimeNumber class which is previously defined. Example:

```
Input number to check whether it is prime number: 29
29 is prime number.
```

Example 2:

```
Input number to check whether it is prime number: 57
57 is not prime number, because it is divisible to 3.
```

TP04.2. Lucky Number

Lucky number is a 6 digits number that sum of first 3 digits equals to sum of last 3 digits.

Create class LuckyNumber with:

- a constructor that takes 1 parameter representing lucky number
- a method isLucky() that returns true only if the number is lucky one
- a method isValid() that returns true only if the number is 6-digit number

Using LuckyNumber class to write a program in Java to determine whether a given number is lucky number. The program will display error message in case given number is not 6 digits number.

Example:

```
Program for testing for lucky number.
Please input 6 digits number: 2020

Invalid input number, please input only 6 digits number.
```

Example 2:

```
Program for testing for lucky number.
Please input 6 digits number: 159543

159543 is not lucky number.
```

Example 3:

```
Program for testing for lucky number.
Please input 6 digits number: 167923

167923 is lucky number.
```

TP04.3. Reversing Number

Create class and write a program in Java to read a 4 digits number from keyboard. Then the program will reverse given number, and display it. If user input number other than 4 digits number, display error message. Example:

```
Program for reversing a 4 digits number.
Please input 4 digits number: 50000

Error: invalid number, please input only 4 digits number.
```

Example 2:

```
Program for reversing a 4 digits number.
Please input 4 digits number: 5000

After reverse: 5
```

Example 3:

```
Program for reversing a 4 digits number.
Please input 4 digits number: 1204

After reverse: 4021
```

TP04.4. Money Exchanges

Create classes to support each currency. Write a program in Java to exchange money in Riels to Dollar, Riels to Thai Baht, Dollar to Riels, Dollar to Thai Baht, and Thai Baht to Riels. Suppose that conversion rate is 1$ = 4000៛, and 1$ = 30B.

Program need to display menu for communicating with user. Example:

```
Welcome to program Money Exchanges!
  1. Riels to Dollar
  2. Riels to Thai Baht
  3. Dollar to Riels
  4. Dollar to Thai Baht
  5. Thai Baht to Riels
  6. Exit
Choose an option: 1
Input money in RIELS: 2200

2200 RIELS = 0.55 USD
```

Example 2:

```
Welcome to program Money Exchanges!
  1. Riels to Dollar
  2. Riels to Thai Baht
  3. Dollar to Riels
  4. Dollar to Thai Baht
  5. Thai Baht to Riels
  6. Exit
Choose an option: 3
Input money in Dollars: 10

10 USD = 40000 RIELS
```

TP04.5. Max among 8 Numbers

Write a program in Java to read 8 integers from keyboard. Then find maximum number.
Note: You need to think of a very simple method.

TP04.6. Shipping

A shipping ship need to transport goods from point A to point C. This ship capable to store maximum 5000 liters of Petrol. The ship can stop only one time to refill, it is point B. The distance between A to B, between B to C, and weight are given by user. The number of liters of petrol used by the ship depends on the weight of goods loaded. The number of liters used is described a below:

- Up to 5000Kg, uses 10L/Km
- Up to 10000Kg, uses 20L/Km
- Up to 20000Kg, uses 25L/Km
- Up to 30000Kg, uses 35L/Km
- More than 30000Kg, cannot be loaded

Write a program in Java to calculate the minimum number of liters needed to **refill at point B** in order to reach point C.

TP04.7. Leap Year

Write a program in Java to tell whether inputted year is Leap year. Leap year is a year that is divisible by 4 but not divisible by 100, or a year that divisible by 400. In case user input number less than 1, the program will display error message.

_____SECTION BELOW IS BEGINNING OF CHALLENGE EXERCISES (+10% ABOVE THE TOTAL SCORES)_____

TP04.CE.1. TP04 Class

Create a class called TP04 that contains all the methods of the 7 above tasks.

Write a program in Java to display a menu containing 7 exercises in TP04 and 1 exit option. When user select an option, the program will call a selected method of TP04 class. Example: if method of TP04.1. is **primeNumber**, then when user choose option 1, it calls **TP04.primeNumber(number)** as example below:

```
-------- Menu ---------
 1. Prime number
 2. Lucky number
 3. Reversing number
 4. Money exchange
 5. Max among 8 numbers
 6. Shipping
 7. Leap year
 0. Exit
Choose an option: 1
----------------------
Input a number: 29
29 is prime number.
```

The program will continue to ask user questions until user choose option 0 (Exit).