Data types, variables, constants, operators, casting

1. Basing on the example below, please implement the program, where:
    a) Declare and initialize three variables of type ***int***: a, b, c (initialize it to any values)
    b) Store the result of the ***a - b - c*** operation in the variable ***result1*** and then display it on the standard output (screen)
    c) Declare and initialize three variables of type ***long***: d, e, f (initialize it to any values)
    d) Store the result of the ***d * e / f*** operation in the variable ***result2*** and then display it on the standard output (screen)

Example:

```
public class Calculator {
   public static void main(String[] args) {
       int a = 5;
       int b = 2;
       System.out.println("a + b = " + (a + b));
   }
}
```

Sample solution:

```
public class Calculator {
   public static void main(String[] args) {
       int a = 5;
       int b = 2;
       int c = 3;
       int result1 = a - b - c;
       System.out.println("a - b - c = " + result1);

       long d = 52L;
       long e = 12L;
       long f = 8L;
       long result2 = d * e / f;
       System.out.println("d * e / f = " + result2);
   }
}
```

2. Please, implement the program, where you will declare and initialize several final variables of various types having any names. Next, try to display them in the following lines of text. Compile it and check what will happen, when you try to set the value again for any previously declared final variable.

Sample solution:

```
public class FinalVariables {
   public static void main(String[] args) {
       final int finalInt = 5;
       final String finalString = "This is the test!";
       final boolean finalBoolean = true;
       final long finalLong = 12L;
       final float finalFloat = 5f;

       System.out.println(finalInt);
       System.out.println(finalString);
       System.out.println(finalBoolean);
       System.out.println(finalLong);
       System.out.println(finalFloat);

       finalInt = 6;       // will cause compilation error
       finalLong = 15L;     // will cause compilation error
   }
}
```

3. Display on the screen in the following lines the values of the logical expressions listed below:

2 * 2 >= 3 && 1 == 1
(12/4) != 3
12 % 4 != 0
3 != 4 || (2 + 3 > 5)

Sample solution:

```
public class BooleanExpressions {
   public static void main(String[] args) {
       System.out.println(2 * 2 >= 3 && 1 == 1);
       System.out.println(12/4 != 3);
       System.out.println(12 % 4 != 0);
       System.out.println(3 != 4 || (2 + 3 > 5));
```

```
    }
}
```

4. Implement the program, where you have to:
    a) Declare and initialize two variables: ***intVar1, intVar2*** of ***int*** type
    b) Declare variable ***shortSum*** of ***short*** type and initialize it as the sum of previously declared variables (intVar1 + intVar2)
    c) Display on the screen the value stored in ***shortSum*** variable
    d) Next, display on the screen the result of the incrementation: ***shortSum***++
    e) Declare variable ***byteSum*** of ***byte*** type and initialize it as the sum of previously declared variables (intVar1 + intVar2)
    f) Display on the screen the value stored in ***byteSum*** variable
    g) Next, display on the screen the result of the incrementation: ++***byteSum***

Sample solution:

```java
public class CastExample {
    public static void main(String[] args) {
        int intVar1 = 12;
        int intVar2 = 15;
        short shortSum = (short) (intVar1 + intVar2);
        System.out.println(shortSum);
        System.out.println(shortSum++);
        byte byteSum = (byte) (intVar1 + intVar2);
        System.out.println(byteSum);
        System.out.println(++byteSum);
    }
}
```

String Class

5. Please, declare three variables of ***String*** type and assign it values. Then declare a fourth variable of type String which will be a concatenation of previously declared variables and display its value on the screen. Please do it in two ways:
    a) Using '+' operator
    b) Using ***concat()*** method from ***String*** class

Sample solutions:

```
public class StringConcatenation {
   public static void main(String[] args) {
        String str1 = "First string ";
        String str2 = "Second string ";
        String str3 = "Third string";
        String sum = str1 + str2 + str3;
        System.out.println(sum);
   }
}
```

```
public class StringConcatenation {
   public static void main(String[] args) {
        String str1 = "First string ";
        String str2 = "Second string ";
        String str3 = "Third string";
        String sum = str1.concat(str2).concat(str3);
        System.out.println(sum);
   }
}
```

6. Declare two variables of ***String*** type and next please declare the ***boolean*** variable, which will store the result of checking if two previously declared ***String*** values are equal (**Hint: use equals() method from String class**). Display ***boolean*** value on the standard output (screen).

Sample solution:

```
public class StringComparing {
   public static void main(String[] args) {
        String str1 = "Checking if Strings are equal";
        String str2 = "Checking if strings are equal";
        boolean result = str1.equals(str2);

        System.out.println(result);
    }
}
```

Conditional statements

7. Please, declare an integer variable and assign any value to this variable. Then using a conditional statement check if it is an even number. If yes, please display proper information on the screen.

Sample solution:

```java
public class ConditionalStatements {
   public static void main(String[] args) {
       int var = 12;
       if (var % 2 == 0) {
           System.out.println("It is an even number!");
       }
   }
}
```

8. Please, modify the program from the previous exercise adding proper information in case the number is odd (**Hint: use another conditional statement construction**).

Sample solution:

```java
public class ConditionalStatements {
   public static void main(String[] args) {
       int var = 12;
       if (var % 2 == 0) {
           System.out.println("It is an even number!");
       } else {
           System.out.println("It is an odd number!");
       }
   }
}
```

9. Declare an integer variable and assign any value to it. Next check, if the value of this variable is greater, or less, or equal to zero. In every case, please display the proper information on the screen. Please use conditional statement construction.

Sample solution:

```
public class ComplexConditionalStatements {
   public static void main(String[] args) {
       int var = 12;
       if (var < 0) {
           System.out.println("Value is less than zero!");
       } else if (var == 0) {
           System.out.println("Value is equal to zero!");
       } else {
           System.out.println("Value is greater than zero!");
       }
   }
}
```

Loops

10. Implement the program displaying all the numbers from the range 5 - 100 (use ***'for'*** loop).

Sample solution:

```
public class ForLoopExample {
   public static void main(String[] args) {
       for (int i = 5; i < 101; i++) {
           System.out.println(i);
       }
   }
}
```

11. Implement the program from the previous exercise, but using a ***'while'*** loop.

Sample solution:

```
public class WhileLoopExample {
   public static void main(String[] args) {
       int i = 5;
       while (i < 101) {
           System.out.println(i++);
       }
   }
}
```

12. Implement the program displaying all the numbers from the range 1 - 100 which are divisible by 5 beginning from 100 (in reverse order).

Sample solution:

```
public class DivisionByFiveExample {
   public static void main(String[] args) {
       for (int i = 100; i > 0; i = i - 5) {
           System.out.println(i);
       }
```

```
    }
}
```

13. Implement the program computing the sum of factors from the arithmetic sequence. The first factor value of the sequence is equal to 5, the difference of each subsequent factor is equal to 2. We want to sum 459 elements (use ***'for'*** loop).

Sample solution:

```
public class ArithmeticSequenceExample {
    public static void main(String[] args) {
         int sum = 0;
         for (int i = 5; i <= 460 * 2 + 1; i = i + 2) {
              sum = sum + i;
         }
         System.out.println("Sum = " + sum);
    }
}
```

14. Implement the program which will be displaying on the standard output (screen) the character sequence as below:

```
*
**
***
****
*****
```

We want to receive as many rows as the value assigned to the variable ***n*** at the beginning of the program.

Sample solution:

```
public class NestedLoopsExample {
    public static void main(String[] args) {
         int n = 10;
         for (int i = 0; i < n; i++) {
              for (int j = 0; j < i + 1; j++) {
                   System.out.print("*");
              }
              System.out.println();
         }
```

```
        }
    }
```

Tables

15. Implement the program, where you have to declare and create a 5-element one-dimensional table of ***String*** type. In the next step, please read 5 names using standard input (using ***Scanner*** class) and simultaneously save them in the previously created table. In the last step, please display on the standard output (screen) a notification: “Hello ${name}!” for every name saved in the table. **Hint: In place of ${name} you should insert proper value from the table.**

Sample solution:

```java
public class OneDimensionalTableExample {
   public static void main(String[] args) {
       int n = 5;
       String[] names = new String[n];
       Scanner scanner = new Scanner(System.in);
       for (int i = 0; i < n; i++) {
           System.out.print("Enter the name: ");
           names[i] = scanner.nextLine();
       }
       for (int i = 0; i < names.length; i++) {
           System.out.println("Hello " + names[i]);
       }
   }
}
```

16. Implement the program, where you have to read 6 integers (using ***Scanner*** class) and save them in the previously declared 6-element table. Then compute an arithmetic average for all table values and display the result on the standard output.

Sample solution:

```java
public class ArithmeticAverage {
   public static void main(String[] args) {
       int n = 6;
       int[] tab = new int[n];
       Scanner scanner = new Scanner(System.in);
       for (int i = 0; i < n; i++) {
           System.out.print("Please enter the integer value: ");
           tab[i] = scanner.nextInt();
```

```
        }
        float sum = 0;
        for (int i = 0; i < tab.length; i++) {
            sum += tab[i];
        }
        System.out.println("Arithmetic average: " + (float) (sum /
tab.length));
    }
}
```

17. Please, implement the program, which will be displaying 2-dimensional table like this below:

```
0    0    0    0    0
0    1    0    0    0
0    0    2    0    0
0    0    0    3    0
0    0    0    0    4
```

Notice: Use a 2-dimensional table as a structure. Do not initialize the table in the static way - please, fill the table values using ***'for'*** loops.

Sample solution:

```
public class MatrixExample {
    public static void main(String[] args) {
        int n = 5;
        int[][] tab = new int[n][n];
        for (int i = 0; i < n; i++) {
            tab[i][i] = i;
        }
        for (int i = 0; i < tab.length; i++) {
            for (int j = 0; j < tab[i].length; j++) {
                System.out.print(tab[i][j]);
            }
            System.out.println();
        }
    }
}
```

18. Using loops and 2-dimensional table storing integers, present structure and data as below:

tab[0,0] = 0; tab[0,1] = 1; tab[0,2] = 2;
tab[1,0] = 3; tab[1,1] = 4; tab[1,2] = 5;

Please, use table ***length*** property.

Sample solution:

```
public class MatrixExample {
   public static void main(String[] args) {
       int n = 2;
       int m = 3;
       int[][] tab = new int[n][m];
       int val = 0;
       for (int i = 0; i < n; i++) {
           for (int j = 0; j < m; j++) {
               tab[i][j] = val++;
           }
       }
       for (int i = 0; i < tab.length; i++) {
           for (int j = 0; j < tab[i].length; j++) {
               System.out.print("tab[" + i + "," + j + "]=" +
tab[i][j] + ";");
           }
           System.out.println();
       }
   }
}
```

Object

Oriented Programming (class, object, fields, methods, constructors, packages, imports)

19. Please implement a method, which takes three integers as input parameters and returns the result of multiplication of these parameters.

Sample solution:

```
public int multiple(int a, int b, int c) {
   return a * b * c;
}
```

20. Please implement a method, which takes three parameters of String type and returns the result of concatenation of these parameters.

Sample solution:

```
public String concat(String a, String b, String c) {
   return a + b + c;
}
```

21. Please, implement getDistinct() method, which as an input parameter has a table of String type values and which returns a table containing unique elements from the input table.

Sample solution:

```
public static String[] getDistinct(String[] table) {
   String[] out = new String[table.length];
   int k = 0;
   for (int i = 0; i < table.length; i++) {
       boolean found = false;
```

```
            for (int j = 0; j < out.length; j++) {
                if (table[i].equals(out[j])) {
                    found = true;
                    break;
                }
            }
            if (!found) {
                out[k++] = table[i];
            }
        }
        return out;
    }
```

22. Implement a class named ***Rectangle***, which has two integer fields: ***a, b*** (the side lengths of the rectangle). Please, add a constructor, which will set values of all fields. Add to class definition also the method named ***getArea()***, which returns the area of the rectangle and ***getPerimeter()*** returning figure perimeter.
    For the created class definition add a test class with ***main()*** method, where you have to create an instance of type ***Rectangle*** using two-arguments constructor, then call ***getArea()*** and ***getPerimeter()*** methods and display returned values on the standard output.

Sample solution:

```
public class Rectangle {
    private int a;

    private int b;

    public Rectangle(int a, int b) {
        this.a = a;
        this.b = b;
    }

    public int getArea() {
        return a * b;
    }

    public int getPerimeter() {
        return 2 * a + 2 * b;
    }
}
```

```java
public class RectangleTest {
   public static void main(String[] args) {
       Rectangle rectangle = new Rectangle(10, 15);
       System.out.println("Rectangle area: " + rectangle.getArea());
       System.out.println("Rectangle perimeter: " +
rectangle.getPerimeter());
   }
}
```

23. Implement a class named ***Employee***, which consists of three fields: ***firstName***, ***lastName*** and ***age***. Add a three-argument constructor, which allows initializing field values and ***getter*** for ***age*** field. In the next step, please create class named ***Company***, which consists of the fields:
    a) ***name*** of ***String*** type - company name
    b) ***employees***, which is a 5-element table of ***Employee*** type (Employee[]) - table will store all company employees

In class Company, please also add one-argument constructor, which takes ***name*** parameter as an argument, create ***setEmployees()*** method definition for setting company employees. Implement ***toString()*** methods for every class (***Employee*** and ***Company***, use Alt+Insert -> toString in IntelliJ), which will be returning whole object data. Implement also ***getAverageAge()***, which returns the average age of company employees.
In the last step, please create a test class named ***CompanyTest***. In the ***main()*** method, create 5 instances of ***Employee*** type, which will be a representation of 5 employees working in the company. Then add them to the table and create an instance of type ***Company*** using one-argument constructor. Set ***employees*** field values using previously created setter. Next, please display all company data on the standard output (hint: ***toString()*** methods will be helpful) and average age computed by calling ***getAverageAge()*** method on ***Company*** type instance.

Sample solution:

```java
public class Employee {
   private String firstName;

   private String lastName;

   private int age;

   public Employee(String firstName, String lastName, int age) {
       this.firstName = firstName;
       this.lastName = lastName;
       this.age = age;
```

```
    }

    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
        return "Employee{" +
                "firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", age=" + age +
                '}';
    }
}

public class Company {
    private String name;

    private Employee[] employees = new Employee[5];

    public Company(String name) {
        this.name = name;
    }

    public void setEmployees(Employee[] employees) {
        this.employees = employees;
    }

public float getAverageAge() {
    float sum = 0;
    for (int i = 0; i < employees.length; i++) {
        sum += employees[i].getAge();
    }
    return sum / employees.length;
}

    @Override
    public String toString() {
        return "Company{" +
                "name='" + name + '\'' +
                ", employees=" + Arrays.toString(employees) +
                '}';
    }
}

public class CompanyTest {
    public static void main(String[] args) {
        Employee[] employees = new Employee[]{
                new Employee("John", "Doe", 35),
                new Employee("Kate", "Doe", 25),
                new Employee("Mark", "Doe", 29),
                new Employee("Ann", "Doe", 12),
                new Employee("Richard", "Doe", 57)
        };
        Company company = new Company("XCompany");
        company.setEmployees(employees);
        System.out.println("Company data: " + company);
        System.out.println("Average employee age: " +
company.getAverageAge());
    }
}
```

Varargs

24. Implement a method taking any amount of integer arguments (hint: use ***varargs***) and returning the sum of argument values. For sum calculation, please use ***'for'*** loop construction. In the last step, please test this method: call method for 1, 2, 3 and 4 arguments. In each case display returned value on the standard output.

Sample solution:

```
public class VarargsExample {
   public static void main(String[] args) {
       System.out.println("Sum for 1 parameter: " + sum(5));
       System.out.println("Sum for 2 parameters: " + sum(5, 6));
       System.out.println("Sum for 3 parameters: " + sum(5, 6, 7));
       System.out.println("Sum for 4 parameters: " + sum(5, 6, 7, 8));
   }

   public static int sum(int...args) {
       int sum = 0;
       for (int i = 0; i < args.length; i++) {
           sum += args[i];
       }

       return sum;
   }
}
```

Date and time

25. Implement a method, which takes one parameter of type ***LocalDate*** and returns an information if this date is earlier than the current one. Please, also implement ***main()*** test method.

Sample solution:

```java
public class LocalDateExample {
   public static void main(String[] args) {
       LocalDate date = LocalDate.of(2020, 06, 26);
       System.out.println("Date " + date + " is earlier than now: " +
isEarlierThanNow(date));
   }

   public static boolean isEarlierThanNow(LocalDate date) {
       LocalDate now = LocalDate.now();
       System.out.println("Now is: " + now);

       return date.isBefore(now);
   }
}
```

Regular Expressions (Regex)

26. Please, implement a method which for a table of String values as method parameter, will return a table containing only those values which represent correct email addresses (hint: for checking if a String value is the proper email address, look for regular expression in the internet).

Sample solution:

```
public static String[] checkEmails(String[] in) {
   Pattern pattern =
Pattern.compile("(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/
=?^_`{|}~-]+)*|\"(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21\\x23-\\x5b\
\x5d-\\x7f]|\\\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f])*\")@(?:(?:[a-z0-9]
(?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\\[(?:(?:2
5[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]
?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\
x21-\\x5a\\x53-\\x7f]|\\\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f])+)\\])");

   int j = 0;
   String[] out = new String[in.length];
   for (int i = 0; i < in.length; i++) {
       Matcher matcher = pattern.matcher(in[i]);
       if (matcher.matches()) {
           out[j++] = in[i];
       }
   }
   return out;
}
```

Fields, methods and static classes

27. Please implement class named ***MathUtils***, which will have two static methods:
    a) Method named ***power()***, which for two integer arguments will return the value of the first argument raised to the power of the second argument.
    b) Method named ***factorial()***, which for integer argument will return the factorial value for this argument (i.e. n! = 1*2*...*(n-1)*n)

Please also create a test class named ***MathUtilsTest*** with ***main(***) test method, where you call these static methods with sample argument values.

Sample solution:

```
public class MathUtils {
   public static int power(int a, int b) {
       int out = 1;
       for (int i = 0; i < b; i++) {
           out *= a;
       }
       return out;
   }

   public static int factorial(int n) {
       if (n < 0) {
           return 0;
       } else if (n == 0) {
           return 1;
       } else {
           int out = 1;
           for (int i = 1; i <= n; i++) {
               out = out * i;
           }
           return out;
       }
   }
}
```

```java
public class MathUtilsTest {
   public static void main(String[] args) {
       int a = -3;
       int b = 5;
       System.out.println(a + "^" + b + "=" + MathUtils.power(a, b));
       int n = 3;
       System.out.println(n + "!=" + MathUtils.factorial(n));
   }
}
```