1. **Classes exercise.**

   Implement console program which will meet the following requirements:

   a. Implement class Book that has next properties:
      i. int id - unique identifier
      ii. String name - name of a book
      iii. Author[] authors - array of authors
      iv. Publisher publisher - book publisher
      v. int publishingYear - year of publishing
      vi. int amountOfPages - amount of pages
      vii. BigDecimal price - price of a book
      viii. CoverType coverType - type of book binding, enum (Paperback, Hardcover)

   b. Implement Author class that has next properties:
      i. int id - unique identifier
      ii. String firstName - author's first name
      iii. String lastName - author's last name

   c. Implement Publisher class that has next properties:
      i. int id - unique identifier
      ii. String publisherName - name of a publisher

   d. Implement multiple constructors for Book, Author and Publisher class.

   e. Implement enum type CoverType with two enums - Paperback and Hardcover

   f. Override toString methods for Book, Author and Publisher class.

   g. Implement class BookService with the next methods

```
public Book[] filterBooksByAuthor(Author author, Book[] books) {
        <write your code here>
}

public Book[] filterBooksByPublisher(Publisher publisher, Book[]
books) {
        <write your code here>
}

// methods keeps books with publishing year inclusively.
public Book[] filterBooksAfterSpecifiedYear(int yearFromInclusively,
Book[] books) {
        <write your code here>
}
```

i. ALL METHODS SHOULD USE STREAM API

h. Implement a Demo class that has the main method. In demo class perform next actions:
   i. Create an array of books. You can use the next variable for demo purposes.

```
Book[] books = new Book[] {
new Book(1, "Book_1", new Author[] { new Author(1, "Jon", "Johnson") }, new Publisher(1, "Publisher_1"), 1990, 231, BigDecimal.valueOf(24.99), CoverType.PAPERBACK),

new Book(2, "Book_2", new Author[] { new Author(1, "Jon", "Johnson"), new Author(2, "William", "Wilson") }, new Publisher(2, "Publisher_2"), 2000, 120, BigDecimal.valueOf(14.99), CoverType.PAPERBACK),

new Book(3, "Book_3", new Author[] { new Author(3, "Walter", "Peterson") }, new Publisher(1, "Publisher_1"), 1997, 350, BigDecimal.valueOf(34.99), CoverType.HARDCOVER),

new Book(4, "Book_4", new Author[] { new Author(4, "Craig", "Gregory") }, new Publisher(3, "Publisher_3"), 1992, 185, BigDecimal.valueOf(19.99), CoverType.PAPERBACK) };
```

   ii. Create an instance of the BookService type and demonstrate the work of BookService methods, namely - filterBooksByAuthor, filterBooksByPublisher, filterBooksAfterSpecifiedYear.

2. **Inheritance and polymorphism exercise**

Implement console program which will meet the following requirements:

a. Implement a hierarchy of Sweets. All Sweets should have next properties: name, weight and sugarWeight. Weight is a weight of the specific sweet in kilograms. sugarWeight is a weight of sugar in this sweet in kilograms.

   I don’t want to give you specific directions here to give you an opportunity to decide what will work the best here to start describing the Sweet hierarchy: an interface or an abstract class?

b. Candy, Lollipop and Cookie - two other types from Sweet hierarchy.

c. Create class Present. Present has next behavior:

```
// the method filters sweets by sugar weight inclusively
public Sweet[] filterSweetsBySugarRange(double minSugarWeight,
double maxSugarWeight) {
        <write your code here>
}

// the method calculates total weight of the present
public double calculateTotalWeight() {
        <write your code here>
}

// the method that adds sweet to the present
public void addSweet(Sweet sweet) {
        <write your code here>
}

// the method returns copy of the Sweet[] array so that nobody could
// modify state of the present without addSweet() method.
// Also array shouldn't contain null values.
public Sweet[] getSweets() {
        <write your code here>
}
```

N.B.: during implementation of these methods - use stream API

3. Implement console program which will meet the following requirements:
    a. Program starts and asks user to enter random words separated by space
    b. Program asks user to enter minimum length of string to filter words which were entered
    c. Program creates array object from entered words
    d. Program calls specific method which takes String[] as a parameter and returns array of strings which contains words that have length more or equal to value specified by user

Method should look like this:

```
public static String[] filterWordsByLength(int minLength, String[] words)
{
        <write your code here>
}
```

N.B.: during implementation of this method - use stream API