1DV532 – Assignment 1

- The assignment comprises seven exercises and **passing all the seven exercises is mandatory** to pass the assignment.
- Your submission will receive a grade from A to F where F is Fail.
- You are allowed to improve your work after the initial submission before the deadline.
- Passing grade is final; you will not get an opportunity to correct/improve after grading.
- Collaboration and discussions are encouraged, but each student must report **individually! You are not allowed to copy code from other students, books articles, blogs, wikis or any other source!** Each submission will pass through a plagiarism/clone detection system before correction. **If plagiarism is detected, the assignment will be failed and a formal investigation will be initiated.**

Preliminaries (Basic steps to structure and organize your code)

1. **Install Java**
   Download and install Java SE JDK: www.oracle.com/technetwork/java/javase/downloads. We recommend you to use the latest stable version. Also, there are plenty of instruction videos available in YouTube. Just search for "Install Java X" where X is your operating system.

2. **Install Eclipse (you may use any other IDE of your choice)**
   Download and install Eclipse IDE for Java Developers: http://www.eclipse.org/downloads/. Once again, there are plenty of instruction videos available in YouTube. Just search for "Install Eclipse X" where X is your operating system.

3. **Setup Eclipse Workspace**
   Before you start programming, do the following.
   - Create an Eclipse *workspace* (a folder) with the name `java_courses` on some location in your home directory.
   - Create a *Java project* with the name `1D532` inside the workspace.
   - Create a *package* with the name `YourLnuUserName_assign1` inside the project. For example, it might look something like `wo222ab_assign1`.
   - Save all program files from the exercises in this assignment inside the package `YourLnuUserName_assign1`.
   - In the future, create a new package (`YourLnuUserName_assignX`) for each assignment and a new project (with the course code as name) for each new course using Java.

Department of Computer Science and Media Technology, SE-351 95 Växjö • Email *nadeem.abbas@lnu.se* •

Exercises

1. Write a program **Print.java** that prints the phrase *Write once, run everywhere!*
   - on one line,
   - on four lines, one word on each line,
   - inside a rectangle made up by the characters *

2. Write a program **Numbers.java** that asks the user to enter a three-digit integer number. The program then calculates and prints, the sum and product of the three digits the integer consists of.

   Below is an example of the program execution:

```
Enter a three-digit integer number: 483
Sum of the digits of the integer number is: 15.
Product of the digits of the integer number is: 96
```

3. An ISBN-10 (International Standard Book Number) consists of 10 digits: $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}$. The last digit, d10, is a checksum, which is calculated from the other nine digits using the following formula:
   (d1 * 1 + d2 * 2 + d3 * 3 + d4 * 4 + d5 * 5 + d6 * 6 + d7 * 7 + d8 * 8 + d9 * 9) % 11

   Note: % represents the modulus operator

   If the checksum is 10, the last digit is denoted as X according to the ISBN-10 convention.

   Write a program **ISBN.java** that prompts the user to enter the first 9 digits and displays the 10-digit ISBN (including leading zeros). Your program should read the input as an integer. Below are two sample runs of the program:

   Run 1:

```
Enter the first 9 digits of an ISBN as integer: 013601267
The ISBN-10 number is: 0136012671
```

   Run 2:

```
Enter the first 9 digits of an ISBN as integer: 013031997
The ISBN-10 number is 013031997X
```

4. An Armstrong number is an n-digit number that equals the sum of the n^{th} power of its digits. For example 153 is a three-digit number where the sum of the cubes of the individual digits (1 + 125 + 27) equals the number itself (153).
   Write a program **ArmstrongNumber.java** that prompts user to enter a range for Armstrong numbers. The range is entered by asking user to enter a *starting* and an *ending* number for the range. The program then computes and prints Armstrong number, if any, in the entered range. Next, the program should prompt for a new range until the user decides that she or he is through. Use variables of the type integer to store the start and end numbers of the range.

Below is an example of the program execution:

```
Enter the starting number of the range :100
Enter the ending number of the range :1000

The  Armstrong numbers between the given range are :
153
370
371
407
Do you want to repeat? (Y/N) : Y

Enter the starting number of the range :200
Enter the ending number of the range :300

The  Armstrong numbers between the given range are :
Do you want to repeat? (Y/N) : N
```

5. Write a program **Days.java** that prompts user to enter two non-negative integer values:
   - The first value is for today's day of the week and must be from 0 to 6, for today's day of the week. The program should display an appropriate error message if the entered value is not from 0 to 6.
   - The second value is to compute the future day. The future day is computed by adding the second value to the first value.
   - The program should then determine and print today's day of the week and the future day of the week based on codes as follows: Sunday is 0, Monday is 1, …, and Saturday is 6.

   Below are the outputs from two example executions of the program:

   Run 1:

```
Enter today's day: 1
Enter the number for the future day: 3
Today is Monday and the future day is Thursday
```

   Run 2:

```
Enter today's day: 0
Enter the number for the future day: 31
Today is Sunday and the future day is Wednesday
```

6. The Babylonian algorithm to compute square root of a positive number *n* is as follows:
   1. Make a guess at the answer (you can pick n/2 as your initial guess).
   2. Compute r = n / guess.
   3. Set guess = (guess + r) / 2.
   4. Go back to step 2 until the last two guess values are within 1% of each other.

   Write a program **SquareRoot.java** that prompts user to enter an integer value for *n*, iterates through the Babylonian algorithm until the guess is within 1% of the previous guess and outputs the answer as a real number to two decimal places.

Below is an example of the program execution:

```
This program estimate square roots.
Enter an integer to estimate the square root of: 25
Current guess: 7.25
Current guess: 5.349137931034482
Current guess: 5.011394106532552
Current guess: 5.000012953048684

The estimated square root of 25 is   5.00
```

7. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

   Write a program **Fibonacci.java** that prints a Fibonacci sequence starting with 1 and 2 and whose values do not exceed one thousand. The program should also compute and print a sum of the odd-valued terms in the required Fibonacci sequence.