Game Lab 1

Learning goals:

- Writing, compiling and running simple java program
- Printing a message to the screen
- Writing a comment to explain your code
- Using proper indentation

Part 1 (output):

In the old days, fancy graphics were not a part of games yet. And with the current prices of Graphics cards we may yet see the rise of text-based adventures. A famous example of such a game is Zork I: The Great Underground Empire
(try it here: https://playclassic.games/games/adventure-dos-games-online/play-zork-great-underground-empire-online/play/). The user is presented with an interactive story and not much more. Actions are taken by writing “what you want to do” in the game.

During this course you are going to write a (simple) text-based adventure game. Every week you will add what you have learned to this game. At the end of the course you will have a fully playable text-adventure called “Jerry’s Adventure”.

First! Our game needs a title, and you should be credited as the sole author of your game!

So for this week’s assignment first, print the name of your game followed by a blank line and “By: “ followed by your name. Your output should look like this:

```
       __                     _          ___       __                 __
      / /__  ____________  _( )____   /   | ____/ /   _____  ____  / /___  _________
 __  / / _ \/ ___/ ___/ / / ///  ___/  / /| |/ __  / | / / _ \/ __ \/ __/ / / / ___/ _ \
/ /_/ /  __/ /  / /  / /_/ / (__  )  / ___ / /_/ /| |/ /  __/ / / / /_/ /_/ / /  /  __/
\____/\___/_/  /_/   \__, / /____/  /_/  |_\__,_/ |___/\___/_/ /_/\__/\__,_/_/   \___/
                    /____/

by: your name
```

but replace *`your name`* with your own name.

1. Download the empty `Game.java` file from the portal. You can only write code between the lines marked:
   `// ----- Write your code below`
   …
   `// ----- Write your code above`
   Between these lines you can write anything you like as long as it completes the exercise. (If you like you may delete these comments before you hand in your file).

The \ character has a special meaning in Java! This means you have to “escape” it using another \ right in front of it. Get the output exactly as above by making sure to include line-endings (\n), spaces and escapes in the Ascii-Art.

2. Write, compile and run your code.
3. Check if your program writes the correct response.
4. Write at least one comment in your source code explaining to whomever reads it what’s going on with this weird-looking garbled text.

Hint: if you are struggling to make a nice banner, use the code below as a reference:

[redacted]

Part 2 (input):

Learning goals:

- Using basic data types such as integers, doubles and booleans
- Using a method to perform tasks
- Printing a player's input to the screen

Of course, no game is complete without a user's input. We need to know what the user wants the character in the game to do. In order to do this, we need to collect input from the user as a way of communicating with your program.

Using your existing `Game.java` file from Part 1, we will add a functionality to ask for the user's name and birth year.

1. Ask for the player's name.
2. Print a message greeting him/her.
3. Extend your program to include the user's age calculation.
    a. What data type do you think the birth year should be?
    b. Make sure you use the scanner variable (and only that variable) for user input.
    c. So, first we ask for the player's birth year. What is the data type returned by the `Scanner` class' `nextLine()` method? What about `nextInt()`?
    d. Using the user's birth year, calculate their age.
    e. Then tell them what age they are.

    Here are two example runs of the program:

```
       __                    _          ___       __                 __
      / /__  ______________ _( )_____   /   | ____/ /   _____  ____  / /___  __________
 __  / / _ \/ ___/ ___/ / / /// ___/  / /| |/ __  / | / / _ \/ __ \/ __/ / / / ___/ _ \
/ /_/ /  __/ /  / /  / /_/ / (__  )  / ___ / /_/ /| |/ /  __/ / / / /_/ /_/ / /  /  __/
\____/\___/_/  /_/   \__, / /____/  /_/  |_\__,_/ |___/\___/_/ /_/\__/\__,_/_/   \___/
                    /____/

by: your name

What is your name?: Tom
Hi Tom!
When were you born?: 1983
You are 40 years old.
```

Another example:

```
       __                     _          ___       __                 __
      / /__  ______________  ( )_____   /   | ____/ /   _____  ____  / /___  __________
 __  / / _ \/ ___/ ___/ / / /|// ___/  / /| |/ __  / | / / _ \/ __ \/ __/ / / / ___/ _ \
/ /_/ /  __/ /  / /  / /_/ /  (__  )  / ___ / /_/ /| |/ /  __/ / / / /_/ /_/ / /  /  __/
\____/\___/_/  /_/   \__, /  /____/  /_/  |_\__,_/ |___/\___/_/ /_/\__/\__,_/_/   \___/
                    /____/

by: your name

What is your name?: Pim
Hi Pim!
When were you born?: 2012
You are 11 years old
```

4. We may want to add some functionality to our age calculation later. So, how about we put this code inside a separate method?

   In your program, write a new method called `calculateAge.` Define the start of the method as follows:

   `public static int calculateAge` ….

   This method has a parameter `int birthYear` and it returns an `int` that represents the player's age. Complete the method where the … start and call it from the main method. Make sure the age **calculation** is now performed ***only*** by the `calculateAge` method, so you now have to change your `main` method in order to restore the output!

   From your `main` method, call the `calculateAge` method, store it in a variable and make sure your output still matches the output shown in step 1.

   Hint: you can calculate the age of the user "by hand" using a hardcoded value for the current year and the birth year provided. However, to make your implementation more flexible, you can use classes/methods such as:
   ```
   Year.now().getValue();
   ```

5. Now the game starts, we should tell the user where they are and ask what they want to do:

```
       __                     _         ___       __                 __
      / /__  ______________  _( )_____  /   | ____/ /   _____  ____  / /___  __________
 __  / / _ \/ ___/ ___/ / / /// ___/ / /| |/ __  / | / / _ \/ __ \/ __/ / / / ___/ _ \
/ /_/ /  __/ /  / /  / /_/ /  (__  ) / ___ / /_/ /| |/ /  __/ / / / /_/ /_/ / /  /  __/
\____/\___/_/  /_/   \__, /  /____/ /_/  |_\__,_/ |___/\___/_/ /_/\__/\__,_/_/   \___/
                    /____/

by: your name

What is your name?: Pim
Hi Pim!
When were you born?: 2012
You are 10 years old

You are standing in an abandoned university office. There are neither students
nor teachers around you. There's a table in front of you with various papers,
pens, a small puzzle toy, and a calculator.
A large window shows an empty office building; there are no Zombies in the empty
building (as far as you can tell). Behind you is a dark and mysterious door that
leads to a well-lit corridor with a fireproof ceiling and floor. You feel a
sense of Wi-Fi around you, the grinding of an LCD operated coffee machine can be
heard in the distance. You are not thirsty, but you rather have a craving for
justice.

What would you like to do?: Open the door
Open the door
```

6. Add the output to the program, next, ask the user for an action and store it in a variable of type `String`. Print the content of this variable as confirmation.
   If your program immediately exits without waiting for the user to input the action, check this article: https://stackoverflow.com/questions/13102045/scanner-is-skipping-nextline-after-using-next-or-nextfoo
   Using `nextInt()` of the `Scanner` class does not actually read the line-end after reading the input, so you need another `nextLine()` immediately after calling `nextInt()` to "consume" the line-end.

7. Write, compile and run your code. Take a breath...

8. Check if your program writes the correct response.

If not, go through your code and check if your assumptions are correct. Try to understand the state of your program, its variables and what is being printed after each line. Make a "mental map" of your program and try to run it in your mind. It is time to debug!

9. Write comments in your source code so everyone knows what's going on.

10. Once you are happy with your result and it looks like the output shown above, save your code to a file named `Game.java` and store it in a safe location. You will need it in the upcoming tutorials!