Introduction to Programming I

Lab 6

Alexey Shikulin
Furqan Haider
Munir Makhmutov
Sami Sellami

More into Java

Exercise 1

Create a class which implements a Calculator. The calculator should be able to perform the following operations:

- addition
- subtraction
- multiplication
- division

Your main method should be able to receive three parameters: first number, the operation symbol which should be performed (+, -, *, /) and the second number.

*Examples: 1 + 2; 2 * 4; 10 / 2*

Handle division to zero. If your program encounters such cases, return -1.

Your calculator should only work with positive integer numbers. However, it should be able to display float outcomes, if needed

Exercise 1: Solution (Operation enumeration)

```java
package ru.makhmutov.task1;

public enum Operation {
 ADD("Addition"),
 SUBTRACT("Subtraction"),
 MULTIPLY("Multiplication"),
 DIVIDE("Division");

 private final String op;

 /**
  * Constructor for Operation enum
  *
  * @param op Operation name
  */
 Operation(String op) {
   this.op = op;
 }

 /**
  * This is getter method for operation name
  *
  * @return Operation name
  */
 public String getOp() {
   return op;
 }
}
```

Exercise 1: Solution (Calculator class 1/3)

```java
package ru.makhmutov.task1;

import java.util.InputMismatchException;
import java.util.Locale;
import java.util.Scanner;

public class Calculator {

 private static final int UPPER_BOUNDARY = 1000;
 private static final int LOWER_BOUNDARY = 0;

 /**
  * The entry point of the Calculator program.
  * The program scans 2 positive integer values
  * and performs one of chosen operations
  *
  * @param args Array with parameters of the program
  */
 public static void main(String[] args) {
   seeOperations();
   performOperation();
 }

 /**
  * This method displays all operations
  * and their id numbers
  */
 private static void seeOperations() {
   System.out.println("Possible operations: ");
   for (Operation operation : Operation.values()) {
     System.out.format("%d. %s%n", (operation.ordinal() + 1),
       operation.getOp());
   }
 }

 /**
  * This method allows insertion of operation type,
  * two integer values and performing the operation
  */
 private static void performOperation() {
   try (Scanner scanner = new
Scanner(System.in).useLocale(Locale.ENGLISH))
   {
     Operation operation = scanOperation(scanner);
     int value1;
     int value2;
     value1 = scanNumber(scanner, LOWER_BOUNDARY, UPPER_BOUNDARY,
"first");
     value2 = scanNumber(scanner, LOWER_BOUNDARY, UPPER_BOUNDARY,
"second");
     switch (operation) {
       case ADD:
         System.out.println(value1 + value2);
         break;
       case SUBTRACT:
         System.out.println(value1 - value2);
         break;
       case MULTIPLY:
         System.out.println(value1 * value2);
         break;
       case DIVIDE:
         if (value2 != 0) {
           System.out.println(value1 / value2);
           break;
         } else {
           System.out.println("Error: Division by 0");
           System.exit(-1);
         }
       default:
         System.out.println("Wrong operation chosen!");
```

Exercise 1: Solution (Calculator class 2/3)

```
    /**
     * This method allows reading valid input integer values.
     *
     * @param scanner         The object of Scanner class needed
     *                        for scanning the number
     * @param lowerBoundary   The lower boundary for input value
     * @param upperBoundary   The upper boundary for input value
     * @return The obtained value received via scanning
     */
    private static int scanNumber(Scanner scanner, int lowerBoundary, int upperBoundary, String type) {
        int value = -1; // initially out of allowed boundaries
        boolean validityFlag;
        System.out.print("\nType the " + type + " value: ");
        do {
            try {
                value = scanner.nextInt();
            } catch (InputMismatchException e) {
                System.out.print("\nUse only integer value, try one more time: ");
                validityFlag = false;
                scanner.next();
                continue;
            }
            if (lowerBoundary == upperBoundary && value < lowerBoundary) {
                System.out.format("%nPlease enter the number not lower than %d: ", lowerBoundary);
                validityFlag = false;
            } else if (lowerBoundary != upperBoundary && (value < lowerBoundary || value > upperBoundary))
{
                System.out.format("%nPlease enter the number in range [%d; %d]: ", lowerBoundary,
upperBoundary);
                validityFlag = false;
            } else {
                validityFlag = true;
            }
        } while (!validityFlag);
        return value;
    }
```

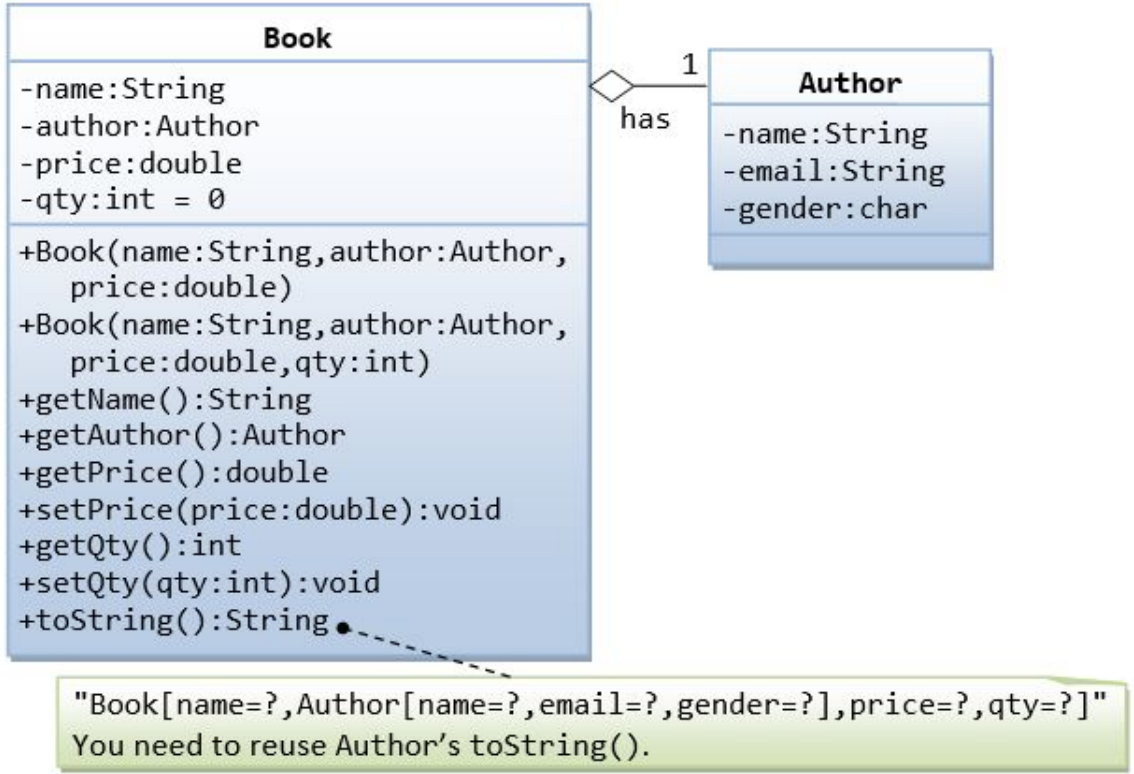
Exercise 1: Solution (Calculator class 3/3)

```
  /**
   * This method allows to scan the operation number
   *
   * @param scanner The object of Scanner class needed
   *                for scanning the number
   * @return The obtained value received via scanning
   */
  private static Operation scanOperation(Scanner scanner) {
    double value = 0.5; // the initial value should not be integer to be a barrier for the program
    boolean inserted = false;
    System.out.print("Type the operation number: ");
    do {
      try {
        value = scanner.nextDouble();
      } catch (InputMismatchException e) {
        System.out.print("\nDo not enter characters, try one more time: ");
        scanner.next();
        continue;
      }
      for (Operation operation : Operation.values()) {
        if (value == (operation.ordinal() + 1)) {
          inserted = true;
          break;
        }
      }
      if (!inserted) {
        System.out.print("\nPlease insert applicable values: ");
      }
    } while (!inserted);
    return Operation.values()[(int) (value - 1)];
  }
}
```

Exercise 2

Create two classes Book and Author. The Author class should contain properties such as name, email, gender. The Book class should contain name, author, price, quantity (qty). Use encapsulation

Create third class Main, which creates array of Books, updates Books and displays information about them



Exercise 2: Solution (Book class)

```java
package ru.makhmutov.task2;

public class Book {

 private final String name;
 private final Author author;
 private double price;
 private int quantity = 0;

 public Book(String name, Author author, double price) {
   this.name = name;
   this.author = author;
   this.price = price;
 }

 public Book(String name, Author author, double price, int quantity) {
   this.name = name;
   this.author = author;
   this.price = price;
   this.quantity = quantity;
 }

 public String getName() {
   return name;
 }

 public Author getAuthor() {
   return author;
 }

 public double getPrice() {
   return price;
 }

 public int getQuantity() {
   return quantity;
 }
```

```java
 public void setPrice(double price) {
   if (price > 50) {
     this.price = price;
   } else {
     System.out.println("Error: to low price");
     System.exit(2);
   }
 }

 public void setQuantity(int quantity) {
   if (quantity > 0) {
     this.quantity = quantity;
   } else {
     System.out.println("Error: non-positive amount");
     System.exit(3);
   }
 }

 @Override
 public String toString() {
   return "Book{" +
     "name='" + name + '\'' +
     ", author=" + author +
     ", price=" + price +
     ", quantity=" + quantity +
     '}';
 }
}
```

Exercise 2: Solution (Author class)

```
package ru.makhmutov.task2;

public class Author {

 private String name;
 private String email;
 private char gender;

 public Author(String name, String email, char gender) {
   this.name = name;
   this.email = email;
   this.gender = gender;
 }

 public String getName() {
   return name;
 }

 public void setName(String name) {
   this.name = name;
 }

 public String getEmail() {
   return email;
 }

 public void setEmail(String email) {
   this.email = email;
 }

 public char getGender() {
   return gender;
 }

 public void setGender(char gender) {
   if (gender == 'M' || gender == 'F') {
     this.gender = gender;
   } else {
     System.out.println("Error: non-existing gender");
     System.exit(1);
   }
 }

 @Override
 public String toString() {
   return "Author{" +
     "name='" + name + '\'' +
     ", email='" + email + '\'' +
     ", gender=" + gender +
     '}';
 }
}
```

Exercise 2: Solution (Main class 1/3)

```
package ru.makhmutov.task2;

import java.util.Random;

public class Library {

 private static final String[] AUTHORS = {"Alexander Pushkin", "Sergey Esenin", "Nikolai Gogol", "Lev Tolstoi", "Jack London", "Salikh Saidashev"};
 private static final String[] EMAILS = {"a.pushkin@innopolis.ru", "best.author.ever@gmail.com", "rifmoplet@mail.ru", "call_me_boss@innopolis.ru"};
 private static final String[] BOOKS = {"Peace & War", "Eugene Onegin", "Dead Souls", "Thinking in Java"};
 private static final double MIN_PRICE = 50;
 private static final double MAX_PRICE = 5000;
 private static final int MIN_QUANTITY = 1;
 private static final int MAX_QUANTITY = 1000;

 /**
  * The entry point of the Library program.
  * It allows adding books to the library
  * and then display them
  *
  * @param args Array with parameters of the program
  */
 public static void main(String[] args) {
     Book[] books = getBooks();
     displayBooks(books);
 }

 /**
  * This method allows getting the array of
  * generated books
  *
  * @return the array of books
  */
 private static Book[] getBooks() {
     Book[] books = new Book[3];
     for (int bookNo = 0; bookNo < books.length; bookNo++) {
         books[bookNo] = addBook();
     }
     return books;
 }
```

Exercise 2: Solution (Main class 2/3)

```java
/**
 * This method allows displaying all the books
 *
 * @param books the array of books
 */
private static void displayBooks(Book[] books) {
    for (Book book : books) {
        System.out.println(book.toString());
    }
}

/**
 * This method allows generating book for
 * further adding it to the array of books
 *
 * @return the generated book
 */
private static Book addBook() {
    Author author = addAuthor();
    return new Book(generateString(BOOKS), author,
            generatePrice(MIN_PRICE, MAX_PRICE), generateQuantity(MIN_QUANTITY, MAX_QUANTITY));
}

/**
 * This method allows generating author for
 * further linking it to the book
 *
 * @return the generated author
 */
private static Author addAuthor() {
    return new Author(generateString(AUTHORS), generateString(EMAILS), 'M');
}
```

Exercise 2: Solution (Main class 3/3)

```
    /**
     * This method allows randomly choosing the
     * string from the given array
     *
     * @param stringArray the input String array
     * @return the chosen string
     */
    private static String generateString(String[] stringArray) {
        Random random = new Random();
        return stringArray[random.nextInt(stringArray.length)];
    }

    /**
     * This method allows generating the price
     *
     * @param lowerBoundary the lower boundary of a price
     * @param upperBoundary the upper boundary of a price
     * @return the generated price
     */
    private static double generatePrice(double lowerBoundary, double upperBoundary) {
        Random random = new Random();
        return lowerBoundary + random.nextInt((int) (upperBoundary - lowerBoundary));
    }

    /**
     * This method allows generating the quantity
     * of books
     *
     * @param lowerBoundary the lower boundary of a quantity
     * @param upperBoundary the upper boundary of a quantity
     * @return the generated quantity of books
     */
    private static int generateQuantity(int lowerBoundary, int upperBoundary) {
        Random random = new Random();
        return lowerBoundary + random.nextInt(upperBoundary - lowerBoundary);
    }
}
```

Exercise 3

Create a class Time which receives the time in hours, hours and minutes, hours minutes and seconds. This class instances should be created from another class TimeCreator. Create 2 instances of Time and calculate the difference between these times. Time class should have a method to advance the time by one second, and return this instance to support chaining.

```
Time t;
….
t.inc().inc().inc();
```