IJSE

Institute of Software Engineering

Graduate Diploma in Software Engineering

Batch - GDSE69

Module - Programming Fundamentals

Assignment

Name : Shimara Appuhami

Nic : 200362310434
/
/

Assignment 03

1. What is the purpose of Java data types, and why are they important in programming?

★ Data types are especially important in Java because it is a strongly typed language. This means that all operations are type-checked by the compiler for type compatibility.The compiler will consider it to be a syntax error if you try to violate this rule. Thus, strong type checking helps prevent errors and enhances reliability.

///
//////////

2. Differentiate between primitive data types and reference data types in Java. Provide examples of each.

★ **Primitive Data Types**

A primitive data type specifies the size and type of variable values, and it has no additional methods.There are eight primitive data types in Java

- Byte
- Short
- Int
- Long
- Float
- Double
- Char
- Boolean

★ **reference data types**

reference data types are called Non-primitive data types.

- Spring
- Arrays
- Interface
- classes

The main difference between **primitive** and **non-primitive** data types are:

- Primitive types are predefined in Java. Non-primitive types are created by the programmer and is not defined by Java (except for `String`).
- Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
- A primitive type has always a value, while non-primitive types can be `null`. ● A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.

//
//////////

3. What are the primitive data types supported by Java and their data ranges?

`byte` 8 byte Stores whole numbers from -128 to 127

`short` 16 bytes Stores whole numbers from -32,768 to 32,767

`int` 32 bytes Stores whole numbers from -2,147,483,648 to 2,147,483,647

`long` 64 bytes Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

`float` 32 bytes Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits `double`

64 bytes Stores fractional numbers. Sufficient for storing 15 decimal digits

`boolean` 1 bit Stores true or false values

`char` 16 bytes Stores a single character/letter or ASCII values

//
//////////

4. Which of the following primitive data types are not considered as a numerical data type?

a. boolean
b. byte
c. float
d. short
e. double
f. char
g. int

//
//////////

5. What is the numerical range of a char?

a. -128 to 127
b. -2^15 to 2^15 – 1
c. 0 to 232
d. 0 to 2^16- 1
//
//////////

6. Both char and short are 16-bit data types in Java. Can you explain how they differ in terms of their representation in memory?

★ Short Data Type

The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.

The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

Example:

short s = 10000, short r = -5000

★ Char Data Type

The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' to '\uffff' (or 65,535 inclusive).The char data type is used to store characters.

Example:

char letterA = 'A'

//
//////////

7. Which of the following do not denote a primitive data value in Java?
a. "p"
b. 'h'
c. 50.5f
d. "Java"
e. true

//
//////////

8. Explain how int data types are represented in computer memory. Include information about the number of bits used and the range of values they can hold.

★ Integers are whole numbers or fixed-point numbers with the radix point fixed after the least-significant bit. They are contrast to real numbers or floating-point numbers, where the position of the radix point varies. It is important to take note that integers and floating-point numbers are treated differently in computers. They have different representation and are processed differently. Floating-point numbers will be discussed later.

★ Computers use a fixed number of bits to represent an integer. The commonly-used bit-lengths for integers are 8-bit, 16-bit, 32-bit or 64-bit. Besides bit-lengths, there are two representation schemes for integers:

1. Unsigned Integers: can represent zero and positive integers.
2. Signed Integers: can represent zero, positive and negative integers. Three representation schemes had been proposed for signed integers:

    a.Sign-Magnitude representation

    b.1's Complement representation

    c.2's Complement representation

You, as the programmer, need to decide on the bit-length and representation scheme for your integers, depending on your application's requirements. Suppose that you need a counter for counting a small quantity from 0 up to 200, you might choose the 8-bit unsigned integer scheme as there is no negative numbers involved.

//
//////////

9. How many different things can be represented using n bytes when each bit can be either 0 or 1?

★ n bits can represent 2^n different values. However, they don't have to be integers ranging from 0 to 2^n-1. They could represent signed integers, or multiples of 10 or whatever set of numbers you want. Just remember that you can only represent 256 different numbers within that set.

//
//////////

10. Help Mr. David to do the following using appropriate data type:
a. Declare a variable to store the first letter of David's name
b. Declare a variable to store count of his children.
c. Declare a variable to store his weight nearest two decimal points
d. Declare a variable to store the name of his spouse.
e. Declare a variable to indicate whether he is married or not

- ★ class Main {
- ★ public static void main(String args[]){
- ★ String first_letter_of_name ="D";
- ★ int count_of_children=2;
- ★ double weight=65.00;
- ★ String name_of_spouse="Shani";
- ★ String whether="Married";
- ★ System.out.println(first_letter_of_name+"\n"+count_of_children+"\n"+weight+"\n"+name_of_spouse+"\n"+whether);
- ★ }
- ★ }

//
//////////

11. Imagine a conversation between a computer science teacher and a curious student who uses IDE for coding. The student notices that IDE assigns different colours to keywords like String compared to int, double, and boolean. The student wonders why String is highlighted differently since all these words are data type names. How would you answer the student's question?

- ★ int, double, and boolean are primitive data types. They are fundamental data types built into the language, and they are used to represent simple values like integers, floating-point numbers, and true/false values.
- ★ String, on the other hand, is not a primitive data type; it's a class in many programming languages, including Java. It represents a sequence of characters, and it's used to work with text and manipulate strings of characters.

Because String is a class and not a primitive data type, the IDE highlights it differently to distinguish it from the basic data types like int, double, and boolean. This distinction helps programmers understand the different roles these keywords play in their code and can also provide useful context for coding.

the IDE uses different colours to help you quickly identify and differentiate between fundamental data types and more complex data types or classes like String. It's all about making your code more readable and easier to understand.

///
//////////

12. What is type casting in Java, and why is it necessary? Provide an example of both implicit and explicit type casting.

★ In Java, type casting is the process of converting one data type into another. Type casting can be categorised into two types: implicit (automatic) type casting and explicit (manual) type casting.

1. **Implicit Type Casting (Widening Conversion):** In implicit type casting, the conversion is done automatically by the compiler when there is no loss of data during the conversion. It involves converting a data type with a smaller range to a data type with a larger range. For example, converting an `int` to a `double` is an implicit type cast.

   Example of implicit type casting:

   int intValue = 5;

   double doubleValue = intValue;
2. **Explicit Type Casting (Narrowing Conversion):** In explicit type casting, the conversion is performed manually by the programmer. It's necessary when converting a data type with a larger range to a data type with a smaller range, and there may be a potential loss of data. Explicit casting is done by specifying the target data type in parentheses before the value or variable to be cast.

   Example of explicit type casting:

   double doubleValue = 5.7;

   int intValue = (int) doubleValue;

///
//////////

13. Explain the difference between widening and narrowing type conversions in Java. When might you use each type of conversion?

★ **Widening Casting (smaller to larger type)**

★ Widening Type Conversion can happen if both types are compatible and the target type is larger than source type. Widening Casting takes place when two types are compatible and the target type is larger than the source type.

★ **Narrowing Casting (larger to smaller type)**

★ When we are assigning a larger type to a smaller type, **Explicit Casting** is required.

///
//////////

14. Define the term "keyword" in Java. Provide examples of Java keywords and explain their significance in the language.
In Java, a "keyword" is a reserved word that has a predefined meaning and purpose in the Java programming language. Keywords cannot be used as identifiers (e.g., variable names, class names, method names) because they have special significance and are reserved for specific language constructs. They are an essential part of the language's syntax and semantics.

Here are some examples of Java keywords and their significance:

`public`, `private`, `protected`, `package-private`:

- **Significance:** These keywords are used to specify the access level or visibility of classes, methods, and fields in Java. They control which parts of the code can access and use these elements.

`class`, `interface`, `enum`:

- **Significance:** These keywords are used to define and declare classes, interfaces, and enumerated types (enums) in Java. They are fundamental for creating different types of Java entities.

`static`, `final`:

- **Significance:** The `static` keyword is used to define class-level members (methods and fields) that belong to the class, not to instances of the class. The `final` keyword is used to declare constants or to make a member or class immutable.

//

15. What are identifiers in Java?

- ★ In Java, identifiers are names given to various program elements, such as variables, classes, methods, interfaces, and more. Identifiers serve as labels or symbols for these program elements and are used to uniquely identify and reference them within your code. Identifiers are an essential part of the Java programming language, and there are specific rules and conventions for naming them
- ★ examples>>
  Public
  Void
  Enum
  Class
  new

//

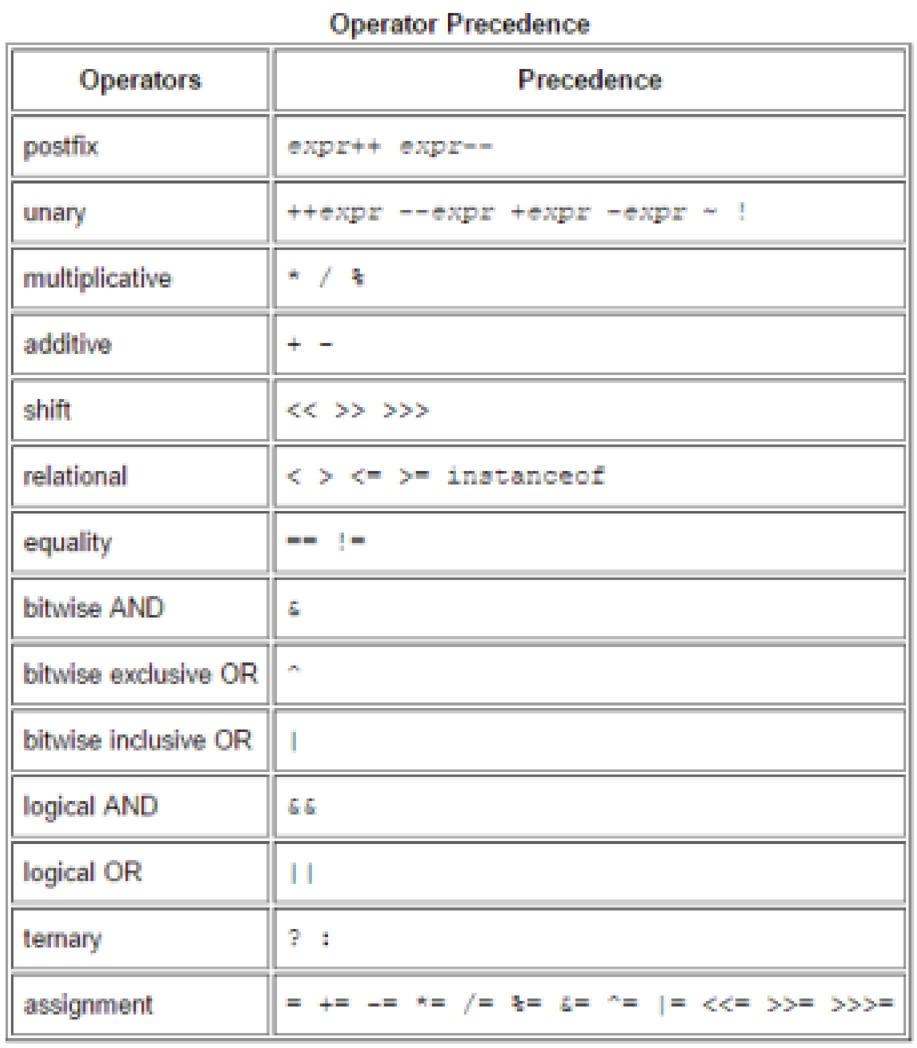
16. What is a "final" variable in Java, and how does it differ from regular variables?

- ★ final keyword is used in different contexts. First of all, final is a non-access modifier applicable only to a variable, a method, or a class. The following are different contexts where final is used.
- ★ Immutable Value:
  - ○ A final variable's value is immutable, meaning it cannot be modified or reassigned once it has been initialized. Regular variables can have their values changed at any time.
- ★ Initialization:

  - Final variables must be initialized at the point of declaration or within the constructor of the class where they are defined. Regular variables can be initialized later.
- ★ Constant Values:
  - Final variables are often used to represent constants, as they hold values that should not change during the program's execution. This is commonly seen with constants like mathematical constants or configuration settings.
- ★ Thread Safety:
  - Final variables are useful for ensuring thread safety in multi-threaded applications. Because the value of a final variable cannot change, it is safe to read from multiple threads without synchronization concerns.
- ★ Compiler Checks:
  - The Java compiler enforces that final variables must be initialized, and it ensures that their values are not changed, making it easier to catch certain types of coding errors.

//
//////////

17. Discuss the concept of operator precedence and associativity rules in Java. Provide a list of common operators and their relative precedence and associativity.



Operator Precedence

Operators	Precedence
postfix	`expr++ expr--`
unary	`++expr --expr +expr -expr ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
relational	`< > <= >= instanceof`
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

//

18. What are unary, binary, and ternary operators, and how are they distinguished in Programming?

★ **Unary Operators:**
  - ○ Unary operators are operators that work with a single operand (a single input).
  - ○ They perform operations on a single value or variable.
  - ○ Unary operators can be prefix (e.g., ++i) or postfix (e.g., i++) operators.
  - ○ Common unary operators in Java include:
    - ■ Increment (++) and Decrement (--)
    - ■ Plus (+) and Minus (-)
    - ■ Logical NOT (!)
    - ■ Bitwise NOT (~)
    - ■ Type Cast ((type))

○ Example:

★ int x = 5;

★ x++;

○

★ **Binary Operators:**

○ Binary operators are operators that work with two operands (two inputs).

○ They perform operations that involve two values or variables.

○ Most mathematical and logical operations are binary.

○ Common binary operators in Java include:

■ Addition (+) and Subtraction (-)

■ Multiplication (*) and Division (/)

■ Modulus (%)

■ Logical AND (&&) and Logical OR (||)

■ Equality (==) and Inequality (!=)

○ Example:

★ int a = 10;

★ int b = 20;

★ int sum = a + b;

○

★ **Ternary Operator:**

○ The ternary operator is a unique operator in that it works with three operands (three inputs).

○ It's used for conditional expressions, making it a shorthand for an if-else statement.

○ The ternary operator in Java is written as condition ? trueExpression : falseExpression, where condition is a Boolean expression, and trueExpression and falseExpression are the values to be returned based on the condition. ○ Example:

java

★ int x = 10;

★ int y = 20;

★ int result = (x > y) ? x : y;

//

19. What are the differences between the ' && ' and ' & ' operators in Java?

Logical AND ('&&') Operator:

● The '&&' operator is used for logical conjunction (logical AND).

● It is primarily used for evaluating conditional expressions in control flow statements like if and while.

- Short-circuiting: The '&&' operator employs short-circuiting, which means it evaluates the second operand only if the first operand is true. If the first operand is false, the second operand is not evaluated.
- Result type: The result of the '&&' operator is a boolean value (true or false).

Example:

★ boolean condition1 = true;
★ boolean condition2 = false;
★ boolean result = condition1 && condition2; // result is false

Bitwise AND ('&') Operator:

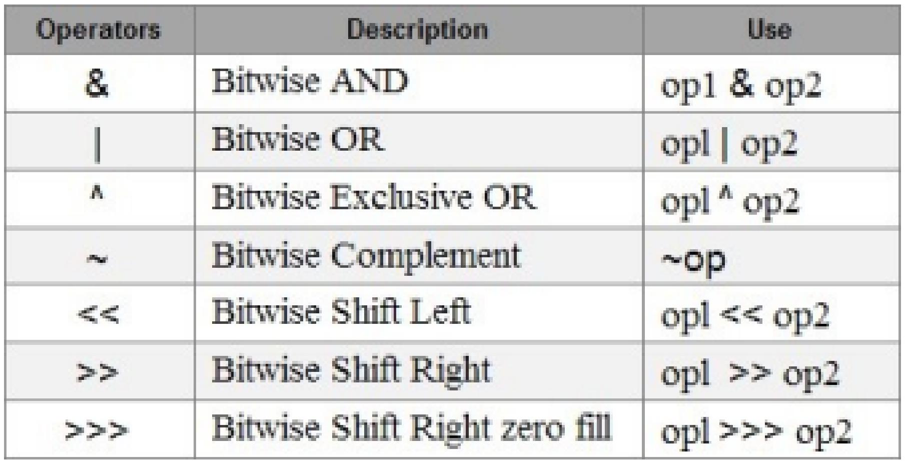
- The '&' operator is used for bitwise operations on individual bits of integer types (int, long, etc.).
- It performs bitwise AND between corresponding bits of its operands.
- Bitwise operators work on integer values by converting the operands to binary and applying the operation to each pair of corresponding bits.
- There is no short-circuiting with the '&' operator; it evaluates both operands regardless of the value of the first operand.
- Result type: The result of the '&' operator is the integer result of the bitwise AND operation.

Example:

★ int value1 = 6;
★ int value2 = 3;
int result = value1 & value2;

//

20. What are Java's bitwise and bit shift operators?



Operators	Description	Use
&	Bitwise AND	op1 & op2
|	Bitwise OR	opl | op2
^	Bitwise Exclusive OR	opl ^ op2
~	Bitwise Complement	~op
<<	Bitwise Shift Left	opl << op2
>>	Bitwise Shift Right	opl >> op2
>>>	Bitwise Shift Right zero fill	opl >>> op2

★ These bitwise and bit shift operators are essential for performing low-level bit manipulation tasks, such as working with binary data, optimizing algorithms, and implementing specific data encodings.

///
//////////

21. What is ternary operator in Java? Give an example.

★ In Java, the ternary operator is a shorthand conditional operator that provides a concise way to express conditional statements. It is also known as the conditional operator because it takes three operands and is used for making decisions based on a condition. Example:

★ int x = 10;
★ int y = 20;

★ int max = (x > y) ? x : y;

///
//////////

22. What is the output when the following code fragment is executed?
int n, k = 5;
n = (100 % k == 0 ? k + 1 : k - 1);
System.out.println("n = " + n + " k = " + k);

★ n = 6 k = 5

///
//////////
23. What is the output when the following code fragment is executed?

```
int i = 5, j = 6, k = 7, n = 3;
System.out.println( i + j * k - k % n );
System.out.println( i / n );
```

- ★ 46
- ★ 1

///
//////////

24. Which of the following statements are legal? And explain your answer.

a. byte b1=100;
- ★ Legal
- ★ 100 is within the range of values that can be assigned to a byte (-128 to 127).

b. byte b2=128;
- ★ Illegal
- ★ 128 is outside the range of values that can be assigned to a byte (-128 to 127).

c. byte b3=-128;
- ★ Legal
- ★ -128 is within the range of values that can be assigned to a byte (-128 to 127).

d. byte b4=0;
- ★ Legal
- ★ 0 is within the range of values that can be assigned to a byte (-128 to 127).

e. short s1=100;
- ★ Legal
- ★ 100 is within the range of values that can be assigned to a `short` (-32,768 to 32,767)

f. short s2=32768;
- ★ Illegal
- ★ 32,768 is outside the range of values that can be assigned to a short. The range for short is -32,768 to 32,767.

g. short s3=32767;
- ★ Legal
- ★ 32,767 is within the range of values that can be assigned to a `short`.

h. short s4=-32768;
- ★ Legal
- ★ -32,768 is within the range of values that can be assigned to a `short`.

///

//////////

25. What are the legal statements of the following? Explain your answer.
a. char c1='A';
 contains a sequence of characters or escape sequences enclosed in single quotation mark symbols,
b. char c2='7';
 contains a sequence of characters or escape sequences enclosed in single quotation mark symbols,
c. char c3='AB';
 contains a sequence of characters or escape sequences enclosed in single quotation mark symbols,
d. boolean b1=true;
 boolean b = true; boolean d = false; The boolean literals represent the logical value either true or false
e. boolean b2=False;
 boolean b = true; boolean d = false; The boolean literals represent the logical value either true or false
f. boolean b3=false;
 boolean b = true; boolean d = false; The boolean literals represent the logical value either true or false
g. boolean b4=True;
 boolean b = true; boolean d = false; The boolean literals represent the logical value either true or false
h. boolean b5="false";
 boolean b = true; boolean d = false; The boolean literals represent the logical value either true or false
i. boolean b6=0;
 boolean b = true; boolean d = false; The boolean literals represent the logical value either true or false

//
//////////

26. What is the exact output of the program below?
```
public static void main(String[] args) {
int n = 4, k = 2;
System.out.println(++n );
System.out.println( n );
System.out.println( n++ );
System.out.println( n );
System.out.println( -n );
System.out.println( n );
System.out.println( --n );
System.out.println( n );
System.out.println( n-- );
System.out.println( n );
System.out.println( n + k );
System.out.println( n );
System.out.println( k );
System.out.println("" + n + " " + k );
System.out.println( n );
System.out.println( " " + n );
System.out.println( " n" );
System.out.println( "\n" );
System.out.println( " n * n = ");
System.out.println( n * n );
System.out.println( 'n' );
}
```

- ★ 5
- ★ 5
- ★ 5
- ★ 6
- ★ -6
- ★ 6
- ★ 5
- ★ 5
- ★ 5
- ★ 4
- ★ 6
- ★ 4
- ★ 2
- ★ 4 2
- ★ 4
- ★ 4
- ★ n
- ★ n * n =
- ★ 16
- ★ n

//

//////////
27. What is the output of the program below?

```
public static void main(String[] args) {
int n;
System.out.println( (n = 4) );
System.out.println( (n == 4) );
System.out.println( (n > 3) );
System.out.println( (n < 4) );
System.out.println( (n = 0) );
System.out.println( (n == 0) );
System.out.println( (n > 0) );
System.out.println( (n == 0 && true) );
System.out.println( (n == 0 || true) );
System.out.println( (!(n == 2) ));
}
```

- ★ 4
- ★ true
- ★ true
- ★ false
- ★ 0
- ★ true
- ★ false
- ★ true
- ★ true
- ★ true

//
//////////

28. Convert following integer numbers into binary, octal and hexadecimal forms:

a. 10
- ★ Binary: 1010
- ★ Octal: 12
- ★ Hexadecimal: A

b. 16
- ★ Binary: 10000
- ★ Octal: 20
- ★ Hexadecimal: 10

c. 128

- ★ Binary: 10000000
- ★ Octal: 200
- ★ Hexadecimal: 80

d. 255

- ★ Binary: 11111111
- ★ Octal: 377
- ★ Hexadecimal: FF

e. 32767

- ★ Binary: 111111111111111
- ★ Octal: 77777
- ★ Hexadecimal: 7FFF

f. 1

- ★ Binary: 1
- ★ Octal: 1
- ★ Hexadecimal: 1

g. 0

- ★ Binary: 0
- ★ Octal: 0
- ★ Hexadecimal: 0

h. 26

- ★ Binary: 11010
- ★ Octal: 32
- ★ Hexadecimal: 1A

i. 31

- ★ Binary: 11111
- ★ Octal: 37
- ★ Hexadecimal: 1F

//

29. Convert following integer numbers into 2’s Complement binary form (8bits)

a. -10

★ Absolute value in binary: 00001010
★ Invert bits: 11110101
★ Add 1: 11110110

b. -100

★ Absolute value in binary: 01100100
★ Invert bits: 10011011
★ Add 1: 10011100

c. -64

★ Absolute value in binary: 01000000
★ Invert bits: 10111111
★ Add 1: 11000000

d. -1

★ Absolute value in binary: 00000001
★ Invert bits: 11111110
★ Add 1: 11111111

e. -2

★ Absolute value in binary: 00000010
★ Invert bits: 11111101
★ Add 1: 11111110

f. -128

★ Absolute value in binary: 10000000
★ Invert bits: 01111111
★ Add 1: 10000000 (Note that it remains the same because there is no positive representation for -128 in an 8-bit 2's complement form.)

g. 0

★ In 2's complement, 0 is represented as all 0s: 00000000

h. -127

★ Absolute value in binary: 01111111
★ Invert bits: 10000000
★ Add 1: 10000001

i. -32

- ★ Absolute value in binary: 00100000
- ★ Invert bits: 11011111
- ★ Add 1: 11100000

///

30. Write an equivalent compound statement. If no such statement is possible, write "No such a statement." Assume that u, v, w are variables of data type int.

a. u--;
- ★ u=u+1;

b. u = v - w + u;
- ★ No such a statement.

c. u = v- w*u
- ★ u = v- (w * u);

d. u = 3*(v + w) + u;
- ★ No such a statement.

e. u = u % v;
- ★ No such a statement.

///

31. Write an equivalent simple statement. If no such statement is possible, write "No such statement." Assume that u, v, w are variables of data type int.

a. u -= (v + w) ;
- ★ u =u -(v + w);

b. u -= w % v - v;
- ★ u= u - (w % v - v);

c. v *= u++;
- ★ v = v * u + 1;

d. u /= --u;
- ★ u = u / u - 1;

e. u += v % w;
- ★ u = u + (v % w);

///

32. Which of the following code fragments are legal?

a. double d='A';
long l=(int)d;

★ legal, although it may not produce the desired results. It assigns the ASCII value of 'A' to the double d, and then it casts d to an integer, resulting in 65. However, it's not a very common or meaningful use of data types.

b. byte b='65';
char ch=b';

★ This code is not legal. The single quotes around '65' should be double quotes to represent a string, and char and byte are not directly convertible from a string literal.

c. char ch='A';
double d=ch;

★ This code is legal. It assigns the character 'A' to the char ch and then assigns ch to the double d. The ASCII value of 'A' will be stored in d.

d. double d='A';

★ This code is legal but not meaningful. It assigns the ASCII value of 'A' to the double d.

e. char ch=(short)d;

★ This code is not legal. It tries to cast a double to a short, and then assign it to a char, which isn't a valid conversion.

f. float f=65;
int x=(char)f;

★ This code is legal, but it may not produce the expected results. It assigns the value 65 to the float f, and then it tries to cast f to a char, which results in the character with the ASCII value 65, which is 'A'. This value is then assigned to the integer x.

//
//////////

33. What will be the output when you compile and run the program? Explain your answers.

```
class Example{
public static void main(String args[]){
byte b1=10,b2=20,b3;
b3=b1+b2; //Line 1
b3=b1+1; //Line 2
b3=b1*2; //Line 3
short s1=10,s2=20,s3;
s3=s1+s2; //Line 4
s3=s1+1; //Line 5
s3=s*1; //Line 6
 int x1=10,x2=20,x3;
x3=x1+x2; //Line 7
x3=b1+b2; //Line 8
x3=b1+1; //Line 9
x3=b1*2; //Line 10
x3=s1+s2; //Line 11
x3=s1+1; //Line 12
x3=s1*1; //Line 13
}
}
```

- ★ line 1 : cannot directly assign an int to a byte.
- ★ line 2 : error
- ★ line 3 : error
- ★ line 4 : error
- ★ line 5 : error
- ★ line 6 : error
- ★ line 7 : legal
- ★ line 8 : legal
- ★ line 9 :legal
- ★ line 10:legal
- ★ line 11:legal
- ★ line 12:legal
- ★ line 13:legal

//

34. Given :

```
class Example{
 public static void main(String args[]){
 long l;
//Line 10
System.out.println(l);
}
}
```

Which of the following statements can be legally placed at Line 10 of the above program.
a. I = 2147483647;
b. I = 2147583647;
c. I = 0xabcd;
d. I = 0bcdL;
e. I = 0101010110L;
//
//////////

35. Given :
class Demo {
public static void main(String args[]) {
int tot = 971;
double avg;
//insert code here //Line 4
System.out.println("Average : " + avg);
}
}
Which of the following statements can be inserted at "Line 4" to get output as "Average : 97.1"
a. avg = (double) tot/10;
b. avg = tot/(double)10;
c. avg = (double)(tot/10)
d. avg = tot/10
e. None of above
//
//////////

36. Write the outputs for the following code lines.
Given Code:

int a=10, b=7, c=-10, d=-7;
a. System.out.println(a%b);
 ★ 3
b. System.out.println(-a%b);
 ★ -3
c. System.out.println(a%-b);
 ★ 3
d. System.out.println(-a%-b);
 ★ -3
e. System.out.println(c%d);
 ★ -3

f. System.out.println(-c%d);

- ★ 3

g. System.out.println(+a%+b);

- ★ 3

//

37. What will be the result of attempting to compile and run the following program?

```
class Example{
public static void main(String asrg[]){
double d;
d=5/2+5/2;
System.out.println(d);
d=5/2.0+5/2;
System.out.println(d);
d=5/2+5.0/2;
System.out.println(d);
d=5/2.0+5/2.0;
System.out.println(d);
}
}
```

a. 4.0 4.0 4 5.0
b. 4.0 4.5 4.5 5.0
c. 4 4.0 4.0 5.0
d. 4.5 4.5 4 5.0
e. 4 4.5 4.5 5

//

38. Which of the following lines can be inserted at the Line 12 to get the output "-1"

```
class Example{
public static void main(String args[]){
int x;
byte b;
//insert code here Line 12
b=(byte)x;
System.out.println(b);
}
}
```

a. x=Short.MAX_VALUE;
b. x=Short.MIN_VALUE;

c. x=-1;
d. x=Byte.MAX_VALUE;
e. x=Byte.MIN_VALUE;
f. x=0;
g. x=Integer.MAX_VALUE;
h. x=Integer.MIN_VALUE;
//
//////////

39. Which statements are true? Select the three correct answers.
a. The result of the expression (1 + 2 + "3") would be the string "33".
b. The result of the expression ("1" + 2 + 3) would be the string "15".
c. The result of the expression (4 + 1.0f) would be the float value 5.0f.
d. The result of the expression (10/9) would be the int value 1.
e. The result of the expression ('a' + 1) would be the char value 'b'.

//
//////////

40. Which of the following code lines are legal?
```
int x=65;
final int y=65;
final int z;
z=65;
char ch;
ch='A'; //Line 1
ch=65;//Line 2
ch=x; //Line 3
ch=y; //line 4
ch=z; //Line 5
```
a. Line 1
b. Line 2
c. Line 3
d. Line 4
e. Line 5
f. None of the above
//
//////////

41. Which of the following are legal lines of code?
a. int a = (int)888.8;
b. byte x = (byte)1000L;

c. long l = (byte)100;
d. byte z = (byte)100L;
//

42. Write the outputs for the following code lines.
Given: int x=10,y=7;
a. System.out.println(x+y);
- ★ 17

b. System.out.println(-x);
- ★ -10

c. System.out.println(-x-y);
- ★ -17

d. System.out.println(-(x-y));
- ★ -3

e. System.out.println(+y);
- ★ 7

f. System.out.println(+y-x);
- ★ -3

//

43. Write the outputs for the following code lines.
int x=-100;
x=+x;
System.out.println(x);
- ★ -100

x=-x;
System.out.println(x);
- ★ 100

x=-x;
System.out.println(x);
- ★ -100

x=x+x;
System.out.println(x);
- ★ -200

x=-x-x;
System.out.println(x);
- ★ 400

x=x-x;
System.out.println(x);
- ★ 0

//
//////////

44. Write the outputs for the following code lines.

```
int x=100;
System.out.print(x++);
System.out.println(x++);
x++;
System.out.println(++x);
System.out.println(x++);
```

- ★ 100101
- ★ 104
- ★ 104

//
//////////

45. Write the outputs for the following code lines.

```
int x=100,y;
y=x++;
System.out.println(x+" "+y);
y=x++;
System.out.println(x+" "+y);
y=x++;
System.out.println(x+" "+y);
```

- ★ 101 100
- ★ 102 101
- ★ 103 102

//
//////////

46. Write the outputs for the following code lines.

```
int x=100,y;
y=++x;
System.out.println(x+" "+y);
y=++x;
System.out.println(x+" "+y);
y=++x;
System.out.println(x+" "+y);
```

- ★ 101 101
- ★ 102 102
- ★ 103 103

///

47. Write the outputs for the following code lines.

```
int x=100;
x=x++;
System.out.println(x);
x=x++;
System.out.println(x);
x=x++;
System.out.println(x);
x=++x;
System.out.println(x);
x=++x;
System.out.println(x);
x=++x;
System.out.println(x);
```

- ★ 100
- ★ 100
- ★ 100
- ★ 101
- ★ 102
- ★ 103

///

48. Explain why the following code segment causes a compilation error.

```
int x;
(x++)++;
```

- ★ The code segment causes a compilation error because the increment operator ++ can only be applied to a variable, not to an expression or the result of another increment operation. In the code (x++)++;, you are trying to use the post-increment operator ++ on the result of (x++), which is not allowed in Java.
- ★ The post-increment operator x++ increases the value of x by 1 but returns the original value of x before the increment. Since (x++) is an expression that results in a value, you cannot increment the result of that expression using another ++ operator.

//
//////////

49. Write the outputs for the following code lines.

```
int x,y;
x=y=100;
x=x++ +x++ + x++ ;
System.out.println(x);
y=++y + ++y + ++y;
System.out.println(y);
y=x=100;
System.out.println();
x=x++ + ++y + ++x + y++;
System.out.println(x+" "+y);
```

★ 303
★ 306
★
★ 404 102

//
//////////

50. Explain the evaluation of following expressions

```
int a=10;
int x;
a. x= a++ + a;
b. x= a + a++;
c. x= ++a + a;
d. x= a + ++a;
e. x= ++a + ++a;
f. x= a++ + a++;
g. x= ++a + a++;
h. x= a++ + ++a;
i. x= ++a + a++;
j. x= a++ + ++a;
```

★ 20
★ 22
★ 25
★ 27
★ 30

- ★ 30
- ★ 36
- ★ 39
- ★ 42
- ★ 45

//

51. Explain the evaluation of following expressions
int a=10,b=20;
int x;
a. x= a + b;
b. x= a +- b;
c. x= ++a + b;
d. x= a + b++;

e. x= ++a + b++;
f. x= a++ + b++;
g. x= ++a + ++ b;
h. x= a++ + ++b;

- ★ 30
- ★ -10
- ★ 31
- ★ 21
- ★ 21
- ★ 11
- ★ 12
- ★ 22

//

52. Write the outputs for the following code lines.
Given code:
int a=10, b=7, c=-10, d=-7;
a. System.out.println(10%7);
b. System.out.println(10%5);
c. System.out.println(10%17);
d. System.out.println(5.0%1.0);
e. System.out.println(5.5%1.1);

- ★ 3
- ★ 0
- ★ 10
- ★ 0.0
- ★ 1.0999999999999996

//
//////////
53. Explain the evaluation of following expressions
int x;
a. x= 7 % 10 / 2 * 2;
   - ★ 7%10=7>>>7/2=3>>>3*2 = 6

b. x= 7 % (10 / 2) * 2;
   - ★ 10/2=5>>>7%5=2>>>2*2= 4

c. x= 7 % 10 / (2 * 2);
   - ★ (2 * 2)=4>>>7%10=7>>>7/4= 1

d. x= 7 % (10 / (2 * 2));
   - ★ (2 * 2)=4>>>10/4=2>>>7%2= 1

e. x= 7 % ((10 / 2) * 2);
   - ★ 10/2=5>>>5*2=10>>>7%10= 7

//
//////////

54. Explain the evaluation of following expressions
int a=100;
a. a= a + (a=6);
   - ★ 106

b. a= (a=6) + a;
   - ★ 12

c. a= (a=6) + (a=5);
   - ★ 11

d. a= a*3 + a;
   - ★ 400

//
//////////

55. What will be the result of attempting to compile and run the following program? Explain your answers.

```
class Example{
public static void main(String[] args) {
int x;
x= 12 - 4 * 2;
System.out.println("12 - 4 * 2 : "+x);

x= (12 - 4) * 2;
System.out.println("(12 - 4) * 2 : "+x);
x= 12 - (4 * 2);
System.out.println("12 - (4 * 2) : "+x);
}
}
     ★ 12 - 4 * 2 : 4
     ★ (12 - 4) * 2 : 16
     ★ 12 - (4 * 2) : 4
```

//
//////////

56. Correct the syntax errors.

```
a. public void class Demo1{

static public main(String args()){
final ratio = 1.8;
int b = c = 10;
d = c + 5;
c = 10c:
b = d / c;
ratio= b / d;
System.out.println("b = +", b);
System.out.println("c = ", b);
System.out.println(“ratio = “ b / d);
          }
}


     ★ public class Demo1 {
     ★ public static void main(String args[ ]){
     ★ final double ratio = 1.8;
     ★ int b =10;
     ★ int c =10;
     ★ int d;
     ★ d = c + 5;
     ★ c = 10*c;
     ★ b = d/c;
     ★ System.out.println("b = +"+ b);
     ★ System.out.println("c = "+ b);
     ★ System.out.println("ratio = "+ (b / d));
     ★ }
     ★ }
```

b. public class Demo2 {

```
public void main(String[] param) {
final char new = \7777\;
final int TEN;
int a, b, c;
TEN= a + 5;
b = a + new;
c = a + 5 * TEN;
(b + c) = b;
System.out.print ("a + a”);
System.out.print ("new + new”);
System.out.print ("TEN = + TEN”);
}
}
```

```
★ public class Demo2 {
★ public static void main(String[] param) {
★ final char newCh = '\u7777';
★ final int TEN;
★ int a, b, c;
★ a=10;
★ TEN= a + 5;
★ b = a + newCh;
★ c = a + 5 * TEN;
★ b=(b + c);
★ System.out.print ("a + a");
★ System.out.print ("new + new"); ★
System.out.print ("TEN = + TEN"); ★
}
★ }
```

c. void public class Demo3{

```
void public main(String args()) {
final int OFFSET 32;
int b; c= 7; d;
```

d = c + 5;
c = d - b;
b = d / c;
c = b % OFFSET++;
c++ = d + b;
System.out.println("B = + ", b);
System.out.println("C = ", c);
System.out.println("D " + d);
}
}

★ public class Demo3 {
★
★ public static void main(String[] args) { ★ final
int OFFSET = 32;
★ int b, c, d;
★ d = c + 5;
★ c = d - b;
★ b = d / c;
★ c = b % OFFSET++;
★ d += b;
★
★ System.out.println("B = " + b); ★
System.out.println("C = " + c); ★
System.out.println("D = " + d); ★ }
★ }

d. public class Demo4{

public void main(String args) {
final char new = “A”;
final int SIXTY_SIX = “B”;
int a, b, c,
10 = a;
b = a + new;
c = a + 5 * SIXTY SEVEN;
a = ++ (b + c);
System.out.print (“a: “ + a);
System.out.print (“b: “ + b);
System.out.print (“(a+b) : “ + (a+b));
}}

★ public class Demo4{

- ★ public static void main(String args[]) {
- ★
- ★ final char newch = 'A';
- ★ final int SIXTY_SIX = 'B';
- ★ int a, b, c;
- ★ a=10;
- ★ b = a + newch;
- ★ c = a + 5 * SIXTY_SIX;
- ★ a = a++ +(b + c);
- ★ System.out.println ("a: " + a);
- ★ System.out.println ("b: " + b);
- ★ System.out.println ("(a+b) : " + (a+b));
- ★ }}
- ★
- ★

//
//////////

57. What are the outputs of the following commands?
System.out.println(12+8/5%4*(5-4/5)+4*5); //Line 1
System.out.println(4%5*3-4/7+4%2-5/(5*4%5)); //Line 2
System.out.println(5-8%4*5(5/8*(3%4)*4)+8/4+1); //Line 3
System.out.println(1.5%2.1-5.4*1.1/(5.4%5)); //Line 4
System.out.println((5+4)%4+(5/8.0)+4); //Line 5
System.out.println(5-4*6(5%4-3)*5+6/(1.0/2.0)-5*4); //Line 6
System.out.println(7+3-4*4%6+4*2.5-3%2); //Line 7
System.out.println(5-7*(9%4)+5+8/7+2); //Line 8
System.out.println((2-5%5)-10.8%5.1*5*4); //Line 9
System.out.println(5+5-4/(3%1+5+(7-8)*4+5)); //Line 10
System.out.println(9%4*5+6%10-5*4); //Line 11
System.out.println(9%1+5-(5+5%2)-5/8+5); //Line 12
System.out.print(((7 * 2) % 5)+" "); //Line 13
System.out.print(" " + (7 % 5)); // Line 14

- ★ 37
- ★ 8
- ★ 8
- ★ -13.349999999999
- ★ 5.625
- ★ -37.0
- ★ 15.0

- ★ 6
- ★ -10.000000000000028
- ★ 10
- ★ -9
- ★ 4
- ★ 4 2

//
//////////
58. What are the outputs of the following commands?

```
int a=1,b=2,c=3,d=4;
int x;
x=a++ + b++ + c++ + d++;
System.out.println(a+" "+b+" "+c+" "+d+" "+x); //Line 1
x+=a+=b+=c+=d; //Line 2
System.out.println(a+" "+b+" "+c+" "+d+" "+x); //Line 3
x=a=b=c=d; //Line 4
System.out.println(a+" "+b+" "+c+" "+d+" "+x); //Line 5
```

- ★ 2 3 4 5 10
- ★ 14 12 9 5 24
- ★ 5 5 5 5 5

//
//////////

59. What are the outputs of the following commands?

```
boolean b = false;
System.out.println(10>4 && true !=b==(10+3/2==8)==true); //Line 1
System.out.println(b=false==true||true!=(b=false)); //Line 2
System.out.println((b=false)=true?4:5==5&true==3*2<=29); //Line 3
System.out.print (('a' == 'a')+" "); //Line 4
System.out.print(('a' == 'b')+" "); //Line 5
System.out.print((5 != 6)+" "); //Line 6
System.out.print((5.0 == 5L)+" "); //Line 7
System.out.println((true == false)); //Line 8
```

- ★ false
- ★ true
- ★ error
- ★ true false true true false

///
//////////

60. What are the outputs of the following commands?
boolean b1, b2, b3;
b1 = true != false; //Line 1
b2 = 5%3 == 2 ^ true == !false ; //Line 2
System.out.println((b3 =true) & b2 || b1 == false); //Line 3
System.out.println(b3 = b2 == b1); //Line 4
b3= true; //Line 5
System.out.println(b3^b2&b1|false != (b3 = false)); //Line 6
System.out.println(!b3==b2 && b2 != b1 ||!b1 != b2); //Line 7

★ Line 3>>>>false
★ Line 4>>>>false
★ Line 6>>>>true
★ Line 7>>>>false

///
//////////

61. Write a program in Java that prompts the user to input temperature in Centigrade. The program should then output the temperature in Fahrenheit.
Temperature in degrees Celsius = (Temperature in degrees Fahrenheit - 32) * 5/9

★ import java.util.*;
★
★ public class Main {
★ public static void main(String args[]) {
★ Scanner input=new Scanner (System.in);
★ System.out.print(" input temperature in Centigrade : ");
★ int celsius =input.nextInt();
★ double Fahrenheit = (celsius - 32) * 5/9;
★
★ System.out.print("temperature in Fahrenheit : "+Fahrenheit);
★ }
★ }

///

//////////

62. Write a Java program to take weight and height of a man as input and compute the body mass index (BMI). BMI is a person's weight in kilograms divided by the square of height in Meters.

★ import java.util.*;
★
★ public class Main {
★ public static void main(String args[]) {
★ Scanner input=new Scanner (System.in);
★ System.out.print(" input weight : ");
★ double weight=input.nextDouble();
★
★ System.out.print(" input height : ");
★ double height=input.nextDouble();
★
★ double bmi=weight/(height*height);
★
★ System.out.print("BMI : "+bmi);
★ }
★ }

//
//////////

63. Write a program in Java to take traveled distance in meters and time in seconds as user input and calculate the speed in meters per hour.

★ import java.util.*;
★
★ public class Main {
★ public static void main(String args[]) {
★ Scanner input=new Scanner (System.in);
★ System.out.print(" Enter travelled distance in meters: ");
★ double meters=input.nextDouble();
★
★ System.out.print(" Enter time in seconds: ");
★ double seconds=input.nextDouble();
★
★ double meters_per_hour=(meters/seconds)*3600;
★

★ System.out.print("Speed in meters per hour : "+meters_per_hour);
★ }
★ }

//
//////////

64. Write a program in Java that reads an integer between 100 and 999 and adds all the digits in the integer.

★ import java.util.Scanner;
★
★
★ public class Main {
★
★ public static void main(String args[]) {
★
★ Scanner input = new Scanner(System.in);
★ System.out.print("Input a number: ");
★ int num = input.nextInt();
★ int total = 0;
★
★ do {
★ if(num>100||num<999){
★
★ int temp = num % 10;
★ total += temp;
★
★ num /= 10;
★ }
★ } while(num != 0);
★ System.out.println("Sum of digits : " + total);
★ }
★ }

//
//////////

65. Write a Java program to calculate the percentage of the marks taking the obtained marks and maximum marks as the input and display the percentage with the appropriate Message.

★ import java.util.Scanner;
★

★
★ public class Main {
★
★ public static void main(String args[]) {
★
★ Scanner input = new Scanner(System.in);
★ System.out.print("Input obtain marks : ");
★ double obtain = input.nextInt();
★
★ System.out.print("Input maximum marks : ");
★ double max = input.nextInt();
★ double percentage=(obtain/max)*100;
★ System.out.print("percentage : "+percentage+" %");
★
★ }
★ }

///
//////////

66. Write a Java program that takes a delay time in seconds as input and calculates the number of days, hours, minutes, and seconds it took for a mail to reach its destination.
Expected Output:
Input delay time in seconds: 86399
23:59:59

★ import java.util*;
★
★ public class Main {
★
★ public static void main(String args[]) {
★ Scanner input = new Scanner(System.in);
★
★ System.out.print("Input delay time in seconds: ");
★ int delayInSeconds = input.nextInt();
★
★ int days = delayInSeconds / (60 * 60 * 24);
★
★ int remainingSeconds = delayInSeconds % (60 * 60 * 24);
★ int hours = remainingSeconds / (60 * 60);
★ remainingSeconds %= (60 * 60);
★ int minutes = remainingSeconds / 60;
★ int seconds = remainingSeconds % 60;

★
★ System.out.printf("%02d:%02d:%02d:%02d%n", days, hours, minutes, seconds); ★
★ }
★ }

//
//////////

67. Write a Java program to calculate the commission earned by a salesperson when user input the total sales amount and the commission rate.

★ import java.util.Scanner;
★
★
★ public class Main {
★
★ public static void main(String args[]) {
★ Scanner input = new Scanner(System.in);
★
★ System.out.print(" qty of product : ");
★ int qty = input.nextInt();
★
★ System.out.print(" price of product : ");
★ double price = input.nextInt();
★
★ double salesAmount=price*qty;
★ double commission=(salesAmount*5)/100;
★
★ System.out.println(" total sale : "+salesAmount);
★ System.out.println(" commission : "+commission);
★
★
★ }
★ }

//
//////////

68. Write a Java program to solve the linear equation ax + b = 0 for different user inputs a and b.

★ import java.util.*;

★
★ public class Main {
★ public static void main(String[] args) {
★ Scanner input = new Scanner(System.in);
★
★ System.out.print("Enter the value a: ");
★ double a = input.nextDouble();
★
★ System.out.print("Enter the value b : ");
★ double b = input.nextDouble();
★
★ if (a == 0) {
★ if (b == 0) {
★ System.out.println("identity equation");
★ } else {
★ System.out.println("no equation");
★ }
★ } else {
★ double x = -b / a;
★ System.out.println("Solution for x: " + x);
★ }
★
★
★ }
★ }

//
//////////

69. Write a Java program to compute the average temperature of the day from five temperature readings.

★ import java.util.*;
★
★ public class Main {
★ public static void main(String[] args) {
★ Scanner input = new Scanner(System.in);
★
★ double sum=0;
★
★ for(int i=1;i<=5;i++){
★ System.out.print(" enter the temperature "+i+" :");

★ double temperature = input.nextDouble();
★ sum=sum+temperature;
★ }
★ double avgtemp=sum/5;
★ System.out.print(" avarage temperature of day :"+avgtemp);
★
★ }
★ }

//

70. If you have N eggs, then you have N/12 dozen eggs, with N%12 eggs left over. (This is essentially the definition of the / and % operators for integers.). Write a Java program that interacts with a user to calculate and disp laythe number of dozen eggs, gross, and any extra eggs based on the number of eggs thehas user . A gross is defined as 144 eggs.

★ import java.util.*;
★
★ public class Main {
★ public static void main(String[] args) {
★ Scanner input = new Scanner(System.in);
★
★ System.out.print("Enter the number of eggs: ");
★ int numEggs = input.nextInt();
★
★ int dozenEggs = numEggs / 12;
★ int extraEggs = numEggs % 12;
★
★ int gross = numEggs / 144;
★
★ System.out.println("Number of Dozen Eggs: " + dozenEggs);
★ System.out.println("Number of Gross: " + gross);
★ System.out.println("Extra Eggs: " + extraEggs);
★
★
★ }
★ }

//

71. Write a Java program to compute the bonus for an employee based on their monthly

salary. The bonus is calculated as follows: it is $1000 plus 2% of the amount above $7000 of the employee's annual salary. Assume that every employee has an annual salary above $7000.

```
★ import java.util.*;
★
★ public class Main {
★ public static void main(String[] args) {
★ Scanner input = new Scanner(System.in);
★
★ System.out.print("Enter the monthly salary: ");
★ double monthlySalary = input.nextDouble();
★
★ double annualSalary = monthlySalary * 12;
★
★ double bonus = 1000 + (annualSalary - 7000) * 0.02;
★
★ System.out.println(" Bonus: $" + bonus);
★
★
★ }
★ }
```

//
//////////

72. Given three integers runs, innings, and notOut, representing the total number of runs scored, the total number of innings played by the player, and the number of times he remained not out, respectively, you are asked to write a Java program to calculate the batting average of the player using the formula:
average = runs / (innings - notOut)

```
★ import java.util.*;
★
★ public class Main {
★ public static void main(String[] args) {
★ Scanner input = new Scanner(System.in);
★
★ System.out.print("Enter the total number of runs : ");
★ double runs = input.nextDouble();
★
★ System.out.print("Enter the total number of innings : ");
★ double innings = input.nextDouble();
```

★
★ System.out.print("Enter the number of times the player remained not out: "); ★ double notOut = input.nextDouble();
★
★ double average = runs / (innings - notOut);
★
★ System.out.println("Batting Average: " + average);
★
★ }
★ }

//

73. Write a Java program that takes the radius of a circle as input and calculate the area and perimeter of it.
★ import java.util.*;
★
★ public class Main {
★
★ public static void main(String[] args) {
★ Scanner input = new Scanner(System.in);
★
★ System.out.print("Enter the radius : ");
★ double radius = input.nextDouble();
★ double x=3.14;
★ double area = x*(radius*radius);
★ double perimeter = 2 * x * radius;
★
★ System.out.println("Area of the circle: " + area);
★ System.out.println("Perimeter of the circle: " + perimeter);
★
★ }
★ }

//

74. How would you write the following arithmetic expression in Java?
a.
4
3(r + 34)
− 9(a + bc) +

3 + d(2 + a)
a +bd

- System.out.println(4/(3(r+34))-9(a+bc)+3+d(2+a)/(a+bd));

b. 5.5 × (r + 2.5)
2.5 + t

- System.out.println(5.5*(r+2.5))^2.5+t ;

//
//////////

75. Write a Java program to calculate the Z-scores for a student's A/L exam results in three subjects. The program should take user input for the marks obtained by the student in each subject and use the following formula and table data to calculate the Z-scores: Subjects 1 2 3
Mark Obtained by the Student x1 x2 x3
Mean Mark of the Subject m1 m2 m3
Standard Deviation Mark of the Subject sd1 sd2 sd3
z score of the student = [(x1 − m1)/sd1 + (x2 − m2)/sd2 + (x3 − m3)/sd3]/3

```
★ import java.util.*;
★
★ public class Main {
★
★ public static void main(String[] args) {
★
★ Scanner input = new Scanner(System.in);
★
★ System.out.print("Enter marks obtained in Subject 1: "); ★
double x1 = input.nextDouble();
★
★ System.out.print("Enter marks obtained in Subject 2: "); ★
double x2 = input.nextDouble();
★
★ System.out.print("Enter marks obtained in Subject 3: "); ★
double x3 = input.nextDouble();
★
★ System.out.print("Enter mean mark for Subject 1: ");
★ double m1 = input.nextDouble();
★
★ System.out.print("Enter mean mark for Subject 2: ");
★ double m2 = input.nextDouble();
★
★ System.out.print("Enter mean mark for Subject 3: ");
```

★ double m3 = input.nextDouble();
★
★ System.out.print("Enter standard deviation for Subject 1: "); ★
double sd1 = input.nextDouble();
★
★ System.out.print("Enter standard deviation for Subject 2: "); ★
double sd2 = input.nextDouble();
★
★ System.out.print("Enter standard deviation for Subject 3: "); ★
double sd3 = input.nextDouble();
★
★ double zScore = ((x1 - m1) / sd1 + (x2 - m2) / sd2 + (x3 - m3) / sd3) / 3.0; ★
★ System.out.println("Z-score of the student: " + zScore);
★
★
★ }
★ }
//

76. Write a Java program to calculate the distance between two points when user input coordinates of two points.
For the first point, enter the x and y coordinates (e.g., (x1, y1)).
For the second point, enter the x and y coordinates (e.g., (x2, y2))

Distance = √(x2 − x1)

2 + (y2 − y1)
2
//

77. Write a Java program to calculate the compound interest earned on an investment based on user input for principal amount, time, and interest rate.
Compound Interest, CI = P (1 +R) T
100

- Where P is the principal amount.
- R is the annual interest rate.
- T is the time period in years

//

78. Write a Java program to calculate the roots of Quadratic equations which have real and distinct roots.

$$x = \frac{-b \pm \sqrt{b2 - 4ac}}{2a}$$

//
//////////

****